REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie

UNIVERSITE DE YAOUNDE I

Faculté des sciences

CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIE ET GEOSCIENCES

UNITE DE RECHERCHE ET DE FORMATION DOCTORALE EN MATHEMATIQUES, INFORMATIQUE BIOINFORMATIQUE ET APPLICATIONS

REPUBLIC OF CAMEROON
Peace-Work-Fatherland

UNIVERSITY OF YAOUNDE I

Faculty of science

POSTGRADUATE SCHOOL OF SCIENCE, TECHNOLOGY & GEOSCIENCES

RESEARCH & TRAINING UNIT FOR DOCTORATE IN MATHEMATICS, COMPUTER SCIENCES AND APPLICATIONS

LABORATOIRE D'INFORMATIQUE ET APPLICATIONS
LABORATORY OF COMPUTER SCIENCE AND APPLICATIONS

Une approche basée sur les machines volontaires pour le calcul haute performance

Thèse présentée et soutenue publiquement le 16 avril 2021 en vue de l'obtention du **Doctorat/PhD en Informatique**

Par :
ADAMOU HAMZA
DEA en Informatique

Devant le Jury composé de :

Président :	FOUDA NDJODO Marcel, Professeur, Université de Yaoundé I
Rapporteurs :	TCHUENTE Maurice, Professeur, Université de Yaoundé MEHAUT Jean-François, Professeur, Université de Grenoble Alpes, France
Membres :	SAKHO Ibrahima, Professeur, Université de Lorraine, France EMVUDU WONO Yves, Professeur, Université de Yaoundé I NDOUNDAM René, Maître de Conférences, Université de Yaoundé I DJOTIO Thomas, Maître de Conférences, Université de Yaoundé I

2021

REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie
- . - . - . -
UNIVERSITÉ DE YAOUNDÉ I
Faculté des Sciences
Département d'Informatique
B.P. 812 Yaoundé

REPUBLIC OF CAMEROON
Peace – Work – Fatherland
- . - . - . -
UNIVERSITY OF YAOUNDÉ I
Faculty of Sciences
Department of Computer Science
P.O.Box 812 Yaoundé

ATTESTATION DE CORRECTION DE LA THÈSE DE DOCTORAT/PHD

Nous soussignés, Professeurs **FOUDA NDJODO Marcel, TCHUENTE Maurice** et **NDOUNDAM René**, membres du jury de la thèse de Doctorat/PhD présentée par Monsieur **ADAMOU HAMZA**, matricule **03Y398**, thèse intitulée :

« **Une approche basée sur les machines volontaires pour le calcul haute performance** » et soutenue en vue de l'obtention du diplôme de **Doctorat/PhD en Informatique**,

Attestons que toutes les corrections demandées par le jury de soutenance en vue de l'amélioration de ce travail, ont été effectuées.

En foi de quoi, la présente attestation lui est délivrée pour servir et valoir ce que de droit./.

Président	**Rapporteur**	**Examinateur**
Pr. FOUDA NDJODO Marcel	**Pr. TCHUENTE Maurice**	**Pr. NDOUNDAM René**

Dédicaces

Je dédie ce travail à :

- mes parents
- mon épouse et mes enfants
- ma famille
- ma belle-famille

Remerciements

La réalisation de ce travail a nécessité la contribution de plusieurs personnes et institutions ; je tiens à leur exprimer toute ma gratitude.

Je remercie en tout premier lieu mon directeur, le Professeur Maurice TCHUENTE, pour m'avoir guidé depuis mon DEA et tout au long de cette thèse.

Je remercie ensuite le professeur Jean-François Méhaut pour avoir codirigé une partie de mes travaux, pour m'avoir accueilli au Laboratoire Informatique de Grenoble lors de plusieurs séjours de recherche et pour m'avoir donné accès aux plateformes de calcul grenobloise et française. Merci au professeur Christophe Cérin d'avoir inspiré les travaux sur la question de la prédiction de disponibilité des ressources dans le calcul sur machines volontaires.

Je remercie les universités de Ngaoundéré et de Yaoundé 1 pour avoir assurés ma formation dans l'enseignement supérieur. Je remercie en particulier les facultés des sciences et les départements d'Informatique de ces universités.

Je remercie l'ensemble du personnel des départements d'informatique et de mathématiques de l'université de Yaoundé dans lesquels j'ai mené cette thèse. Vous avez tous été d'un grand apport dans l'aboutissement de ce travail, je vous en suis infiniment reconnaissant. Je remercie également les étudiants.

Je remercie en particulièrement mes collègues le Dr Jiomekong Fidel pour le travail sur les aspects liés au développement logiciel et le Dr Nzekon Armel sur les systèmes de recommandation.

Je remercie le professeur Djuidjé Germaine du département de physique pour tous ces riches échanges sur les problèmes de calcul rencontrés dans le domaine de la physique. Merci pour le partage d'expérience sur la caractérisation du phénomène de la résonance stochastique.

Je remercie le personnel du CUTI ainsi que son directeur d'avoir accepté de participer aux expérimentations en tant que volontaires.

Je remercie mes amis du Cameroun et du monde entier pour leur soutien sans faille. Merci également aux différentes associations dont je suis membre pour les encouragements et leurs prières.

Je remercie tous ceux que je n'ai pas encore remercié et qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Merci.

Résumé

Les systèmes de calcul haute performance ou HPC visent à résoudre les problèmes qui nécessitent une puissante de calcul impossible à obtenir sur une plateforme de calcul séquentiel dans un temps raisonnable. Les solutions basées sur les supercalculateurs, les grappes, les grilles ou le cloud ne sont pas généralement à la portée de tous en raison des coûts d'accès, de mise en place ou des procédures administratives. Dans ce travail, nous proposons d'exploiter le temps libre des machines dites volontaires pour constituer une plateforme HPC à très faible coût adaptée aux environnements à ressources limitées que l'on retrouve en majorité dans les pays en voie de développement comme le Cameroun. Cette approche repose sur une architecture hybride dans laquelle le serveur joue le rôle de répertoire global d'informations et un ensemble de machines agissant comme des volontaires et des clients. Le principal avantage de cette approche est qu'elle ne nécessite aucun investissement supplémentaire en matériel, mais repose sur les machines volontaires. Il permet donc de garantir l'autonomie et la quasi-gratuité. Il convient toutefois de noter que le partage des machines volontaires avec leurs propriétaires et le caractère instable de l'électricité et de l'Internet posent le problème de disponibilité des ressources de calcul. Une machine sur laquelle un calcul est lancé peut quitter le système à tout moment et, par conséquent, entraîner une perte de calcul ou de données. Les techniques de tolérance aux pannes, telles que la redondance ou le *checkpointing*, sont coûteuses en déploiement. Pour réduire ces pertes ainsi que les coûts des mécanismes de tolérance aux pannes, nous utilisons l'idée qui consiste à sélectionner les ressources les plus susceptibles d'être disponibles jusqu'à la fin de chaque calcul. Les méthodes d'apprentissage automatique sont utilisées pour construire des modèles de prédiction de disponibilité des machines. L'évaluation de trois prédicteurs (classifieur naïf de Bayes , régression de Lasso et perceptron multicouche) nous a permis de constater qu'aucun d'eux n'est meilleur que les autres en termes de taux moyen de prédiction. Pour tirer parti des avantages de chaque prédicteur, nous proposons un système de prédiction de disponibilité basé sur la sélection de modèles. Nous avons également mis en place un système HPC appelé VC_UY1 et basé sur les machines des enseignants et des étudiants de l'Université de Yaoundé I. Ce système est composé d'un serveur et de 164 volontaires. Il fournit une puissance théorique de 12 PFlops. Nous avons mené les expérimentations sur le problème mathématique de la multiplication des matrices de grande taille et sur le problème physique de la caractérisation de la résonance stochastique. Les résultats montrent que les systèmes de calcul sur machines volontaires peuvent fournir une puissance de calcul significative à un coût très faible. La notion de prédiction de disponibilité a été étendue aux systèmes de recommandation pour déterminer quels produits sont susceptibles d'être achetés à un moment donné. Les expérimentations menées sur trois jeux de données des systèmes de recommandation (MovieLens-2k, MovieLens-ls et Last.fm) ont montré que la prédiction d'achat des produits permet d'améliorer les résultats des recommandations.

Mots clés

Calcul haute performance, Environnements à ressources limitées, Calcul sur machines volontaires, Prédiction de disponibilité des ressources, Système de recommandation, Calcul matriciel, Résonance stochastique.

Abstract

High Performance Computing (HPC) systems aim to solve complex computing problems (in a short time) that are either too large for standard computers or would take too much time. Solutions based on supercomputers, clusters, grids and cloud are not suitable for everyone because their costs generally exceed funding (particularly for academics) and the administrative complexity of access creates a higher barrier. In this work, we propose to use idle computing resources of volunteers to build very low-cost HPC systems suitable for resource-poor settings found mainly in developing countries such as Cameroon. This approach is based on a hybrid architecture in which the server acts as a global information directory and a set of computers acting as volunteers and clients. This approach requires no additional hardware investment, but builds on devices already owned by users and their communities. It therefore guarantees autonomy and practically free access. It should be noted, however, that computing resources are generally shared with owners and with power and internet outages, the availability of volunteer computers cannot be guaranteed for any period of time. A volunteer on which a task is in running may become unavailable at any time without prior warning and, consequently, might have several negative effects, including loss of data or computation. Fault tolerance techniques, such as redundancy or *checkpointing*, increase significantly deployment cost. To reduce these losses, we use the idea of selecting the resources most likely to be available until the end of each task. Machine learning methods are used to build resource availability prediction models. The evaluation of three predictors (naive Bayes classifier, Lasso regression and multilayer perceptron) allowed us to observe that none of them is better than the others in terms of average prediction accuracy. To maintain the benefits of each predictor, we provide a resource availability prediction system based on model selection. We apply it for a classification method (the multilayer perceptron) and a linear regression method (the Lasso method). In addition to volunteer computing systems, we used the resource availability prediction model to improve the performance of recommendation systems. Finally, we have successfully implemented an HPC system called VC_UY1 based on the personal computers of students and lecturers at the University of Yaoundé I. This system consists of a server and 164 volunteer computers. It provides a theoretical power of 12 PFlops. We have conducted experiments on the mathematical problem of the multiplication of large matrices and on the physical problem of the characterization of stochastic resonance. The results show that volunteer computing systems can provide significant computing power at a very low cost. The resource availability prediction concept has been extended to recommendation systems to determine which products are likely to be purchased at a given time. Experiments conducted on three recommendation system datasets (MovieLens-2k, MovieLens-ls and Last.fm) have shown that product purchase prediction improves recommendation results.

Keywords

High-performance computing, Resource-poor settings, Volunteer computing, Resource availability prediction, Recommandation system, Matrix multiplication, Stochastic resonance.

Table des matières

Chapitre Un

Introduction générale

Les scientifiques et les industriels sont confrontés à des problèmes de complexité croissante, nécessitant une puissance de calcul toujours plus élevée et une grande capacité de stockage. La grande majorité de ces problèmes se retrouvent dans le calcul scientifique, impliquant la construction de modèles mathématiques et de techniques de résolution numérique. La simulation numérique, par exemple, rend possible l'étude de phénomènes de différente tailles et remplace un large éventail d'expériences coûteuses, longues, dangereuses voire impossible dans certains cas [1]. Pour ce faire, les systèmes de calcul haute performance ou *High Performance Computing* (HPC) sont utilisés [2, 3].

Ce chapitre délimite le contexte de notre travail et identifie la problématique associée. Le positionnement ainsi que les principales contributions sont décrits. À la fin, l'organisation de la suite du manuscrit est présentée.

1.1 Contexte et problématique

Le calcul haute performance est une source de progrès significatifs pour la société. Il y a des défis pour l'avenir dans presque tous les domaines. Cette section commence par présenter quelques exemples de besoins en calcul des simulations numériques. Ensuite, les systèmes HPC basés sur les machines dédiées sont introduits, suivis par les systèmes HPC basés sur les machines volontaires. À la fin la problématique de cette thèse est présentée.

1.1.1 Besoins en calcul

La simulation numérique est aujourd'hui omniprésente dans presque tous les domaines. Bien que l'expérimentation réelle demeure une étape essentielle de la recherche scientifique et de l'industrie, la simulation a remplacé un large éventail d'expériences. En climatologie, l'analyse de centaines de millions d'observations rend les prévisions météorologiques plus précises. En sismologie, il est possible de rejouer plusieurs fois un tremblement de terre, en simulant le trajet des ondes sismiques dans la croûte terrestre afin de saisir de manière détaillée le déroulement des évènements [3, 4]. En l'imagerie médicale, l'analyse des images du cerveau permet une cartographie approfondie des fonctions cérébrales, qui pourrait être utilisé pour mieux comprendre certaines pathologies et ensuite pour entrevoir des solutions pour développer des traitements. Des simulations sont également utilisées pour étudier l'évolution du coronavirus et les mécanismes pour limiter sa propagation [1]. Dans l'industrie, un secteur comme automobile utilise le prototypage virtuel où les voitures sont complètement simulées avant d'être construites. Cela permet de réduire les délais de mise en œuvre et de diminuer les coûts.

Quel que soit le domaine, la puissance de calcul que requiert l'exécution des simulations est conséquente et augmente de façon continue. Les modélisations doivent être toujours plus fines et calculées dans un temps raisonnable [1, 3, 5, 6]. L'apparition des premiers supercalculateurs équipés

1. `http://www.cnrs.fr/en/node/4635`

de puissants processeurs et d'une grande capacité de stockage a permis de résoudre les problèmes de simulation numérique à grande échelle. Le calcul parallèle permet de diviser les problèmes en tâches qui vont être exécutées sur les machines parallèles. La parallélisation permet de diminuer le temps d'exécution et de stocker de grandes masses de données. De nombreux problèmes industriels et scientifiques impliquent des calculs complexes à grande échelle qui, sans ordinateurs parallèles, prendraient des années, voire des décennies, pour être calculés. Il est plus que souhaitable d'avoir les résultats disponibles le plus tôt possible car pour de nombreuses applications, des résultats tardifs signifient souvent des résultats inutiles [7].

1.1.2 Calcul haute performance

Le calcul haute performance ou HPC est la solution à la forte demande de calcul [2, 8, 9]. Il utilise le calcul parallèle pour diviser les problèmes en tâches pouvant être traitées simultanément et combine la capacité de traitement des ordinateurs pour obtenir le résultat le plus rapidement possible.Les systèmes HPC peuvent être classés en quatre principales catégories [10, 11] : système de calcul haute performance basé sur superordinateurs (HPC superordinateurs), système de calcul haute performance basé sur grappes (HPC grappes), système de calcul haute performance basé sur grilles (HPC grilles) et système de calcul haute performance basé sur le cloud (HPC cloud).

Dans le passé, les machines utilisées pour construire les systèmes HPC étaient les superordinateurs dédiés, composés d'un grand nombre de processeurs [4, 12, 13]. Mais, avec la demande toujours croissante des applications, ces machines ne pouvaient plus offrir la puissance de calcul à un coût raisonnable [14]. Avec la disponibilité de machines de bureau de plus en plus puissantes et moins chères, plusieurs solutions basées sur la mutualisation des ressources ont été proposées : les grappes de calcul [15, 16], les grilles de calcul [17, 18] et le calcul haute performance sur nuage [19, 20]. Une grappe de calcul est composée d'ordinateurs performants, dédiés au calcul et connectés à un réseau local haute performance. Une grille de calcul est composée d'ordinateurs hétérogènes connectés à une échelle plus grande que les grappes. Un nuage de calcul utilise les centres de données repartis à travers le monde pour donner accès à la demande à plusieurs types de services sur Internet.

Dans les pays développés, le calcul haute performance a permis à des projets à grande envergure d'accéder à une puissance de calcul considérable [21], ce qui a conduit à des découvertes et des inventions, que ce soit dans les universités, les entreprises ou les gouvernements. Malheureusement, il est difficile à utiliser dans des contextes tels que les environnements à ressources limitées (dont la majorité sont les pays pauvres et en voie de développement) en raison des coûts d'accès, de mise en place ou encore de procédures administratives. Les grappes de calcul et le calcul sur nuage nécessitent des ressources financières souvent hors de portée de la plupart des institutions des pays en voie de développement. La complexité des procédures administratives permettant l'obtention des autorisations pour utiliser une grille de calcul est un obstacle majeur [20].

1.1.3 Calcul sur machines volontaires

Les solutions HPC basées sur les superordinateurs, les grappes, le cloud et les grilles nécessitent des investissements significatifs. Ces investissements nécessitent la mobilisation des capitaux, des coûts de fonctionnement en espace, en personnel, en énergie et en refroidissement qui sont loin d'être négligeables. Pour pallier cela, les systèmes de calcul haute performance basés sur les machines volontaires (*Volunteer Computing Systems* (VCS) ont vu le jour [22, 23, 24, 25]). Ces systèmes sont basés sur les ordinateurs appartenant à des particuliers et utilisés pour leurs activités quotidiennes. Ces ordinateurs de plus en plus puissants ne sont utilisés activement que pendant quelques heures par jour et restent inactifs la plupart du temps. L'idée est donc d'exploiter leur puissance de traitement et leur mémoire pendant leurs périodes d'inactivité. La principale différence entre ce système et les grilles est que les ressources proviennent des ordinateurs non dédiés, sous-utilisés et contrôlés par leurs propriétaires. En

plus, les VCS ne nécessitent aucun investissement matériel supplémentaire, mais repose sur des appareils déjà détenus par les utilisateurs et leurs communautés [26]. Aujourd'hui, plusieurs VCS offrent des PFlops de puissance de calcul [27].

1.1.4 Problématique

Dans les systèmes de calcul haute performance basés sur les machines volontaires, les ressources de calcul ne sont pas toujours homogènes. Ainsi, aucune politique de maintenance ne peut être assurée, car les ordinateurs sont détenus et contrôlés par les utilisateurs finaux et sont géographiquement dispersés [24, 25, 28, 29]. De plus, la nature instable de l'énergie électrique et de l'Internet peut entraîner plusieurs problèmes notamment une perte des résultats. Ainsi, l'exploitation des machines volontaires nécessite de répondre à plusieurs questions : comment recruter les volontaires? Quels volontaires recruter? Quel est l'ensemble des volontaires pouvant exécuter un calcul jusqu'à son terme? Pour exécuter un calcul jusqu'à son terme, la sélection des différentes machines pouvant être disponibles jusqu'à la fin du calcul doit être faite. Cette sélection peut être considérée comme un problème de prédiction de disponibilité des ressources [30, 31, 32, 33]. En effet, pour un calcul donné, trouver l'ensemble des machines susceptibles d'être disponibles jusqu'à la fin du calcul est un défi. Il est nécessaire de collecter les données sur la disponibilité des ressources de calcul sur une période et d'utiliser ces données pour ne choisir, pour un calcul donné, que les ressources susceptibles de l'exécuter jusqu'à la fin dans un temps raisonnable.

1.2 Positionnement et contributions

Dans cette thèse, nous nous intéressons au calcul haute performance à très faible coût. Il s'agit de résoudre les problèmes qui nécessitent une puissance de calcul qui ne peut être obtenue sur une machine de bureau dans un délai raisonnable.

L'objectif de cette thèse est de proposer une méthodologie de mise en place des systèmes de calcul haute performance basés sur les machines volontaires. Dans le processus de réalisation de cet objectif, nous avons abouti aux principales contributions suivantes :

Proposition d'un modèle de prédiction de disponibilité des ressources de calcul basée sur l'apprentissage automatique : notre première contribution est un système de prédiction basé sur la sélection de modèles. Un système de prédiction de disponibilité des ressources est utilisé pour rendre les calculs beaucoup plus fiables dans un système de calcul sur machines volontaires. Ce système permet de déterminer si oui ou non une ressource de calcul sera disponible dans une certaine durée de temps et est utilisé à chaque fois qu'un calcul veut être lancé. Des expériences ont été menées dans le but d'évaluer le classifieur naïf de Bayes, le perceptron multicouche et la régression Lasso en termes de taux de prédiction moyen sur les traces des machines volontaires du projet seti@home. Les résultats ont montré la dépendance de la qualité de la prédiction sur les valeurs des paramètres d'entrée (til et pil). La comparaison effectuée montre qu'aucun des prédicteurs choisis n'est meilleur que les autres en performance sur toutes les valeurs des paramètres d'entrée utilisés. Cela signifie qu'un bon taux de prédiction ne sera pas toujours obtenu à partir d'un modèle construit à l'aide d'un seul prédicteur. C'est pour tirer parti des avantages de chaque prédicteur, que nous avons proposé une approche de prédiction de disponibilité basée sur une combinaison de modèles.

Expérimentation de la prédiction de disponibilité dans les systèmes de recommandation : notre deuxième contribution a consisté à expérimenter la prédiction de disponibilité dans les systèmes de recommandation. Les systèmes de recommandation s'appuient généralement sur les informations sur les produits et les préférences des utilisateurs pour suggérer les produits les plus susceptibles d'intéresser un utilisateur. Cependant, dans de nombreux cas (par exemple, le système de recommandation

des vêtements), il serait important de prendre en compte les informations sur le contexte temporel (le moment où une opération est effectuée). La prise en compte du contexte temporel nécessite de répondre à la question suivante : un produit donné est-il susceptible d'être acheté à un moment donné ? Cette question a été formalisée comme un problème de prédiction de produits et les méthodes d'apprentissage automatique ont été utilisées pour construire un système de prédiction d'achat à partir des traces.

Proposition d'une architecture de système de calcul sur les machines volontaires : notre troisième contribution est une architecture hybride de système de calcul sur les machines volontaires. Cette architecture combine la flexibilité et l'évolutivité de l'architecture décentralisée avec la sécurité et la confiance de l'architecture centralisée. Avec cette architecture, les nœuds de calcul volontaires peuvent échanger directement entre eux et gérer les calculs sans l'intervention d'un serveur central. Le serveur agit comme un répertoire global d'informations.

Mise en place d'un environnement de calcul haute performance nommé VC_UY1 à l'Université de Yaoundé I : notre dernière contribution a consisté en la mise en place d'un système de calcul haute performance nommé VC_UY1. Il est composé d'un ensemble de machine volontaires, majoritairement issues des machines des enseignants et des étudiants et d'un logiciel permettant de soumettre les calculs, récupérer les résultats et collecter les données sur le système et sur les volontaires. Une première version du logiciel de gestion du système VC_UY1 a été développée et utilisée pour la multiplication des matrices de grande taille et la caractérisation de la résonance stochastique.

1.3 Structure du document

Ce mémoire est organisée en huit chapitres.

- **Le chapitre 2** présente un état des lieux du calcul haute performance. Il évoque le calcul parallèle, qui est la base du calcul haute performance dans un premier temps. Par la suite, il introduit les différents systèmes de calcul haute performance basés sur les machines dédiées, les systèmes HPC en Afrique et les métriques permettant l'évaluation des systèmes HPC. Ce chapitre se termine par un constat que les systèmes de calcul haute performance basés sur les machines dédiée sont difficiles à mettre en œuvre, difficile d'accès (coût) et donc inadaptées aux environnements à ressources limitées.
- **Le chapitre 3** est dédié aux systèmes de calcul haute performance basés sur les machines volontaires. Il commence par une vue générale de ces systèmes, se poursuit par la présentation des différentes architectures de ces systèmes, décrit la manière dont les ressources et les tâches sont gérées et se clôture par quelques exemples de système. Les systèmes de calcul sur machines volontaires offrent ainsi une plateforme simple d'utilisation, accessible au grand public, évolutive et peu coûteuse, ils sont donc par conséquent plus adaptés aux environnements à ressources limitées.
- **Le chapitre 4** propose une approche de prédiction de disponibilité des ressources de calcul basée sur l'apprentissage automatique. Dans un premier temps, les mécanismes de prédiction de disponibilité des ressources utilisant l'apprentissage et leurs limites sont présentés. Ensuite notre approche basée sur la sélection des méthodes d'apprentissage est décrite. Le chapitre se termine par un bilan.
- **Le chapitre 5** est consacré à une expérimentation de la prédiction de disponibilité dans les systèmes de recommandation. Il commence par des calculs statistiques sur la disponibilité des catégories de produits dans un système de recommandation en fonction de divers éléments du contexte temporel. Ensuite, le principe de la prévision des catégories de produits en fonction

du contexte temporel est présenté. L'expérimentation de la prévision de disponibilité dans les systèmes de recommandation est effectuée et le chapitre est termine par une conclusion.

- **Le chapitre 6** propose une approche de calcul haute performance basée sur les machines de volontaires prenant en compte les contraintes des environnements à ressources limitées. Il présente les environnements à ressources limitées et l'approche proposée pour construire les systèmes HPC dans ces environnements. VC_UY1, qui est l'implantation de cette approche au sein de l'Université de Yaoundé I, est y décrit.
- **Le chapitre 7** présente l'expérimentation du système VC_UY1 sur deux problèmes : le problème de la multiplication des matrices de grande taille et le problème de la caractérisation de la résonance stochastique. Les deux problèmes sont décrits en premier, suivis du protocole expérimental, du déroulement de l'expérimentation et des résultats obtenus. Une discussion conclut ce chapitre.
- **Le chapitre 8** termine cette thèse en passant en revue les différentes contributions. Il examine les résultats apportés et ouvre plusieurs pistes pour de futurs travaux de recherche.

Chapitre Deux

Calcul haute performance

Les systèmes de calcul haute performance ou HPC (*High Performance Computing*) [11] sont les systèmes qui permettent de résoudre rapidement les problèmes de calcul complexes qu'une machine ordinaire ne peut pas ou prendrait trop de temps à résoudre. Il utilise le principe du calcul parallèle pour diviser les problèmes en tâches qui peuvent être exécutées simultanément sur différentes unités de calcul. Au départ, les besoins en HPC étaient satisfaits par des infrastructures dédiées appelées supercalculateurs ou superordinateurs. Mais avec la complexité croissante des problèmes, ces machines ne pouvaient plus offrir une puissance de calcul à un coût raisonnable. Avec la disponibilité d'ordinateurs de plus en plus puissants et bon marché, l'évolution des technologies de mise en réseau à haut-débit et l'émergence de la virtualisation, plusieurs solutions basées sur la mutualisation des ressources de calcul ont vu le jour. En fonction du type de connexion réseau utilisé (Intranet, Internet, réseaux institutionnels, etc.), du type de ressources de calcul (machines dédiées hautes performances, serveurs de calcul, ordinateurs de bureau, ordinateurs portables, téléphones, etc.), de la nature des utilisateurs (communauté de scientifique, industrie, applications publiques/privées, etc.) et les applications qui seront exécutées (applications industrielles, recherche scientifique, etc.), les approches de mutualisation peuvent être classées en quatre catégories : les grappes de calcul, les nuages de calcul, les grilles de calcul et les systèmes de calcul sur machines volontaires. Dans ce chapitre, nous faisons un état des lieux du HPC. Tout d'abord, la section 2.1 introduit le principe du calcul parallèle, les sections 2.2, 2.3, 2.4, 2.5 présentent le HPC basé respectivement sur des superordinateurs, des grappes de calcul, des grilles de calcul et le cloud. Enfin, le chapitre se termine par les systèmes HPC déployés en Afrique, l'évaluation des systèmes HPC et un bilan dans les sections 2.6, 2.7 respectivement.

2.1 Calcul parallèle

La recherche dans le domaine du calcul haute performance visant à améliorer le temps de résolution des problèmes complexes a donné naissance au calcul parallèle [6, 8]. Le calcul parallèle est l'utilisation simultanée de plusieurs unités de traitement pour résoudre un problème plus efficacement. Pour ce faire, les programmes séquentiels sont divisés en tâches (ou encore unité de travail) indépendantes (on obtient un programme parallèle) de telle sorte que chaque unité de traitement puisse exécuter sa tâche simultanément avec les autres. Les unités de traitement peuvent être diverses et comprendre des ressources telles qu'un ordinateur composé de plusieurs processeurs ou cœurs (on parle alors de systèmes multiprocesseurs ou multicœurs), plusieurs ordinateurs indépendants connectés en réseau (on parle alors de système multi-ordinateurs) ou encore la combinaison des éléments mentionnés ci-dessus. Cette section présente le calcul parallèle, qui est la base des systèmes de calcul haute performance. Elle présente successivement la structure et la description d'un programme parallèle (sous-section 2.1.1), les architectures des systèmes parallèles (sous-section 2.1.2), les réseaux d'interconnexion utilisés par les systèmes parallèles (sous-section 2.1.3), les modèles et quelques outils de programmation parallèle (sous-section 2.1.4) et l'évaluation des systèmes parallèles (sous-section 2.1.5).

2.1.1 Structure des programmes parallèles

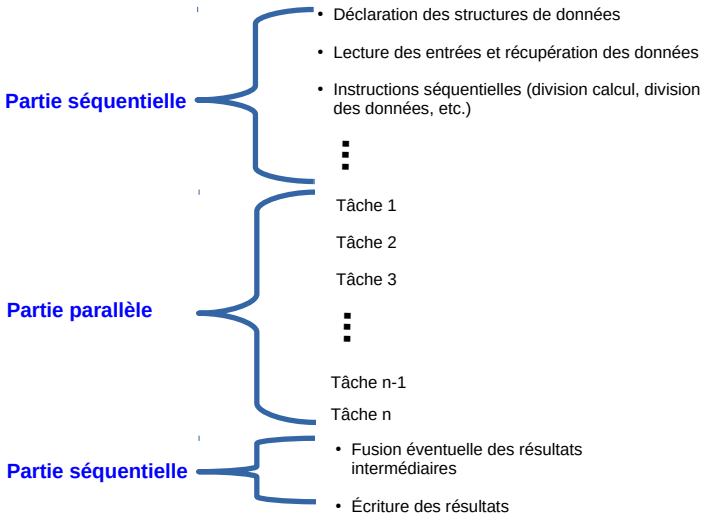

FIG. 2.1 – *Structure générale d'un programme parallèle*

En général, un programme parallèle peut être divisé en deux parties présentées par la figure 2.1 [34] : les parties intrinsèquement séquentielles qui permettent par exemple d'accepter les données d'entrée et les parties potentiellement parallèles qui permettent par exemple de faire des opérations indépendantes. L'objectif principal est de réduire le temps d'exécution en identifiant les zones de parallélisme potentiel et donc de décomposer le problème pour permettre une exécution parallèle. Cette décomposition doit se faire de façon efficace. En effet, un nombre très élevé de sous-problèmes représente un coût considérable car, outre le coût de décomposition et de combinaison des tâches (qui augmente avec le nombre de sous-problèmes), il faut prendre en compte la nécessité de communication entre les tâches tant pour transmettre les données que pour synchroniser l'exécution. Le meilleur niveau d'efficacité est obtenu en divisant le problème de telle sorte que les sous-problèmes soient uniformément répartis entre les unités de traitement disponibles. Par conséquent, Il faut choisir une granularité (taille des tâches) correcte lors du découpage du problème [6].

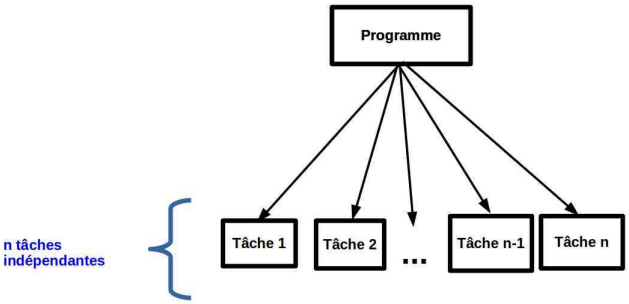

FIG. 2.2 – *Programme parallèle avec les tâches indépendantes*

Les tâches d'un problème peuvent être indépendantes (figure 2.2) ou dépendantes (figure 2.3) :

– Les tâches sont indépendantes lorsqu'elles sont exécutées sans avoir besoin des informations provenant des autres ;

– Les tâches sont dépendantes lorsqu'elles doivent coopérer en s'échangeant les données. Les données doivent donc être transférées entre les unités de calcul. Dans ce cas, une bonne gestion de la communication doit permettre de réduire les coûts induits par le temps d'attente d'une donnée par exemple. Lorsque plusieurs unités de traitement se partagent les données, la synchronisation permet de contrôler l'accès concurrent à ces données. Par exemple, certaines opérations doivent être effectuées dans un ordre particulier pour observer la dépendance. D'autres peuvent être exécutées dans un ordre arbitraire mais doivent être regroupées de sorte que certaines séquences s'exécutent de manière atomique, sans l'interférence des autres séquences.

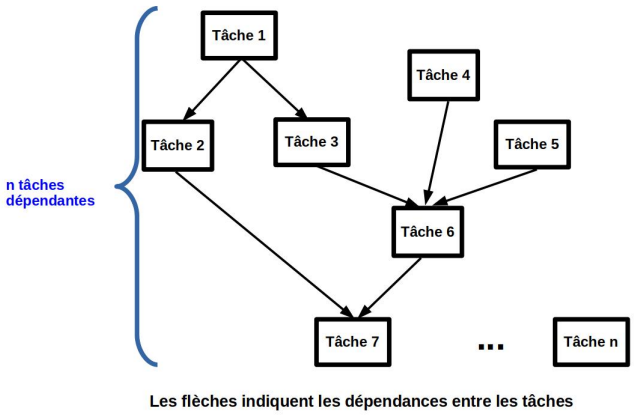

FIG. 2.3 – *Programme parallèle avec les tâches dépendantes*

Le découpage du calcul peut se faire au niveau des tâches (parallélisme des tâches), au niveau des données (parallélisme des données) ou encore au niveau des tâches et chaque tâche est partitionnée par données [6, 34] :

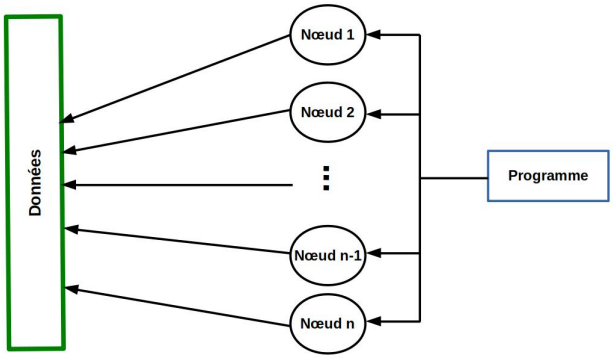

FIG. 2.4 – *Structure d'un programme parallèle implémentant le parallélisme de données*

– Le parallélisme des données (figure 2.4) : privilégie le développement du parallélisme par l'exécution de la même tâche (souvent le même programme) sur des données différentes. Cette approche est très fréquente où il est possible de diviser les données en différents fragments et d'effectuer un traitement identique et régulier sur les différents fragments.
– Parallélisme des tâches (figure 2.5) : forme de parallélisme dans laquelle différentes tâches sont

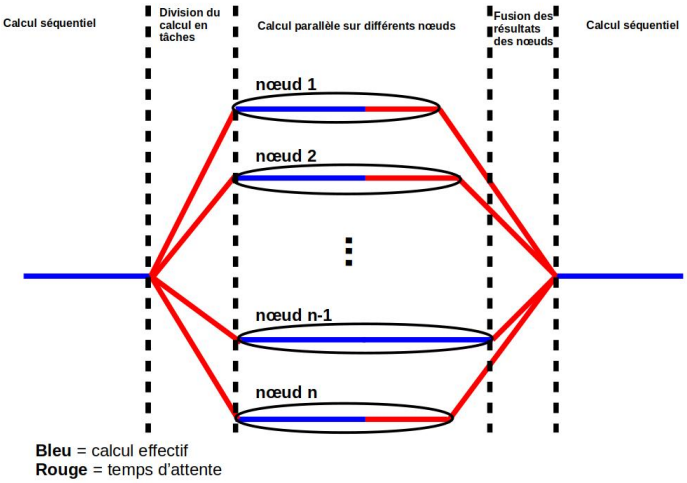

FIG. 2.5 – *Structure d'un programme parallèle implémentant le parallélisme de tâches*

exécutées en parallèle sur les mêmes données. Une des difficultés ici est d'assurer un bon équilibre des charges allouées à chaque unité de traitement.

La recherche dans le domaine du calcul parallèle a conduit au développement d'architectures de machines spécifiques et de réseaux de communication. Dans les sections qui suivent, ces architectures seront présentées.

2.1.2 Architectures des systèmes parallèles

Dans les systèmes parallèles, la coordination et la synchronisation entre les différentes unités de calcul dépendent de la manière avec laquelle les informations sont échangées entre ces unités de calcul et de l'organisation de la mémoire [3]. Une distinction est donc faite entre : les architectures à mémoire partagée, les architectures à mémoire distribuée et les architectures hybrides.

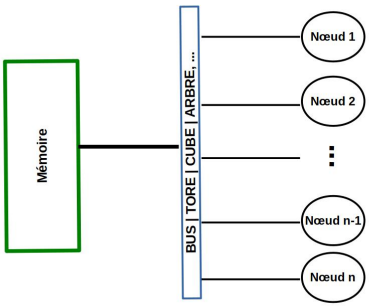

FIG. 2.6 – *Architecture des machines parallèles à mémoire partagée*

– Architectures à mémoire partagée (voir figure 2.6) : les architectures à mémoire partagée sont composées de plusieurs unités de calcul partageant une mémoire commune. La mémoire est connectée aux unités de calcul par un réseau à haut-débit. Pour accéder aux données, les unités de calcul utilisent le même espace d'adressage global. Chaque modification effectuée par une unité de calcul à un emplacement de la mémoire est visible par toutes les autres. Ces architectures sont

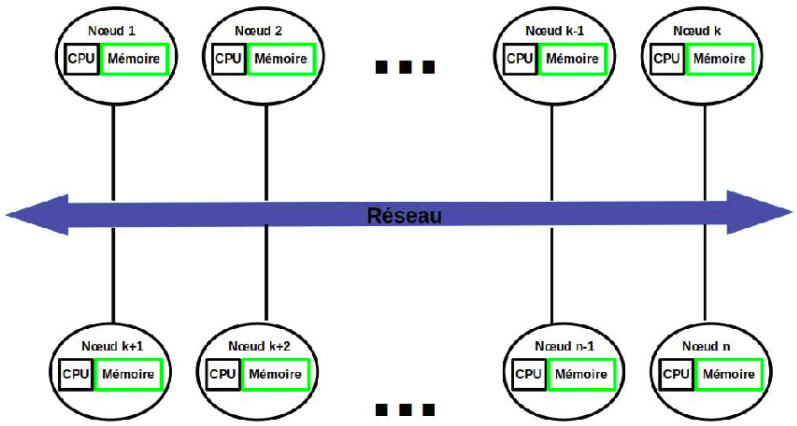

FIG. 2.7 – *Architecture des machines parallèles à mémoire distribuée*

simples à mettre en place. Cependant, ils posent un problème lorsque le nombre d'unités de calcul accédant simultanément aux données stockées en mémoire partagée devient trop important.

– Architectures à mémoire distribuée (figure 2.7) : les architectures à mémoire distribuée disposent de plusieurs unités de calcul accédant à leur propre mémoire locale séparément des autres. Une modification dans la mémoire locale d'une unité de calcul n'a pas d'effet sur la mémoire des autres. La communications se fait par l'intermédiaire d'un réseau. Bien qu'avec ces architectures la communication entre les unités de traitement soit plus lente qu'avec les systèmes à mémoire partagée, elle est facile à étendre.
– Architectures hybrides (figure 2.8) : les architectures dites hybrides combinent une architecture à mémoire partagée et une architecture à mémoire distribuée. Elles sont constituées de plusieurs nœuds multiprocesseurs ou multicœurs autonomes connectés en réseau.

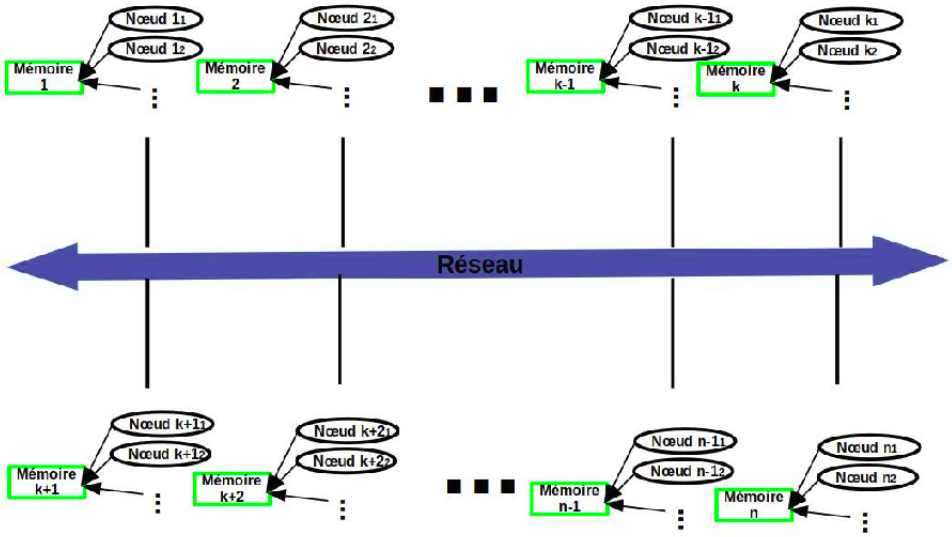

FIG. 2.8 – *Architecture des machines parallèles à mémoire hybride*

2.1.3 Réseau d'interconnexion des systèmes parallèles

Les réseaux d'interconnexion haute performance permettent de relier les unités de calculs et de prendre en charge le transfert de données entre les nœuds. Lorsqu'un nœud de calcul a besoin des

données provenant d'ailleurs, ces données lui sont transmises par le biais de messages échangés sur ce réseau [3, 34]. Ces réseaux sont généralement caractérisés par deux paramètres :

- La latence : temps nécessaire pour envoyer des données dans le réseau.
- La bande passante (exprimée en bits par seconde bps) : fréquence maximale à laquelle les données peuvent être envoyées sur le réseau.

Les réseaux haute performance sont à faible latence et à bande passante élevée. Ils utilisent des protocoles de communication tels que TCP/IP pour permettre aux nœuds de calcul de communiquer entre eux. Les technologies réseaux haute performance proposées sur le marché sont : SCI (*Scalable Coherent Interface*), Gigabit Ethernet, Myrinet et InfiniBand. Les deux principales technologies utilisées dans les machines haute performance d'après le TOP500 [1] de novembre 2020 sont les technologies Gigabit Ethernet et Infiniband.

2.1.4 Modèles et outils de programmation parallèle

Les modèles de programmation sont des vues abstraites de haut niveau des technologies et des applications qui cachent la plupart des détails architecturaux [35]. Les différents modèles de programmation fournissent les outils tels que les compilateurs spéciaux, les bibliothèques et les frameworks. Ces outils permettent de simplifier la programmation en masquant de nombreux détails sur l'exécution parallèle tels que le transfert et le routage des messages, l'allocation et la migration des tâches, etc. [35]. Une distinction est faite entre : les modèles à mémoire partagée, les modèles de transmission de messages et les modèles hybrides. Avec les modèles à mémoire partagée, les tâches s'exécutent en parallèle, communiquent en écrivant et en lisant dans une mémoire partagée. Toutes les unités de calculs accèdent à un même espace d'adressage global. Avec les modèles de transmission de messages, les unités de calcul s'exécutent en parallèle, communiquent et se synchronisent entre eux en envoyant et en recevant explicitement des messages. Les modèles hybrides combinent les deux modèles précédents.

Divers langages de programmation et bibliothèques permettent de réduire considérablement la complexité du développement de programmes sur les systèmes parallèles. Généralement, les applications sont programmées en utilisant plusieurs langages :

- Langages impératifs séquentiels tels que C et Fortran. Les extensions openMP du C et du Fortran, Cilk et TBB (*Thread Building Blocks*) permettent de programmer les nœuds composés de plusieurs processeurs avec une mémoire partagée. Lorsque les nœuds ne partagent pas la même mémoire, les librairies MPI (*Message Passing Interface*), PVM (*Parallel Virtual Model*) sont utilisées ;
- Langages dédiés au calcul parallèle comme *High-Performance Fortran* (HPF) et *Co-Array Fortran* basés sur Fortran ; UPC (*Unified Parallel C*) basé sur le langage C ; et Titane basé sur le langage Java ;
- Langages orientés objet tels que C++ et les frameworks tels que CCA (*Common Component Architecture*) sont également utilisés pour fournir une couche d'abstraction aux langages de programmation parallèle de bas niveau ;
- Langages de script comme Python et Perl sont utilisés pour lier les composants écrits dans les langages mentionnés plus haut. La bibliothèque IPython (*Interactive Python*), écrit en python permet d'écrire et de simuler les programmes parallèles.

2.1.5 Évaluation des systèmes de calcul parallèle

Depuis l'apparition des machines parallèles, de nombreuses méthodes ont été utilisées pour estimer leurs performances. Les principaux aspects évalués sont la puissance du système, le temps d'exécution des programmes, l'efficacité et l'évolutivité du système. Ces méthodes permettent de procéder à deux types d'évaluations : l'évaluation théorique et l'évaluation en utilisant les benchmarks.

1. https://top500.org : Classe les 500 systèmes de calcul haute performance les plus puissants du monde

2.1.5.1 Utilisation des métriques théoriques

Toutes les métriques d'évaluation d'un HPC sont basées sur le temps d'exécution. Le temps d'exécution d'un programme est le temps écoulé entre le début et la fin de son exécution. Si nous considérons une machine séquentielle, le temps d'exécution est la somme du temps d'exécution de toutes les instructions du programme. Le calcul du temps d'exécution des programmes sur les machines parallèles est obtenu en additionnant le temps utilisé pour la division du calcul, sa distribution, le temps que la dernière unité de calcul utilise pour terminer le calcul, et le temps de combinaison des résultats. Ainsi, le coût $C_{p(n)}$ d'un programme parallèle de taille d'entrée n exécuté sur p processeurs est donné par $C_{p(n)} = T_{fin} - T_{début}$, où $T_{début}$ est le temps de début d'exécution et T_{fin} le temps de la fin. $C_{p(n)}$ mesure la quantité totale de travail effectuée par tous les processeurs. Pour une évaluation quantitative du temps d'exécution sur les machines parallèles, nous mesurons :

- La puissance du système : mesure le nombre d'opérations en virgule flottante que le système peut effectuer simultanément. Elle est donnée par formule $P = f \times nb_{op}$ où P désigne la puissance, f la fréquence et nb_{op} le nombre d'opérations que le système peut effectuer en même temps. L'unité de mesure de la puissance des systèmes de calcul parallèle est le *Floting Point Operation per Second* (FLOPS) [13]. Par exemple, une machine d'un teraFLOPS peut effectuer mille milliards d'opérations par seconde et une machine d'un petaFLOPS, un million de milliards d'opérations par seconde et un exaFLOPS peut atteindre un milliard de milliards d'opérations par seconde.
- Accélération ou *Speedup* : exprime le gain relatif en temps d'exécution qui peut être obtenu en utilisant une exécution parallèle sur P processeurs par rapport à la meilleure implémentation séquentielle. Elle est obtenue en prenant le rapport du temps d'exécution entre le programme séquentiel et sa version parallèle. Elle est donnée par la formule $S(n) = T(1)/T(n)$ où $S(n)$ est l'accélération obtenue avec n processeurs, $T(1)$ est le temps sur un seul processeur et $T(n)$ est le temps sur n processeurs.
- Efficacité : mesure la vitesse d'accélération par processeur supplémentaire. Elle est donnée par la formule $E(n) = S(n)/n$.

L'évolutivité est également utilisée pour évaluer les systèmes parallèles. Elle désigne la capacité d'un système parallèle à fournir une accélération proportionnelle au nombre de processeurs. Elle permet de mesurer la performance de l'algorithme parallèle et du système parallèle au fur et à mesure que de nouvelles unités de calcul sont ajoutées. L'évolutivité dépend du matériel (vitesse du bus par exemple), de la capacité de l'algorithme à être parallélisé et de la programmation.

2.1.5.2 Utilisation des benchmarks

L'évaluation théorique donne une idée de la performance d'un système parallèle. Une évaluation expérimentale utilisant les benchmarks est généralement utilisée. Ces benchmarks sont composés de programmes qui permettent une évaluation standardisée des performances. Quelques exemples de benchmarks célèbres dans le domaine du calcul haute performance :

- Linpack [36] : benchmark permettant de tester les performances des systèmes parallèles via la résolution d'un système d'équations linéaires à grande échelle sur des matrices denses. Les programmes proposés par Linpack nécessitent de calculs en double précision sur chacun des nœuds. Linpack est utilisé par le classement top500 pour savoir les machines les plus puissantes du monde ;
- *NAS parallel benchmarks* [37] : collection de onze petites applications parallèles développées par la *NASA Advanced Supercomputing* (NAS) dans le but d'évaluer les systèmes parallèles.

Pour mettre en place un système HPC, les différents nœuds de calcul sont assemblés pour former un supercalculateur ou *supercomputer*, une grappe de calcul ou *cluster computing*, une grille de calcul ou *grid computing* et un nuage de calcul ou *cloud computing*. Ces systèmes sont présentés dans les sections suivantes.

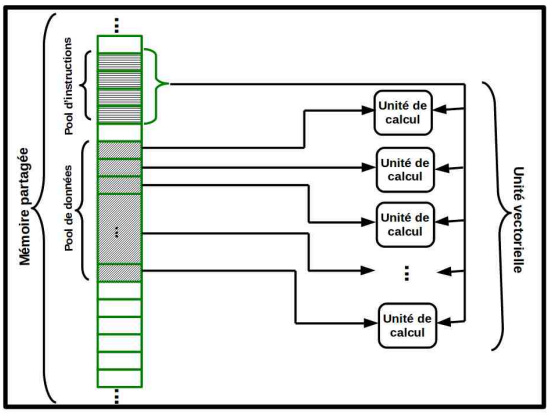

FIG. 2.9 – *Architecture des machines vectorielles*

2.2 Calcul haute performance sur super-ordinateur

Les premiers systèmes parallèles appelés superordinateurs ou encore supercalculateurs [12, 13, 38, 39, **?**, 6, 35, 4] sont les machines contenant des centaines, des milliers ou même des millions d'unités de calcul formant un système massivement parallélisé et organisé en un réseau de processeurs. Ils sont caractérisés par un grand nombre de processeurs puissants reliés par un réseau interne à haute performance (réseaux commutés ou maillés), un débit d'entrée/sortie extrêmement élevé et une très grande mémoire primaire et secondaire. Le grand nombre de processeurs leur permet d'effectuer de nombreuses tâches en parallèle et l'espace mémoire leur permet de stocker une grande quantité de données nécessaires pour résoudre des problèmes gourmands en calcul et/ou en mémoire. Les programmes sont généralement écrits en Fortran, C, C++, enrichis d'extensions de langages ou de bibliothèques pour le parallélisme comme MPI, openMP et PVM [35]. Dans la suite, nous présenterons les architectures des superordinateurs à la section 2.2.1 et quelques exemples à la section 2.2.2.

2.2.1 Architectures

Les architectures de superordinateurs diffèrent par la conception de leurs nœuds de calcul, de leurs commutateurs et de leur interface nœud-commutateur. Il existe deux architectures principales : les architectures de machines vectorielles et les architectures de machines superscalaires [3] :

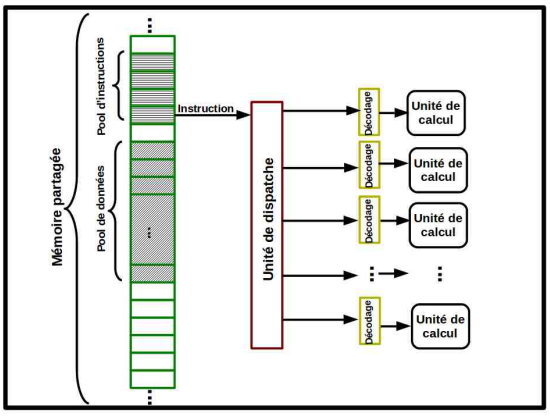

FIG. 2.10 – *Architecture des machines superscalaires*

– Architectures de machines vectorielles (figure 2.9) : les superordinateurs basés sur les architectures vectorielles utilisent les processeurs vectoriels (associés pour fonctionner en parallèle) pour appliquer un bloc d'instructions à un ensemble de données. Les architectures vectorielles favorisent ainsi les calculs s'appliquant aux tableaux de nombres comme les images.
– Architectures des machines superscalaires (figure 2.10) : les architectures superscalaires sont utilisées pour construire des superordinateurs basés sur les processeurs superscalaires. Ces processeurs exécutent plusieurs instructions simultanément parmi une suite d'instructions. Pour ce faire, il est capable d'identifier l'absence de dépendances entre les instructions.

2.2.2 Exemples

Pendant plusieurs décennies, le superordinateur le plus rapide du monde, basé sur les processeurs vectoriels, a été fabriqué par Seymour Cray et a été nommé Cary-1. Il offrait une puissance de calcul de 160 MFLOPS. Par la suite, il a été remplacé par Cray X-MP, offrant une puissance de 800 MFLOPS, Cray-2 atteignant 1,9 GFLOPS. Une fois sur le marché, plus de 100 Cray-1 (avec un coût de $7,9 million (4,28 milliard XAF) en 1977) ont été vendus, à des prix allant de $5M (2,71 milliard XAF) à $8M (4,34 milliard XAF) [3, 6]. Par la suite, les machines ILLIAC implémentant les architectures superscalaires (massivement parallèles) ont été proposées sur le marché. ILLIAC IV avait 256 processeurs et offrait une vitesse allant jusqu'à 1 GFLOPS. D'autres superordinateurs ont été fabriqués par les constructeurs comme Nec, Fujitsu, nCube, Convex, Alliant, etc. [6]. La machine LINKS-1 de l'Université d'Osaka au Japon a utilisé une architecture massivement parallèle avec 514 microprocesseurs; le supercalculateur Numerical Wind Tunel de Fujitsu utilisait 166 processeurs vectoriels pour une performance de 1,7 GFlops par processeur ; Hitachi SR2201 a atteint une performance de 600 GFlops en utilisant 2048 processeurs ; NEC SX décrit une série de superordinateurs vectoriels conçus, fabriqués et commercialisés par NEC. SX-6 a permis d'atteindre une puissance de 35,86 TFLOPS ; une machine nCUBE-3 peut utiliser jusqu'à 65536 processeurs, pour 6,5TFLOPS ; le supercalculateur Japonais Earth Simulator basé sur un ensemble de processeurs vectoriels a trôné au sommet du top500 de 2002 à 2004.

Aujourd'hui, les superordinateurs représentent moins de 17% des machines les plus puissantes au monde répertoriées par le Top500. En effet, les facteurs comme le coût d'achat, l'évolutivité difficile (pour avoir une machine plus puissante, il faut l'acheter) [40, 41], la disponibilité d'ordinateurs personnels de plus en plus puissants et moins chers et l'amélioration des technologies de réseaux d'interconnexion ont poussé les utilisateurs vers des solutions basées sur la mutualisation des ressources de calcul.

2.3 Calcul haute performance sur grappe

Une grappe de calcul ou *cluster* est un système informatique composé d'un ensemble d'ordinateurs autonomes (comprenant un ou plusieurs processeurs ou cœurs, la mémoire, les unités d'entrée/sorties) interconnectés par un réseau local haute performance et capable de fonctionner comme un pool unique de ressources informatique intégrées [15, 16, 40, 42, 43]. Le logiciel installé sur chaque nœud de la grappe est identique et l'accès aux ressources externes (nœud principal ou serveur de fichier) est uniforme, donnant aux usagers l'illusion qu'ils utilisent un ordinateur très puissant alors que le système peut être composé d'une centaine voir des milliers d'ordinateurs [44]. Les grappes sont généralement déployées pour améliorer les performances et la disponibilité par rapport à un seul ordinateur, tout en étant beaucoup plus rentables que les ordinateurs simples de vitesse comparables [45, 46]. Par rapport au supercalculateur, elles ont des performances évolutives, un meilleur système de tolérance aux pannes et une croissance modulaire. Dans la suite de la section, nous présentons l'architecture d'une grappe (section 2.3.1) et quelques exemples de grappes (section 2.6.2).

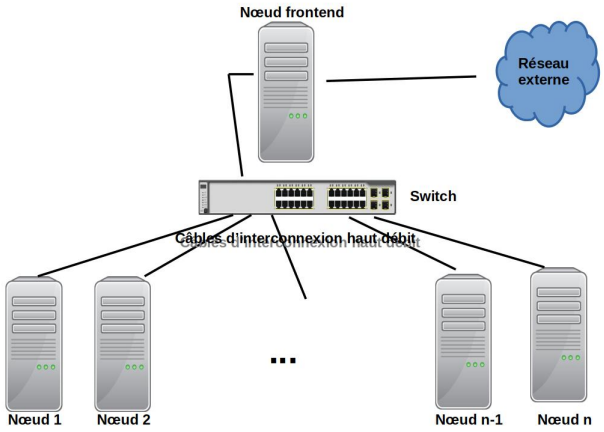

FIG. 2.11 – *Architecture d'une grappe de calcul*

2.3.1 Architecture

L'architecture d'une grappe de calcul (figure 2.11) est composée d'un ensemble de nœuds connectés via du matériel réseau. Comme les machines massivement parallèles historiques, chaque nœud utilise des composants standards qui sont la mémoire, le disque dur, les unités d'entrées/sorties et le processeur. Les interconnexions sont basées sur un réseau haut débit [3, 15].

L'environnement matériel de la grappe est composé de :

- Nœuds : chaque nœud de la grappe peut-être un ordinateur ordinaire ou un serveur puissant. Les principaux types de nœuds sont :
  - Nœud utilisateur : permet l'accès externe à la grappe. Les utilisateurs vont s'y connecter pour compiler et exécuter leurs calculs ;
  - Nœud de commande : permet de fournir les services réseau de base et de faire la planification des calculs sur les nœuds. Le logiciel de planification des tâches utilisé y est installé ;
  - Nœud de gestion : ce nœud est responsable de la gestion des nœuds de contrôle et du contrôle des opérations réseau ;
  - Nœuds de stockage : les nœuds de stockage agissent comme la mémoire et le serveur de données de la grappe ;
  - Nœud d'installation du système : responsable de l'installation du système d'exploitation et de divers logiciels tels que les bibliothèques de calcul ;
  - Nœud de calcul : a pour but d'effectuer les calculs. En fonction des besoins spécifiques et du budget, l'on va décider des configurations et du nombre de nœud de calcul à avoir dans une grappe.
- Réseau d'interconnexion : le réseau d'interconnexion permet à tous les composants de la grappe d'être connectés pour former un système complet. Il offre un moyen rapide et fiable de communication de données entre les nœuds et potentiellement en dehors de la grappe. Le matériel d'interface réseau est responsable de la transmission et de la réception des paquets de données. Pour minimiser au maximum les délais de transmission des données, les nœuds sont placés physiquement dans la même pièce et connectés en utilisant un réseau à faible latence et large bande passante.

En fonction du contexte, le nœud utilisateur, le nœud de contrôle, le nœud de gestion, le nœud de stockage et le nœud d'installation peuvent se trouver sur le même ordinateur appelé le nœud maître de la grappe, et les autres nœuds s'occupent du calcul. Un système de remplacement du nœud maître permet de gérer les pannes ou les congestions. L'ajout et la suppression des composants est faite de

façon transparente à l'utilisateur. L'équilibrage de charge permet la mise à l'échelle des performances en répartissant la charge entre plusieurs nœuds de calcul ou grappes. Pour offrir une haute disponibilité, les applications et les données sont rendus disponibles sur plusieurs grappes reliées entre elles [47].

L'environnement logiciel des grappes est composé de :

- Logiciels système : permettent l'installation, l'initialisation, l'administration du système ; la planification et l'allocation des ressources ;
- Les outils de programmation parallèle : sont composés des bibliothèques, des compilateurs et des débogueurs de performance et d'exactitude. Les bibliothèques de passage de messages sont généralement utilisées (MPI et PVM) pour les échanges entre les nœuds de calcul [3, 34].

2.3.2 Exemples

Étant donné que les grappes sont plus faciles à déployer et à passer à l'échelle que les supercalculateurs, elles se sont rapidement imposées dans le monde du calcul parallèle. D'après le top500 de novembre 2020, elles constituent 93% du parc des machines les plus puissantes du monde. La machine la plus puissante du monde, nommée *Fugaku*[2], déployée au *RIKEN Center for Computational Science* à Kobe au Japon est une grappe de calcul composée de 7,6 million de cœrs qui fournit une puissance de calcul de l'ordre de 500 000 TFLOPS et consomme environ 30 kW d'énergie. Elle remplace la grappe *K computer* qui a occupée la tête du top500 des machines les plus puissantes du monde pendant deux éditions consécutives (juin et novembre 2011). Plusieurs autres constructeurs tels qu'IBM, NEC et HP ont également développé des grappes de calcul.

2.4 Calcul haute performance sur grille

La popularité d'Internet, la disponibilité d'ordinateurs puissants et des technologies réseau haut débit à faible coût ont conduit à la possibilité d'utiliser les ordinateurs répartis géographiquement sur plusieurs centaines de kilomètres comme plateforme de calcul à grande échelle. Cette plateforme s'appelle une grille de calcul. Par définition, une grille de calcul [17, 18, 48] est une infrastructure virtuelle composée de plusieurs ressources potentiellement partagées, distribuées, hétérogènes, délocalisées et autonomes, connectées à un réseau comme Internet et capable de fonctionner comme un seul pool de ressources informatiques intégrées. Elles sont généralement gérées par une organisation spécifique et les ressources de calcul sont fournies par diverses institutions de soutien (entreprises, les groupes de recherche, les laboratoires et les universités) ou encore par des particuliers. Il existe deux types de grilles : celles qui permettant l'exploitation de machines en jachère provenant des particuliers et celles qui exploitent des ressources dédiées provenant des domaines administratifs et des entreprises [49]. Dans cette section, nous traiterons les grilles dont les ressources proviennent des organisations. Les grilles dont les ressources proviennent des machines inactives des particuliers sur Internet encore appelées systèmes de calcul sur machines volontaires seront traitées au chapitre 3. Dans les sections 2.4.1 et 2.4.2, nous présentons respectivement l'architecture des grilles et les exemples de grilles de calcul.

2.4.1 Architecture

Une grille a une architecture en couches [18], extensible et ouverte (voir figure 2.12) facilitant l'identification des classes générales de composants. Les composants de chaque couche partagent des caractéristiques communes suivantes :

- **La couche Fabric ou encore couche matrice :** fournit les interfaces pour le contrôle local des ressources pouvant être des ordinateurs personnels, des grappes ou des pools d'ordinateurs

2. `https://top500.org/system/179807/`

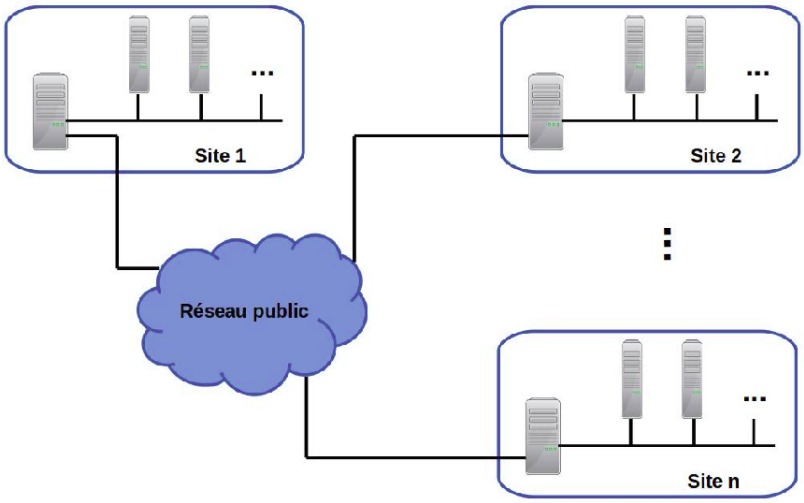

FIG. 2.12 – *Architecture d'une grille de calcul*

distribués. Les principaux éléments que l'on trouve sur cette couche sont :

1. **Ressources de calcul** : sont les ressources utilisées pour effectuer les calculs haute performance. La puissance d'une grille dépend du nombre de ressources de calcul utilisées. En général, de nouvelles ressources peuvent être ajoutées à tout moment, certaines peuvent être retirées temporairement ou définitivement.
2. **Ressources de stockage** : les ressources de stockage sont utilisées pour la gestion des données (acquisition, enregistrement et restauration);
3. **Les ressources réseau** : ces ressources permettent le contrôle des transferts de données dans le réseau ;
4. **Référentiels de code** : les référentiels de code permettent la gestion du code (par exemple, la gestion des versions) ;
5. **Catalogues** : permettent la gestion (enregistrement, requêtes et mise à jour) du catalogue des ressources.

– **Couche connectivité** : utilisée pour la définition des principaux protocoles de communication et d'authentification requis pour les transactions réseau. Au niveau de cette couche, l'identité des utilisateurs et des ressources est vérifiée par des protocoles d'authentification ;
– **Couche ressource** : au niveau de la couche ressource, les protocoles pour la négociation sécurisée, l'initialisation, la surveillance, le contrôle, la comptabilité et le paiement des opérations de partage sur les ressources individuelles sont définis. Deux principales classes de protocoles sont à considérer : les protocoles d'information et les protocoles de gestion. Les protocoles d'information sont utilisés pour obtenir des informations sur la structure et l'état d'une ressource. Les protocoles de gestion sont utilisés pour négocier l'accès à une ressource partagée ;
– **Couche collective** : contient les protocoles et les services qui ne sont associés à aucune ressource spécifique mais sont de nature globale et capturent les interactions entre les collections de ressources. Une grande variété de services est implémentée : les services d'annuaire, les services de surveillance et de diagnostic, les services de réplication de données, etc.
– **Couche d'application** : cette couche est composée des applications des utilisateurs à exécuter sur le système.

Le modèle de programmation le plus utilisé dans les grilles de calcul est le modèle par passage de messages [6]. Les API tels que MPI et PVM sont généralement utilisées pour programmer les plateformes de grilles. Notons, cependant, que de nombreuses grilles offrent aux utilisateurs un outil

qui effectue la découverte des ressources, la planification des travaux et la surveillance des processus sur les ressources de la grille [50]. Plusieurs outils sont proposés : Globus Toolkit TM, Legion, Unicore, gLite. La plateforme gLite [18] par exemple est basée sur une architecture orientée service qui facilite les interactions entre les applications encapsulées dans les services de grille. Les utilisateurs peuvent déployer différents services en fonction de leurs besoins.

2.4.2 Exemples

En plus d'être faciles à déployer et à passer à l'échelle, le coût de déploiement et d'accès aux grilles de calcul est bien inférieur à celui des grappes. C'est la raison pour laquelle les grilles de calcul sont très populaires. Plusieurs exemples de grilles ont été noté dans la littérature : Grid'5000, PlanetLab, EUROGRID, Globus, BioGrid, Condor, DataTAG, GRACE, Legion, DataGRID, EuroGrid. Le French Grid'5000[3] par exemple est réparti sur 8 sites, est composée de 31 grappes et 12 328 cœurs. Chaque site héberge une grappe de calcul d'un laboratoire universitaire en France. Les sites sont connectés par un réseau haut débit. PlanetLab est une grille de calcul composé de plus de 507 sites dans le monde. Pour utiliser cette grille, il faut avoir un compte dans une entreprise ou dans une université et héberger des nœuds de calcul. EUROGRID est un réseau européen de grille de calcul regroupant les principaux centres de calcul haute performance de différents pays européens [28].

Le faible coût de déploiement et d'accès aux grilles de calcul en fait la solution la plus utilisée dans les universités pour résoudre des problèmes scientifiques, mathématiques et académiques intensifs dans les pays développés. Cependant, la complexité administrative pour l'utiliser est un frein pour le grand public [28].

2.5 Calcul haute performance sur cloud

Tout comme les supercalculateurs, les grappes et les grilles de calcul, le but du calcul sur le cloud ou *cloud computing* en anglais est de permettre aux usagers d'avoir accès à une grande puissance de calcul en agrégeant les ressources et fournissant une vue système unique [51]. Il est composé d'un ensemble de ressources de calcul interconnectées et virtualisées. Le paiement du calcul est se fait à l'utilisation et la capacité élastique du système (ajout de nouvelles ressources en fonction des besoins en calcul) donne l'illusion de ressources infinies à l'utilisateur [52, 53]. On parle alors de *HPC-as-a-service* (HPCssS), exactement comme les autres formes de services disponibles dans le cloud tels que *Software-as-a-Service* (SaaS), *Platform-as-a-Service* (PaaS), *Infrastructure-as-a-Service* (IaaS) [54, 55]. Dans la suite de cette section, nous présentons l'architecture des HPC cloud (section 2.5.1), et quelques exemples de HPC cloud (section 2.5.2).

2.5.1 Architecture

Un système de calcul haute performance basé sur le cloud (voir architecture 2.13) est composé d'une plate-forme frontale, des plate-formes back-end et du réseau de communication [53].

- Plateforme frontale encore appelé client cloud : elle est composée des clients qui soumettent le calcul. Ces clients sont des ordinateurs personnels, des serveurs ou encore des smartphones. Ils utilisent les services Web pour allouer et libérer les ressources de calcul requises en fonction de la charge de travail. Ces services offrent une interface (le plus souvent Web) pour créer et exécuter les applications ou des services nécessitant la création d'une infrastructure grappe de calcul, la soumission de travaux, les services d'allocation appropriée des ressources de calcul.
- Plateformes backend : les plateformes backend sont composées d'un ensemble de serveurs de calcul le plus souvent virtualisés. Pour accéder à ces ressources de calcul, les clients spécifient

3. `https://www.grid5000.fr/w/Grid5000:Home`

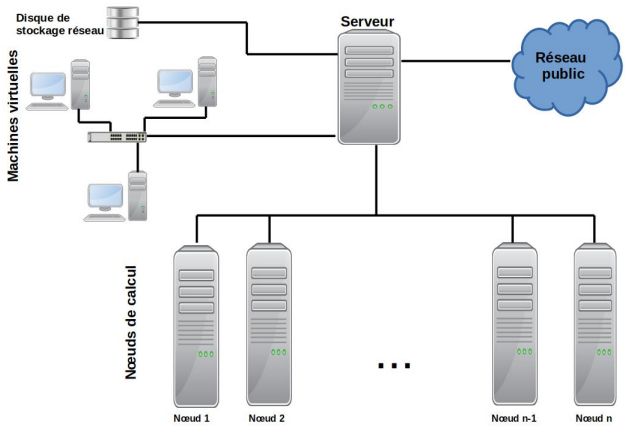

FIG. 2.13 – *Architecture d'un nuage de calcul*

leurs exigences et configurent l'ensemble des machines nécessaires pour résoudre leur problème. Une base de données permet de stocker les données d'applications de l'utilisateur. Ces données sont utilisées pour effectuer certains audits dans le but d'optimiser l'utilisation des ressources de calcul acquises.

– Réseau d'interconnexion : il existe deux types de réseaux de communication. Les ressources de calcul sont connectées à l'aide d'un réseau à haut débit et à faible latence ; il est recommandé que les nœuds de calcul alloués aux utilisateurs soient le plus proche possible. Les utilisateurs accèdent aux ressources en utilisant le réseau Internet qui n'est pas un réseau à haut débit.

Les utilisateurs interagissent avec les clouds de calcul à l'aide d'une API telle que MPI ou d'un outil mise en place par le fournisseur de cloud. Lors de l'utilisation, l'élasticité donne la possibilité de faire évoluer la capacité des nœuds en augmentant les instances des machines avec des tailles de mémoires et de cœurs.

2.5.2 Exemples

Plusieurs solutions cloud ont été identifiées : Amazon EC2, Penguin Computing, Google Colab, Microsoft Azure, etc. Elastic Compute Cloud (EC2) d'Amazon est la plate-forme HPC cloud la plus populaire [56]. Amazon vend plusieurs types d'instances de machines qui comprennent des nœuds de grappes de calcul, qui se distinguent par différentes capacités de calcul et de réseau. L'instance la plus haut de gammes, appelée *Cluster Compute Eight Extra Large* est l'instance destinée aux applications HPC [56]. Penguin Computing fournit le cloud Penguin On Demand (POD). C'est un modèle de calcul permettant d'exécuter le code dans lequel chaque utilisateur reçoit un nœud de connexion virtualisé.

Le calcul sur le cloud donne de nouvelles pistes d'utilisation des HPC, ouvrant de nouvelles possibilités de développement aux universitaires et aux entreprises de toutes les tailles. La facturation à l'utilisation permet de répartir l'investissement consacré à l'utilisation de plusieurs centaines ou milliers de processeurs entre plusieurs clients. Il permet aux entreprises de créer des grappes temporaires à volonté et de cesser de les payer dès que les besoins sont satisfaits [56]. Ainsi, pour les utilisateurs qui connaissent très bien leurs besoins en calcul, le HPC cloud est la solution adéquate. Cependant, les scientifiques ne prédisent généralement pas toujours la durée que prendra une expérimentation. Ceci peut rendre le travail difficile et entraîner des pertes car une fois les ressources choisies avec un budget, si le budget est épuisé, même s'il reste encore des expérimentations à faire, tout est arrêté. Ceci est illustré par Marathe et al. [56] : ils ont entrepris une série d'expériences sur l'EC2 d'Amazon et ont dû stopper car chaque expérience pour mesurer le temps de démarrage sur 64 nœuds leur coûtait $15 et

ils ne disposaient pas d'un budget conséquent.

2.6 Systèmes de calcul haute performance en Afrique

Les systèmes de calcul haute performance sont rares en Afrique. Nous avons identifié deux grappes de calcul : Inspur présentée à la section 2.6.1 et Lengau présentée à la section 2.6.2.

2.6.1 La grappe Inspur de ZCHPC

Zimbabwe Center for High Performance Computing (ZCHPC) a acquis une grappe de calcul en 2015 [57]. Elle possède 100 nœuds de calcul et quatre nœuds accélérateurs (le tout englobant 1400 cœurs) et près de 2 To de mémoire. Les nœuds de calcul sont reliés entre eux par un réseau QDR InfiniBand et le système de stockage possède un réseau Fibre Channel 8 Gbit/s. Elle délivre une puissance de calcul de près de 300 TFLOPS et est considérée comme la deuxième machine la plus puissante du continent. Bien qu'installée au ZCHPC, elle agit comme un HPC national desservant plus de 400 utilisateurs dans les universités et autres institutions du Zimbabwe. Elle est utilisée pour l'agriculture, les mines, l'économie et les finances, l'industrie de fabrication et la protection civile. L'un des bénéfices immédiats est la capacité à fournir en moins de cinq minutes les prévisions météorologiques de trois jours au Zimbabwe.

2.6.2 Lengau

Lengau [58] est une grappe de calcul Dell EMC de 1,4 PFLOPS installée au *South Africa's Center of High Performance Computing* (CHPC) en Afrique du Sud. Elle est composée de 1039 serveurs Dell PowerEdge (avec 32 856 cœurs) basés sur les processeurs Intel Xeon E5. Elle a une capacité de stockage de 4 Po. Elle est considérée comme la machine la plus puissante du continent. Sur la liste du top500 des machines les plus puissantes au monde, elle a été classée 121e en 2016 et 496e en 2019. Près de 1500 scientifiques et industriels se sont enregistrés pour l'utiliser, dont 500 sont activement engagés dans plus de 200 programmes de recherche. Certains des programmes de recherche ayant utilisé Lengau sont : la modélisation moléculaire, les prévisions météorologiques, la modélisation climatique, la science des matériaux, la recherche astronomique, etc. L'observation de l'utilisation de Lengau du 14 au 20 juin 2019 montre qu'elle est utilisée à 90% en moyenne, très proche pour atteindre sa pleine capacité.

2.7 Conclusion

Ce chapitre était consacré aux systèmes de calcul haute performance sur machines dédiées. Il a tout d'abord introduit le calcul parallèle, qui est la base des systèmes de calcul haute performance. Ensuite, il a présenté les HPC basés sur les superordinateurs, les HPC basés sur la mutualisation de ressources de calcul telles que les grappes, les grilles et le cloud de calcul et leurs limites. Enfin, il a décrit deux systèmes HPC en Afrique. Un résumé de l'évaluation des systèmes HPC de ce chapitre est présenté par le tableau 2.1. Au vu de ce tableau, les solutions présentées dans ce chapitre sont difficiles à mettre en œuvre, difficile d'accès (coût) et donc inadaptées aux environnements à ressources limitées. Dans le chapitre 3, une alternative sera abordée.

Machines	**Coût de mise en place**	**Efficacité**	**Evolutivité**	**Mise en place**
Superordinateurs	Très élevé	Efficace	Mauvaise	Difficile
Grappes de calcul	Élevé	Efficace	Bonne	Facile
Grilles de calcul	Faible	Efficace	Bonne	Facile
Cloud de calcul	Élevé	Efficace	Bonne	Facile

TAB. 2.1 – *Tableau de comparaison des différentes solutions HPC présentées dans ce chapitre*

Chapitre Trois

Calcul sur machines volontaires

Dans le chapitre 2, nous avons présenté les supercalculateurs, les grappes de calcul, les grilles de calcul et le calcul sur le cloud comme solutions aux problèmes de calcul haute performance. Bien qu'efficaces, ces solutions nécessitent d'énormes investissements initiaux en matériel et en logiciels, entraînant parfois des coûts supplémentaires en termes de configuration, de coordination et d'administration [23, 59]. Dans ce chapitre, nous présentons une alternative basée sur l'utilisation des machines volontaires. En effet, la démocratisation des ordinateurs personnels et le coût d'achat de plus en plus bas les rendent accessibles à tous. Cependant, ces ordinateurs sont généralement sous-utilisés. L'idée est donc d'agréger leur temps d'inactivité pour former un système de calcul haute performance [23]. Dans la suite de ce chapitre, nous commençons par les généralités sur les systèmes de calcul sur machines volontaires dans la section 3.1, les architectures, la gestion des ressources et des tâches dans les sections 3.2 et 3.3 respectivement. À la fin du chapitre, nous mentionnons quelques exemples de systèmes dans la section 3.4 et le bilan à la section 3.5.

3.1 Généralités

Un système de calcul sur machines volontaires ou encore *Volunteer Computing System* (VCS) est un système composé d'un ensemble de ressources de calcul acquises auprès de volontaires. Les volontaires sont les personnes qui possèdent des ressources et qui acceptent d'offrir du temps CPU et de la mémoire pendant leurs périodes d'inactivité pour le calcul. Pour former une plateforme de calcul globale, les VCS vont acquérir et partager du temps de calcul obtenu à partir des ressources disponibles dispersées dans le monde entier [25, 24]. Les principaux éléments à prendre en compte lors de la mise en place d'un VCS sont les logiciels, les algorithmes d'ordonnancement, la sécurité, la résilience aux pannes, la gestion des données et le modèle de programmation utilisé [60]. Plusieurs critères peuvent être utilisés pour positionner les VCS par rapport aux autres approches [28, 29, 60, 61, 62, 63]:

- Volontariat : le système de volontariat est la pièce maîtresse des VCS. En effet, les propriétaires de ressources vont choisir de faire don de la puissance de calcul inutilisée de leurs machines à un projet scientifique d'intérêt public (par exemple lutte contre le Corona [1]). Afin de disposer d'une puissance significative, le système doit pouvoir inciter le plus grand nombre de volontaires à s'inscrire et d'encourager à maintenir l'engagement des volontaires déjà inscrits. Pour faciliter l'adhésion des volontaires, ceux-ci doivent être informés de le finalité des ressources et de la manière dont elles sont utilisées. Pour faciliter l'accès des volontaires au système, les administrateurs n'ont pas besoin d'installer de logiciel sur les machines volontaires. Un volontaire se contentera de télécharger et d'installer un logiciel qui récupérera les calculs et les exécutera.
- Ressources : l'environnement des VCS est dynamique, le nombre de ressources peut augmenter ou diminuer, les nœuds de calcul peuvent rejoindre ou quitter le système sans préavis, les machines sont hétérogènes et géographiquement dispersées. Les ressources de calcul (ordinateurs

1. https://foldingathome.org/diseases/infectious-diseases/covid-19/

personnels, tablettes, téléphones, etc.) sont diverses en termes de matériel (vitesse CPU, RAM, disque dur, etc.), de logiciel (systèmes d'exploitation, etc.) et de réseau (bande passante, proxy, pare-feu, etc.). Le défi des VCS est la gestion des ressources volatiles et faiblement connectées (bande passante et latence de la communication limitées).

- Performance : les VCS ont la capacité de concurrencer les plus grands systèmes de calcul haute performance du monde. La plus grande plateforme de calcul sur machines volontaires appelée BOINC regroupe plus de 4 millions de volontaires et fournit une puissance de calcul de 33 278 TeraFLOPS en janvier 2021 [2].
- Rentabilité : les VCS sont beaucoup moins chers que les autres systèmes car de nombreux projets de recherche nécessitant de gros calculs peuvent être réalisés sans avoir à dépenser de l'argent pour acheter ou louer une infrastructure de calcul.
- Adoption : des centaines de groupes de recherche utilisent actuellement les systèmes de calcul sur machines volontaires et de nombreuses plate-formes de VCS (par exemple BOINC) ont des millions d'utilisateurs, fournissant une énorme quantité de mémoire et de traitement.
- Évolutif : les VCS sont faciles à faire évoluer car pour faire partie du système, il suffit de télécharger et d'installer les applications à utiliser pour les calculs.
- Mise en place : les VCS permettent de fournir une infrastructure de calcul à grande échelle, sans qu'il soit nécessaire d'investir dans du matériel, de l'espace physique, etc. Les coûts d'investissement et d'exploitation sont supportés par les utilisateurs ou les organismes donateurs. La difficulté dans la mise en place réside dans le développement de logiciels à être utilisés tant au niveau des clients que des volontaires.
- Sécurité : les VCS doivent garantir la sécurité et la non-détérioration de la qualité de service lors de l'utilisation par les volontaires et la validité des résultats. En effet, un pirate peut prendre le contrôle du serveur et distribuer des logiciels malveillants comme calculs aux volontaires. Les ordinateurs des volontaires étant anonymes et non fiables peuvent renvoyer des logiciels malveillants à la suite d'un calcul. Un système permettant de sécuriser le serveur, de vérifier et de valider les résultats des calculs doit être mis en place.

3.2 Architectures

L'architecture générale d'un VCS [22, 62, 64] est composée de trois principaux éléments : clients, nœuds de calcul volontaire et serveur(s). En fonction de l'organisation des composants, on distingue l'architecture centralisée (figure 3.1), l'architecture distribuée (figure 3.2) et l'architecture hybride. Dans la suite de cette section, nous présentons ces architectures d'un point de vue matériel (section 3.2.1) et logiciel (section 3.2.2).

3.2.1 Environnement matériel

Un environnement de VCS se compose principalement de :

- Serveur(s) : le serveur est le gestionnaire du système.
  - Dans l'architecture centralisée, les volontaires et les clients s'enregistrent auprès du serveur. Le serveur reçoit les calculs des clients, planifie et distribue les tâches aux volontaires. Une fois que les résultats de l'exécution des tâches sont retournés, il vérifie leur exactitude, les agrège et les renvoie au client.
  - Dans l'architecture distribuée, il n'y a pas de serveur. Un groupe de volontaires est désigné pour coordonner et gérer le système.

2. https://www.boincstats.com/stats/-1/project/detail/

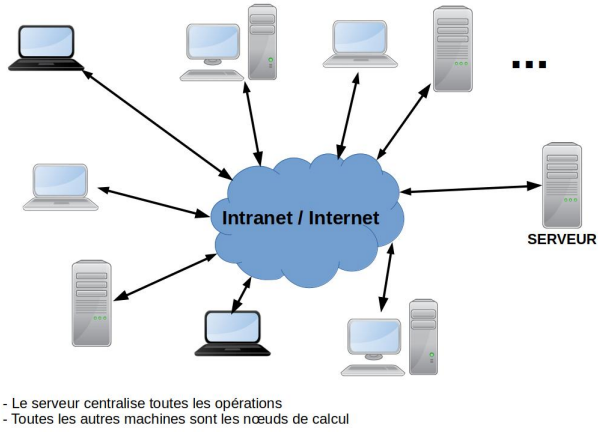

FIG. 3.1 – *Architecture centralisée d'un système de calcul sur machines volontaires*

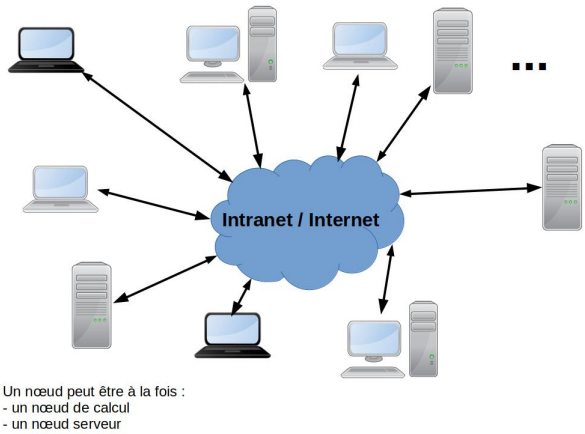

FIG. 3.2 – *Architecture distribuée d'un système de calcul sur machines volontaires*

- **Volontaires** : jouent plusieurs rôles selon l'architecture implémentée.
  - Dans l'architecture centralisée, les volontaires s'inscrivent, reçoivent les tâches du serveur, les exécutent et renvoient les résultats au serveur.
  - Dans une architecture distribuée, un certain nombre de volontaires appelés volontaires coordinateurs est responsable du maintien des informations entre les volontaires, de la coordination, de la planification et de la mise en commun des ressources. Les volontaires s'enregistrent en échangent les informations entre eux et avec les nœuds de coordination. Lorsque une tâche est soumise par un client, les volontaires coordinateurs la distribuent à d'autres volontaires. Les volontaires qui reçoivent les tâches les exécuteront et renverrons les résultats aux volontaires coordinateurs. Ces résultats seront rassemblés, vérifiés, combinés et renvoyés au client. La planification est effectuée de manière distribuée en fonction de la disponibilité des volontaires du système.
- **Client** : le client est la machine qui soumet le calcul. Dans l'architecture centralisée, le client soumet le calcul au serveur et récupère le résultat à la fin. Dans l'architecture distribuée, le volontaire qui joue le rôle du client soumet sa tâche à ses voisins volontaires. Les volontaires coordinateurs collectent les résultats des autres et renvoient le résultat au client à la fin.

Dans l'architecture centralisée, le fait que tous les volontaires ne fassent confiance qu'à un serveur

central est une garantie de la sécurité du système [24]. Dans le même temps, la gestion centralisée peut faire du serveur un point de défaillance unique et créer un goulot d'étranglement si le système devient trop grand. Dans l'architecture décentralisée, le fait qu'un nœud puisse être à la fois un nœud de calcul et un nœud serveur présente l'avantage de la flexibilité et l'évolutivité du système. Cependant, cela peut poser un problème de confiance entre les volontaires [24].

L'architecture hybride combine la flexibilité et l'évolutivité d'une architecture décentralisée avec la sécurité et la confiance d'une architecture centralisée. Avec cette architecture, les nœuds de calcul volontaires peuvent échanger certaines informations entre eux et gérer certaines fonctionnalités sans l'intervention d'un serveur central. Le serveur central peut être utilisé comme répertoire qui contient la description de toutes les ressources [24].

3.2.2 Environnement logiciel

L'environnement logiciel d'un VCS est composé d'un ou de plusieurs modules logiciels permettant de collecter les données sur les volontaires, de soumettre du calcul, de regrouper un ensemble de ressources étant donné un calcul, d'effectuer les calculs, d'agréger les résultats et de renvoyer les résultats finaux au client. Quelle que soit l'architecture implémentée, les modules logiciels doivent inclure la gestion des clients, la gestion du serveur et la gestion des volontaires [24, 65].

- Gestion du serveur. Le logiciel ou le module logiciel utilisé pour gérer le serveur comprend généralement les fonctionnalités suivantes :
  - Collecte des données sur les volontaires : elle consiste à enregistrer les données sur les caractéristiques et le fonctionnement des différents volontaires.
  - Gestion des calculs : elle peut être décomposée en plusieurs sous-fonctionnalités : la réception du calcul d'un client consiste à recevoir un calcul soumis par un client ; la génération de tâches consiste à diviser un calcul en tâches ; l'attribution des tâches consiste à distribuer les tâches aux volontaires sélectionnés pour le calcul ; la collecte et l'agrégation des résultats consiste à récupérer les résultats auprès des volontaires et à les combiner pour obtenir un résultat final ; le retour des résultats au client.
  - Ordonnancement : l'ordonnancement est la sélection des volontaires appropriés à un calcul soumis. Il prend en compte les tâches obtenues après la phase de division du calcul et les informations sur le profil des volontaires enregistrés dans le système. En effet, lorsque le serveur reçoit un calcul et le divise en tâches, l'étape suivante consiste à construire les groupes de machines volontaires chargés d'exécuter ces tâches en fonction de leurs capacités, leur disponibilité, leur réputation et leur fiabilité. Il planifie ensuite la répartition des tâches et les distribue aux différents volontaires sélectionnés en utilisant un algorithme d'ordonnancement.
  - Certification des résultats : une fois que le résultat des calculs sont collectés par le serveur, celui-ci vérifie leur exactitude afin d'identifier les résultats erronés et de relancer les calculs si nécessaire.
  - Gestion de données : un système de stockage des données importantes doit être mis en place pour une meilleure gestion du système. Exemples de données importantes : les données permettant la gestion des ressources logicielles et matérielles disponibles, la connaissance à temps réel de la capacité du système, le profil des volontaires, des tâches déjà exécutés, etc.
  - Sécurité : un système de sécurité doit sécuriser l'environnement du volontaire afin qu'un client ou volontaire ne lui fasse pas exécuter un code malveillant. Il doit permettre également la sécurité des calculs et celle du serveur.
- Gestion du volontaire : la gestion du volontaire est clé du système. L'application installée au niveau du volontaire doit leur permettre de s'inscrire, de récupérer les tâches, de les exécuter

et de renvoyer les résultats. Les développeurs doivent s'assurer que l'application fonctionne de manière transparente sur la machine du volontaire sans dégrader les performances de celle-ci :
  - Enregistrement des volontaires : les volontaires enregistrent des informations telles que le type de ressource de calcul, le système d'exploitation, la puissance du CPU et la taille de la mémoire. Notons que l'idéal est de collecter ces informations automatiquement une fois que le volontaire a installé l'application.
  - Gestion des tâches : les volontaires sélectionnés pour un calcul exécutent la tâche qui leur est allouée et retournent les résultats.
- Gestion du client. Les deux principales fonctionnalités au niveau du client sont : l'envoi des données au serveur, la soumission des calculs et la récupération des résultats.
  - Soumission du calcul : cette fonctionnalité permet au client d'envoyer un calcul à exécuter. Il s'agit ici d'un programme (ou ensemble de programmes) et éventuellement d'un ensemble de données sur lequel le programme va s'appliquer.
  - Collecte du résultat : cette fonctionnalité permet au client de récupérer le résultat du calcul.

3.3 Gestion des ressources de calcul et des tâches

Une bonne gestion des ressources de calcul, une planification et une distribution efficace des tâches sont essentielles dans les systèmes de calcul sur machines volontaires. En effet, le système doit pouvoir répondre aux exigences des clients et des volontaires : les clients qui ont soumis le calcul souhaitent que leur travail leur soit rendu le plus rapidement possible et qu'il soit fiable ; et les volontaires qui ont gracieusement fait don de leurs ressources ont besoin de reconnaissance, de sécurité, d'une utilisation efficace et non embêtante de ces ressources [66]. Cependant, les ressources sont hétérogènes et de disponibilité intermittentes. Indépendamment de l'architecture implémentée par le système, la gestion des ressources et des tâches permet l'acquisition et la conservation des ressources, l'allocation des ressources pour le calcul, l'exécution des tâches, le retour des résultats et la sécurisation des ressources. Dans la suite, la section 3.3.1 présente le recrutement, la motivation et la valorisation des volontaires. La section 3.3.2 présente comment sont enregistrés et gérés les machines, la section 3.3.3 présente comment les ressources sont organisées pour résoudre un problème de calcul, la section 3.3.4 présente comment le calcul est effectué et les résultats sont validés par le système, et enfin, la section 3.3.5 présente comment les ressources sont sécurisées.

3.3.1 Recrutement, motivation et valorisation des volontaires

Lors du recrutement des volontaires, les gestionnaires doivent s'attendre à ce que les propriétaires des ressources puissent venir de partout, qu'ils ne soient pas nécessairement connus des gestionnaires, qu'ils se portent volontaires sans attendre de compensation et qu'ils ne soient pas tenus d'avoir des connaissances en informatique. Pour faciliter l'intégration de nouveaux volontaires, l'installation et la configuration des logiciels pour le calcul doivent être aussi simple que possible [63]. En général, la participation des volontaires au système est principalement motivée par des intérêts personnels dans le projet ou par l'appartenance à une certaine communauté. Pour attirer davantage de volontaires, il est important d'avoir un modèle pour les encourager non seulement à contribuer au système, mais aussi à continuer à y contribuer une fois dans le système [24].

La nature non dédiée des ressources peut entraîner un taux de désabonnement élevé des volontaires. Pour éviter cela, une surveillance continue doit être mise en place. Les informations recueillies à l'issue de ce suivi doivent permettre de prendre des décisions et de développer des approches pour encourager les volontaires à rester dans le système, mais surtout de continuer à contribuer aux calculs [22, 24, 64]. Les volontaires peuvent contribuer au système pour plusieurs raisons :

- **Volontaire** : les volontaires contribuent sans attendre une rémunération en retour, juste pour le développement de la science et de la technologie ;

- **Information et feedback des résultats des travaux** : les volontaires sont informés de l'objectif du projet, des résultats attendus, les résultats obtenus, les performances attendues du système, les calculs effectués par leurs machines et la visualisation des statistiques de contributions ;
- **Utilisation non abusive de la ressource** : pour encourager les volontaires, il faut éviter une utilisation abusive de la ressource de calcul en arrêtant le calcul à chaque fois que le volontaire utilise son ordinateur et en évitant que les données stockées localement occupent trop d'espace disque ;
- **Récompense** : les volontaires reçoivent des crédits ou des prix pour leurs contributions. Généralement, les crédits virtuels, conséquence de l'effort de calcul effectué par la machine et la réputation de la qualité de service, sont attribués et utilisés pour classer les volontaires en fonction des contributions ;
- **Réciprocité** : les volontaires ayant apporté une grande contribution au système sont prioritaires lorsqu'ils ont besoin des ressources ;
- **Enchères** : une contribution financière est apportée aux volontaires en fonction de la quantité et de la qualité des ressources fournies au système.

3.3.2 Gestion du profil des machines

Les VCS sont un système hétérogène. L'hétérogénéité se situe au niveau matériel, des logiciels et du contexte (environnement pauvre en ressources). Tout dispositif capable d'effectuer des calculs peut être une ressource de calcul. Ce matériel peut être extrêmement divers en termes de vitesse et de type de processeur, de RAM, d'espace disque, de capacité d'E/S, de capacité de réseau, de système d'exploitation, etc. De manière générale, le profil du volontaire est composé d'attributs statiques et dynamiques [24, 64] :

- **Attributs statiques** : ce sont des attributs qui ne changent pas en peu de temps. Par exemple, la fréquence et le nombre de cœurs du processeur, la capacité de la RAM et du disque dur, le système d'exploitation, etc.
- **Attributs dynamiques** : ce sont les attributs qui ont un comportement aléatoire dans une courte période de temps. Par exemple, CPU libre, nombre de cœurs libres, l'espace de stockage libre sur la RAM et sur le disque dur, le niveau de batterie des ordinateurs et des téléphones portables, durée de vie d'un nœud de calcul, temps dédié moyen d'un nœud de calcul, la disponibilité, la fiabilité du volontaire, etc.

3.3.2.1 Gestion du profil d'activité du volontaire

Les attributs statiques et dynamiques détermineront la période d'inactivité d'une machine. La notion d'inactivité peut être définie de différentes manières :

- Taux d'utilisation du processeur est inférieur à 1% et si le clavier et la souris n'ont pas été utilisés il y a 30 secondes,
- Après 5mn d'inactivité du clavier et de la souris et la charge du processeur au bout de 1mn, 5mn, 10mn est inférieure à 0,4%, 0,3% et 0,1% que la machine est considérée comme étant inactive.

Les données sont donc collectées sur les attributs statiques et dynamiques afin de construire le profil de l'utilisateur. Les attributs dynamiques détermineront les périodes pendant lesquelles l'utilisateur est actif sur sa machine. Ils permettront ainsi de construire le profil d'activité du nœud de calcul. Les données sur les attributs dynamiques sont collectées pour une période de temps défini (par exemple, tous les quarts d'heure) et pour un nombre de semaines définie N (N également fixé). Ces données sont utilisées pour déterminer l'état de la machine afin d'en déduire les périodes de

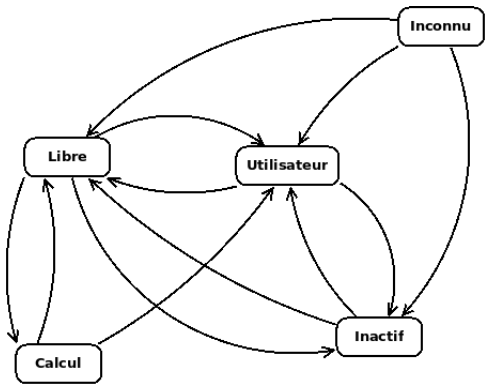

FIG. 3.3 – *Diagramme d'état transition d'un nœud de calcul*

disponibilité possibles. Comme présenté par la figure 3.3, un nœud de calcul a cinq modes possibles [59, 67]:

- **État libre** : dans cet état, la machine n'est pas utilisée par le propriétaire et peut donc être utilisée pour le calcul ;
- **État inactif** : la machine (en veille ou éteinte) n'est utilisée ni par le propriétaire ni pour le calcul ;
- **État calcul** : la machine n'est pas utilisée par le propriétaire, mais est utilisée par le VCS pour le calcul. À la fin d'un calcul, la machine revient à l'état libre pour donner la possibilité de recevoir d'autres calculs ;
- **État utilisateur** : la machine est considérée comme étant utilisée par le propriétaire. Ceci n'implique pas forcément la présence de l'utilisateur, par exemple le cas d'une machine en train de faire une sauvegarde ou un téléchargement ;
- **État Inconnu** : cet état consolide tous les états possibles de la machine où nous ne pouvons pas considérer de manière fiable que la machine est soit à l'état libre soit à l'état utilisateur. C'est le mode d'initialisation du profil d'activité, c'est-à-dire le profil après la première installation du système sur une machine volontaire. En fait, lors de la première installation du système, son profil d'activité est considéré comme étant dans un état inconnu pendant les dernières semaines.

Toutes les données sont stockées dans une base de données d'historique d'activité de la machine (voire figure 3.4). Cette base de données peut être vue comme un tableau infini, indexé par date d'évènement. Chaque enregistrement contient l'état de la machine pour une période de temps (par exemple, le quart d'heure), en indiquant par 1 si la machine est dans l'état libre et par 0 sinon [59, 67]. Par exemple, dans la figure 3.4, l'on peut voir une période où la machine a été détectée comme à l'état libre (suite de 1), puis une période où l'activité de l'utilisateur a été vue (suite de 0). Lors de la première installation d'une machine, elle est dans l'état inconnu et les données sont collectées pendant les N premières semaines pour remplir la base de données.

```
1011010010101011000111 0000...
```

FIG. 3.4 – *Exemple de base de données d'historique d'une machine*

3.3.2.2 Principaux défis de la gestion du profil d'activité

Pour terminer, notons que lors de la gestion du profil des utilisateurs, plusieurs défis doivent être relevés [67] :

- Gestion de la base de données des machines volontaires : il est important d'éviter que les données

qui sont stockées localement ne prennent trop d'espace au point de gêner l'utilisateur.

- Gestion des déconnexions : le système doit être capable de tenir compte des déconnexions fréquentes des machines (extinction par l'utilisateur, problème de connexion Internet, etc.)
- Ajout d'un nouveau volontaire : plusieurs mécanismes sont généralement utilisés lorsqu'un nouveau volontaire est ajouté au système. La nouvelle machine fera une requête au serveur afin d'initialiser sa base de données.

3.3.3 Planification des calculs et allocation des ressources

Le but de la collecte et du stockage des données sur les machines est de guider la planification et l'allocation des ressources de calcul [24]. En effet, pour planifier l'exécution d'un calcul, les profils des volontaires permettent de déterminer leurs périodes de disponibilité. La détermination de la période de calcul sur une machine volontaire est obtenue en utilisant les informations sur l'historique de disponibilités les N dernières semaines. Pour envoyer une tâche à un volontaire, le profil de ce dernier est analysé afin de déterminer les périodes d'utilisation et de disponibilité. Le planificateur pourra déterminer les périodes de disponibilité pendant lesquelles la machine peut être utilisée pour le calcul. Un système de planification efficace permet d'obtenir de meilleures performances possibles. Dès qu'une ressource est disponible, le système doit l'utiliser au maximum possible et s'adapter à des environnements dynamiques, hétérogènes et peu fiables. Le système doit être évolutif et capable de gérer les volontaires sans dégrader les performances à mesure que le nombre de volontaires augmente. De façon détaillé, lorsqu'un utilisateur soumet un calcul au système [60] :

- **Division du calcul** : le calcul est divisé en tâches parallèles ;
- **Identification de l'ensemble des volontaires** : une fois le calcul divisé en tâches, le planificateur utilisera le profil des machines volontaires pour déterminer l'ensemble des machines du système pouvant exécuter chaque tâche jusqu'à la fin. Le principal défi dans l'allocation des ressources est de répartir correctement les tâches en fonction de la capacité de chaque volontaire à les réaliser en temps opportun tout en garantissant une utilisation efficace des ressources, une réduction des coûts d'exploitation et une satisfaction accrue des utilisateurs ;
- **Distribution des tâches aux volontaires** : les tâches sont envoyées aux machines identifiées précédemment. Une bonne répartition permet l'exécution optimale des tâches et augmente le pool d'applications pouvant être exécutées sur la plate-forme.

Pour une planification réussie, les principales exigences du VCS sont [64] :

- Une division efficace des tâches pour qu'elles soient entièrement exécutées par les volontaires ;
- Une planification robuste des tâches et des ressources pour avoir un système avec de bonnes performances ;
- Une confiance permettant aux outils installés sur les machines des volontaires de ne pas endommager leurs systèmes ou s'attaquer à leur vie privée.

Le but de l'ordonnanceur ou du gestionnaire de tâches est de prendre des décisions sur le type et la quantité de calcul (tâches) à donner à un volontaire. En raison de l'hétérogénéité, des performances et de la disponibilité des ressources, il ne peut pas appliquer les mêmes critères de distribution à chaque machine. De plus, la performance et la disponibilité d'un volontaire peuvent soudainement changer. Les ordonnanceurs doivent pouvoir s'adapter aux caractéristiques de l'environnement de calcul volontaire. Il existe quatre stratégies pour la planification :

- **Sélection des ressources** : les volontaires sont classés selon des critères statiques et dynamiques. Les techniques probabilistes basées sur l'historique de disponibilité du volontaire permettent de distinguer les volontaires les plus stables des autres [60] ;
- **Exclusion des ressources** : les volontaires peuvent être exclus en utilisant les critères tels que la lenteur et le manque de fiabilité ;

- **Réplication des tâches** : les tâches sont souvent répliquées un nombre fixe de fois. Les méthodes probabilistes peuvent être utilisées pour faire varier le niveau de réplication en fonction de la volatilité des volontaires ;
- **Prédiction de la disponibilité des hôtes** : les méthodes de prédiction simples telles que le classifieur naïf de Bayes peuvent déterminer la disponibilité des ressources de calcul pour une période de temps donnée. Dans le chapitre 4, nous présentons en détail la prédiction de disponibilité des ressources de calcul dans un VCS.

En pratique, la mise en place d'une politique d'ordonnancement prenant en compte tous ces éléments est difficile. Ainsi, la plupart des politiques d'ordonnancement sont obtenues sur les heuristiques et sont classées en deux catégories :

- Politiques naïves : le calcul est attribué sans tenir compte de l'historique de calcul des volontaires. Nous pouvons citer :
  - Premier arrivé premier servi ou *First Come First Served* (FCFS) : les tâches sont envoyées aux volontaires disponibles pour les exécuter ;
  - Ordonnancement basé localité : les tâches sont de préférence envoyées aux hôtes qui ont déjà les données nécessaires pour les exécuter ;
  - Assignation aléatoire : les tâches sont choisies au hasard et envoyées aux volontaires.
- Politiques basées sur les connaissances acquises : ces politiques tiennent compte de l'historique des volontaires et l'ensemble de la communauté. Nous pouvons citer :
  - Seuils fixes : l'ordonnanceur vérifie les taux de disponibilité et de fiabilité du volontaire, s'ils dépassent un certain seuil prédéfini, l'ordonnanceur lui attribue une tâche ;
  - Seuils variables : l'ordonnanceur fait varier les seuils lors de l'exécution des tâches. Si le nombre de tâches en attente de distribution est supérieur au nombre de volontaires, le seuil diminue, sinon, il augmente ;
  - Politique d'ordonnancement du *World Community Grid* (WCG) : le calcul est alloué en fonction du temps de calcul moyen du volontaire ;
  - Politique adaptative : utilise les techniques de l'apprentissage automatique pour réinitialiser les paramètres d'ordonnancement. Cette politique est présentée en détail au chapitre 4.

Dans l'approche centralisée, un serveur central conserve toutes les informations sur les ressources, l'état d'exécution des tâches et est responsable de la planification. Dans l'approche distribuée, la décision d'ordonnancement est répartie entre les nœuds : chaque nœud possède des informations partielles sur les ressources et l'état d'exécution des tâches. Dans l'approche hiérarchique, la décision d'ordonnancement est prise de manière hiérarchique (par exemple, méta-ordonnanceur - ordonnanceur de haut niveau et ordonnanceur local - ordonnanceur de bas niveau). Un planificateur de haut niveau alloue des tâches aux planificateurs de bas niveau, tandis qu'un planificateur de bas niveau alloue directement des tâches aux machines de son site.

3.3.4 Réception des tâches, exécution, retour et validation des résultats

Deux approches sont généralement utilisées par les volontaires pour avoir accès aux tâches à exécuter. Dans la première approche, le serveur ou le coordinateur des ressources identifie la période de disponibilité du volontaire et lui envoie le calcul ; dans la seconde, le volontaire demande du calcul dès qu'il est libre. Dans les deux cas, l'activité de calcul ne doit pas nuire au volontaire et il faut s'assurer que les résultats fournis par les volontaires sont corrects.

3.3.4.1 Exécution des tâches et retour des résultats

Pendant le fonctionnement du VCS, un contrôle et une régulation de l'utilisation des ressources doivent être mis en place afin que les applications dites invitées ne perturbent pas le volontaire.

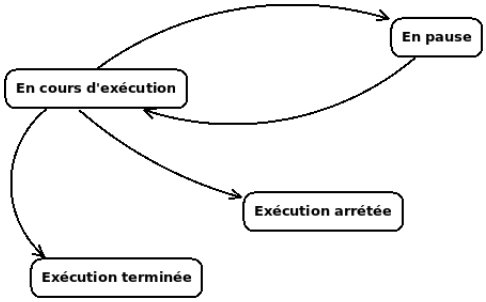

FIG. 3.5 – *Diagramme à état de l'application invitée montrant les événements et leurs conséquences*

Pour ce faire, ces applications ne doivent être activées que lorsqu'aucune activité de l'utilisateur n'est détectée. Si l'application invitée est en cours d'exécution et détecte une activité de l'utilisateur, elle doit immédiatement se mettre en pause ou être arrêtée. On parle de non-intrusion [24]. La figure 3.5 présente un diagramme d'état de l'application invitée montrant les évènements et leurs conséquences.

Comme le montre la figure 3.5, certains évènements tels que l'activité de la souris et du clavier, l'augmentation de la charge du processeur et l'utilisation accrue de la bande passante du réseau peuvent provoquer un changement d'état du système. Lorsque la machine entre dans l'état libre, les éventuels travaux en attente sont exécutés. Le retour de la machine en mode normal est transparent pour l'utilisateur.

Étant donné que le volontaire peut quitter le système à tout moment, un système de tolérance aux pannes doit être mis en place pour permettre un fonctionnement optimal. Plusieurs systèmes de tolérance aux pannes peuvent être utilisés [24, 63] :

- **Les techniques réactives** : les calculs sont dupliqués sur plusieurs hôtes.
- **Les techniques proactives** : elles déterminent les ressources fiables avant de lancer les calculs.

3.3.4.2 Validation des résultats

Une fois l'exécution de la tâche terminée, le résultat est envoyé au serveur/client/nœud coordinateur en fonction de l'approche qui a été choisie pour implémenter le système. Ces résultats peuvent être erronés pour plusieurs raisons : mauvais fonctionnement du matériel/logiciel ou comportement malveillant de certains volontaires. C'est pourquoi le système est généralement doté d'un mécanisme de vérification pour évaluer et valider les résultats obtenus. L'approche la plus simple et la plus utilisée est appelée *le vote à la majorité*. Elle consiste à répliquer la même tâche entre plusieurs volontaires et à comparer leurs résultats à la fin des calculs. Un résultat est accepté si la majorité des volontaires renvoient le même résultat. BOINC, par exemple, utilise le terme quorum pour désigner le nombre de résultats qui doivent correspondre pour valider le résultat final. Ce quorum est fixé par projet [63]. L'inconvénient est le niveau élevé de redondances de calcul. Le contrôle par sondage permet de palier au problème de redondance des calculs. Cette approche consiste à envoyer plusieurs tâches aux volontaires pour tester leur efficacité. Les volontaires qui réussissent le test sont sollicités pour les calculs et ceux qui échouent le test sont écartés. Notons que des volontaires peuvent réussir le test et abandonner le projet par la suite. Avec l'approche basée sur la crédibilité, un volontaire n'est pas digne de confiance tant qu'il ne l'a pas prouvé [63].

3.3.5 Sécurisation des ressources

L'exécution du code sur une machine volontaire nécessite un mécanisme de protection de celle-ci. En particulier, les données, les processus et les sessions réseau de l'utilisateur doivent être isolés de ceux nécessaires à l'exécution des tâches de calcul. Le système doit s'assurer que les logiciels utilisés par les volontaires ne les exposent pas aux attaques informatiques, n'exposent pas leurs machines à

des risques pourraient perturber leurs activités ou encore leurs données [63]. Ces logiciels ne doivent pas avoir accès aux données des volontaires que ce soit en lecture ou en écriture. Une attaque contre le système pourrait avoir des graves conséquences telles que la démission de tous les volontaires.

3.4 Exemples de systèmes

Les premiers systèmes de calcul haute performance basés sur les machines volontaires sont apparus en 1990. Les tous premiers systèmes étaient le *Great Internet Mersenne Prime Search* (GIMPS)[3] et distributed.net[4]. Avec la vulgarisation des ordinateurs personnels de plus en plus puissants et la connexion Internet de plus en plus répandue, plusieurs projets tels que Africa@home [68], LHC@home[5], SETI@home[6], XtremWeb[7], Folding@home[8], Prediction@home [69], etc. ont vu le jour. Certains de ces systèmes ont conduit à de nombreuses découvertes, notamment la compréhension des protéines et la prédiction de leur structure [70]. Cette section présente quelques systèmes de calcul sur machines volontaires . La section 3.4.1 présente BOINC, la section 3.4.2 présente Africa@home, la section 3.4.3 présente seti@home et la section 3.4.4 présente XtremWeb.

3.4.1 BOINC

Berkely Open Infrastructure for Network Computing (BOINC) [65, 71, 72] est la plateforme de calcul sur machines volontaire la plus populaire au monde. Elle a été lancée au *Space Sciences Laboratory* de l'Université de Californie aux États-Unis en 2002 et propose une interface permettant aux volontaires de choisir les projets qu'ils souhaitent soutenir et décider des ressources informatiques à consacrer à chacun d'eux.

BOINC est composé d'un serveur qui héberge les tâches et d'un client qui s'installe et se configure sur les machines des volontaires. Ces clients permettent d'extraire les tâches à partir des projets spécifiés par le volontaire, de les exécuter et de renvoyer les résultats au serveur. Lorsque le serveur reçoit les résultats d'un client, il le valide [71]. Les premiers projets utilisant BOINC ont été lancés en 2004. Aujourd'hui, l'on dénombre plus de 60 projets dans un large éventail de domaines dont Milkyway@home qui étudie la structure galactique, Einstein@home de recherche des ondes gravitationnelles, Rosetta@home qui étudie les protéines d'importance biomédicale et climatePrediction.net qui étudie le changement climatique à long terme [7, 60, 62].

BOINC rassemble près de 4 millions de volontaires, avec plus de 700 000 appareils actifs. Ces appareils ont environ 4 millions de cœurs et fournissent collectivement une puissance de calcul de plus de 90 PFLOPS [73].

3.4.2 Africa@home

Le projet Africa@home [68] a été démarré en 2007 au Département de Mathématiques de l'Université des Sciences et de la Technologie Kwame Nkrumah au Ghana en collaboration avec l'Université de Genève, l'Institut Tropical Suisse, le CERN, et l'ICVolunteers (*International Conference Volunteers*). Basé sur BOINC, son objectif était d'utiliser un système de calcul sur machines volontaires pour disposer d'une puissance de calcul suffisante pour la lutte contre les maladies infectieuses, les catastrophes naturelles, ou des simulations de grande envergure en Afrique. Une première phase de test sur quelques mois avec 500 volontaires a permis de réaliser des simulations équivalentes à 150 ans

3. https://www.mersenne.org
4. https://www.distributed.net
5. https://lhcathome.web.cern.ch
6. https://setiathome.berkeley.edu
7. http://xtremweb.gforge.inria.fr
8. https://foldingathome.org

de temps de calcul. Les résultats de la première simulation a nécessité une puissance de calcul de 6 TFLOPS [68].

3.4.3 SETI@home

Search for Extra-Terrestrial Intelligence at home (SETI@home) est un projet scientifique faisant partie du programme SETI pour la recherche de l'intelligence extraterrestre. Il consiste à collecter les données spatiales constituées de signaux radio, à les stocker et à les traiter. L'analyse et le traitement des données peuvent être parallélisées et ne nécessitent pas de communication entre les nœuds de calcul [60, 70]. Pour ce faire, chaque parcelle de l'espace est scrutée et les données collectées sont transmises à des machines volontaires sur Internet.

SETI@home a rassemblé près de 3,5 millions de volontaires, avec plus de 500 000 actifs et a fournit un total de plus de 100 TFLOPS de puissance de calcul. Il s'agit du premier déploiement réussi d'un système de calcul sur machines volontaires à grande échelle et est actuellement le VCS le plus puissant du monde [74].

3.4.4 XtremWeb

XtremWeb [75] est un système de calcul sur machines volontaires développé au Laboratoire de Recherche en Informatique (LRI) de Paris XI. Il consiste en un environnement de développement des applications nécessitant du calcul intensif. Contrairement à SETI@home, il n'est pas lié à une application particulière, mais, offre un ensemble de bibliothèques permettant le développement et l'exécution de tous types d'applications nécessitant le calcul intensif.

Des ordinateurs de bureau volontaires appartenant à des institutions disposant d'une connexion Internet collaborent avec le serveur XtremWeb qui organise et leur distribue les calculs. Xtremweb a été testé sur une centaine de volontaires du LRI dans le cadre du projet Observatoire Pierre Auger, qui étudie des particules de très haute énergie.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les systèmes de calcul sur machines volontaires comme un moyen attractif d'utiliser de vastes ressources de calcul connectées à Internet en tant que système de calcul haute performance. Les propriétaires des machines ou encore les volontaires fournissent leur ressources de calcul (CPU et mémoire) pendant leurs périodes d'inactivité pour résoudre un problème scientifique nécessitant une grande puissance de calcul. Le seul financement requis est le développement initial des plates-formes logicielles, leur maintenance, ainsi que la fourniture d'une assistance aux utilisateurs. Il n'y a pas de coût lié à l'achat ou la location du matériel. Les VCS offrent ainsi une plateforme simple d'utilisation, accessible au grand public, évolutive et peu coûteuse.

Nous avons également noté un grand nombre de défis pour avoir un système performant : les volontaires doivent être suffisamment nombreux à s'être enrôlés et à participer aux calculs ; la vérification et la sécurisation des résultats doivent être faites ; la sécurité des ressources doit être fait pour empêcher les volontaires malveillants de nuire aux autres ; un système permettant une bonne gestion des ressources de calcul doit être mis en place. Dans le chapitre 4, nous nous appesantirons sur le problème de la disponibilité des ressources de calcul.

Chapitre Quatre

Prédiction de disponibilité des ressources basée sur l'Apprentissage Automatique

Dans le chapitre 3, nous avons présenté les systèmes de calcul sur machines volontaires [60] comme des systèmes qui exploitent le temps libre des machines connectées à un réseau pour fournir du calcul haute performance. Avec ces systèmes, le partage des ressources de calcul avec les propriétaires, le fait que la maintenance de ces ressources soit assurée par les propriétaires, l'instabilité de l'énergie électrique et de l'Internet posent le problème de disponibilité des ressources de calcul dans une période donnée. Les machines peuvent devenir indisponibles sans connaissance préalable et rendre le système non fiable. En effet, une ressource sur laquelle une tâche est démarrée et qui devient indisponible avant la fin de la tâche peut entraîner d'énormes pertes et affecter les performances globales du système [31, 32, 33, 76, 77, 78]. Pour réduire ces pertes, nous exploitons l'idée qui consiste à collecter les données sur la disponibilité des ressources de calcul sur une période donnée et d'utiliser ces données pour choisir, pour un calcul donné, uniquement les ressources susceptibles d'être disponibles jusqu'à la fin du calcul. Les solutions proposées dans la littérature sont basées sur les règles, sur les statistiques et sur l'apprentissage automatique [31, 79]. Dans ce chapitre, nous nous intéressons particulièrement aux solutions basées sur l'apprentissage automatique ou *Machine Learning* (ML). Dans ce qui suit, nous commençons par la prédiction de la disponibilité des ressources en général dans la section 4.1. Les approches de prédictions basées sur le ML et leur application aux systèmes de calcul sur machines volontaires sont décrites dans la section 4.2. Notre approche est présentée dans la section 4.3 et la conclusion à la section 4.4.

4.1 Prédiction de disponibilité des ressources

La prédiction de disponibilité des ressources pour résoudre un problème implique d'identifier le type, la quantité de ressources ainsi que le moment le plus approprié pour les utiliser afin de résoudre le problème. Dans cette section, nous présentons la prédiction de disponibilité des ressources en général et la prédiction de disponibilité des ressources de calcul en particulier. Ainsi, la section 4.1.1, présente les généralités ; la section 4.1.2 présente la prédiction de disponibilité d'une ressource individuelle et d'un ensemble de ressources. Enfin, la section 4.1.3 les prédicteurs de disponibilité.

4.1.1 Généralités

Dans toutes les organisations, la planification des ressources (humaines, financières, matérielles, etc.) est une tâche complexe car ces ressources sont généralement variées et de nature dynamique. Lorsque la planification est réalisée de manière efficace, elle apporte une valeur ajoutée significative

à la structure. Une planification efficace dépend de la capacité à évaluer avec précision les besoins futurs en ressources. Dans la suite seront présentées les généralités sur la prédiction de disponibilité des ressources et le cas particulier des ressources de calcul dans un système de calcul sur machines volontaires.

4.1.1.1 Généralités sur la prédiction de disponibilité des ressources

Une ressource est dite disponible dans un intervalle de temps donné si elle est accessible et utilisable dans cet intervalle de temps. Prédire la disponibilité d'une ressource dans un intervalle de temps consiste à déterminer si cette ressource sera accessible et exploitable dans cet intervalle de temps. Prédire la disponibilité d'un ensemble de ressources implique de déterminer si ces ressources seront accessibles et exploitables dans cet intervalle de temps. Le problème de la prédiction de disponibilité des ressources est posé de la manière suivante : étant donné un ensemble de ressources, quelles sont celles qui sont susceptibles d'être disponibles sur une période de temps allant de t à $t+p$, où t est le moment où la prédiction est faite et p la taille de l'intervalle de prédiction. La prédiction de disponibilité des ressources a une grande variété d'applications : en finance, par exemple, elle est utilisée à long terme pour estimer les besoins futurs en capital d'une entreprise, en gestion des ressources humaines elle permet d'identifier les futurs besoins en main d'œuvre, en marketing elle permet de connaître les prévisions de ventes utilisées pour la planification à moyen et à long terme.

Pour déterminer si une ressource ou un ensemble de ressources sera disponible à un moment donné, les deux principales métriques suivantes sont utilisées [79] :

- Le temps hors service : temps pendant lequel la ressource est hors ligne. Si c'est un produit par exemple, c'est le temps pendant lequel il n'y en a plus en stock ;
- Le temps de service : temps pendant lequel la ressource est accessible et exploitable. Par exemple, le temps pendant lequel un volontaire laisse sa machine pour le calcul dans un VCS.

Les données sur la disponibilité des jours/semaines/mois précédents (généralement disponible est codifié par 1 et non disponible par 0) sont utilisées pour étudier les différents scénarios d'évolution et établir le schéma le plus probable.

4.1.1.2 Cas particulier de la prévision des ressources de calcul

La prédiction de disponibilité des ressources dans les systèmes de calcul sur machines volontaires permet, pour un calcul donné, de déterminer l'ensemble des ressources susceptibles d'être disponibles jusqu'à la fin du calcul [80, 81]. La gestion de la disponibilité est essentielle à la fiabilité et la réactivité des services de calcul déployés. Dans ces systèmes, l'indisponibilité fréquente des ressources de calcul met les planificateurs au défi d'effectuer des placements des tâches efficaces. La manière dont les ressources deviennent indisponibles a des effets différents sur différentes applications en fonction de leur exécution et de leur capacité à être vérifiées ou répliquées [82].

Un problème majeur des ressources de calcul dans les VCS est la faible disponibilité et le fait que les hôtes soient incontrôlables [31]. Le taux d'indisponibilité (plusieurs fois par jour) et le taux de désabonnement sont souvent élevés. Ainsi, caractériser, analyser et modéliser cette disponibilité est une condition essentielle pour élargir les types d'applications qui peuvent être exécutées dans les VCS.

4.1.2 Disponibilité individuelle et collective des ressources

La disponibilité des ressources représente la probabilité de disponibilité à un moment donné ou dans une période donnée (par exemple, dans une semaine). Elle est obtenue à partir du profil d'activité de ces ressources. La prédiction peut prévoir une échelle de temps à court ou à long terme [31]. Prédire à court terme consiste à prédire la disponibilité donnée dans un intervalle de temps d'une à plusieurs heures. La prédiction à court terme est utile pour le déploiement de tâches individuelles [83]. La disponibilité individuelle d'une ressource est calculée à partir du profil d'activité des ressources

contenant l'historique de l'utilisation de la ressource. Dans le cas de la prédiction de disponibilité des ressources de calcul, il existe deux types de prédictions : la prédiction d'une ressource individuelle et la prédiction d'une collection de ressources [31]. La disponibilité individuelle d'une ressource de calcul peut être catégorisée en [84, 85, 86] :

- **Disponibilité de l'hôte** : fait référence au fait qu'un hôte est accessible et prêt à fournir les ressources pour le calcul. Il est généralement codifié par 0 pour hôte non disponible et 1 pour hôte disponible. Les principales causes de l'indisponibilité d'un hôte sont une panne de courant, un arrêt, un redémarrage ou encore une panne de la machine.
  Deux types d'hôtes peuvent être identifiés : ceux qui sont toujours actifs et ceux qui ont des modèles de disponibilité cycliques : par exemple, toujours indisponibles pendant des heures de travail. Ces informations peuvent être utilisées pour fournir des garanties de disponibilité plus élevées [85].
- **Disponibilité du processeur** : est la valeur en pourcentage qui quantifie la fraction de CPU qui peut être exploitée par le système. Les facteurs qui affectent généralement la disponibilité du processeur sont les processus exigeants en calcul au niveau du système et de l'utilisateur. Lorsque l'hôte n'est pas disponible, cela implique également l'indisponibilité du processeur. Généralement, les machines sont occupées par les utilisateurs et seules les ressources excédentaires sont apportées au système de calcul. Dans certains systèmes, une machine est indisponible dès que l'on détecte la présence de l'utilisateur, c'est-à-dire lorsque l'on détecte l'activité du clavier ou de la souris. Dans d'autres systèmes, la machine est disponible dès qu'elle est connectée au réseau. Par la suite, on évalue le pourcentage d'utilisation du processeur dans le but de lancer le calcul pour utiliser pleinement le processeur ; la machine est donc considérée comme disponible lorsque le pourcentage de CPU consommé par son utilisateur local n'atteint pas un seuil donné.
- **Disponibilité de l'exécution des tâches** : détermine si une tâche peut être exécutée sur la machine ou non, en fonction des exigences de l'hôte. Elle peut être codée par 0 pour machine indisponible et 1 pour machine disponible. Les causes de l'indisponibilité incluent une activité prolongée de téléchargement ou de utilisation de la machine. Lorsque l'hôte est indisponible, cela implique aussi l'indisponibilité de l'exécution.

Lors des calculs, les interruptions des ressources individuelles sont fréquentes et ont de nombreuses conséquences [31] :

- Perte de calcul : dans certains cas, tous les travaux depuis le dernier téléchargement des résultats sont perdus (par exemple, si les hôtes abandonnent définitivement le système) ;
- Retard de l'achèvement du calcul : le calcul effectué par un hôte dans une unité de temps étant le produit de sa capacité de calcul et de sa disponibilité moyenne, chaque échec du calcul implique moins de travail dans une unité de temps, et donc un éventuel retard dans l'achèvement du calcul ;
- Dégradation de la disponibilité générale du service : les interruptions individuelles entraînent inévitablement une défaillance globale du service ;
- Nécessité de remplacement / actions de migration temporaire ou permanente : dans certains systèmes, les hôtes sur lesquels les calculs ont échoué déclenchent les actions de migration et de réplication des calculs et par conséquent, de nouvelles planifications. Tout ceci engendre des surcoûts et réduit considérablement l'efficacité globale du déploiement ainsi que de la capacité de calcul de ces infrastructures.

La disponibilité collective quantifie le nombre de ressources disponibles dans le système à un moment. En effet, pour un pool de N hôtes dans un système de calcul sur machines volontaires, il s'agit de déterminer n hôtes disponibles à un moment. Le but de tout système de calcul sur machines volontaires est d'offrir une disponibilité collective pour un service sur une longue période [31, 83]. Globalement, la modélisation du comportement de disponibilité des hôtes répond à deux objectifs

[31] :

- Analyser le comportement de disponibilité passé des hôtes individuels et de leurs groupes : permet l'estimation de la capacité de calcul cumulée, la compréhension des distributions de disponibilité et des taux d'échec ; connaissance de la disponibilité et des tendances à long terme ; le filtrage des hôtes par des critères spécifiques tels qu'un comportement stable, la détection d'anomalies à grande échelle ou d'échecs telles que les pannes partielles du réseau.
- Prédire la disponibilité à court terme des hôtes individuels ou de leurs groupes : court terme signifie une à plusieurs heures. La prédiction de disponibilité des hôtes à long terme (au-delà d'un jour) n'est habituellement pas réalisable sauf pour une petite classe d'hôtes avec un comportement très régulier [85].

4.1.3 Prédicteurs de disponibilité

Dans les VCS, pour un calcul donné, le planificateur doit choisir les volontaires les plus susceptibles d'être disponibles jusqu'à la fin de son exécution. Pour ce faire, le prédicteur de disponibilité doit déterminer la disponibilité des volontaires de cet instant à la fin éventuelle de l'exécution du calcul, c'est-à-dire l'exécution de toutes les tâches dont le calcul a été composé. Supposons que nous sommes à l'instant t, le principal problème est de déterminer la disponibilité de chaque ressource sur une période de temps allant de t à $t + p$ [79, 83, 86].

Plusieurs facteurs peuvent affecter le temps d'exécution d'une application sur un VCS, notamment la disponibilité de l'hôte, la charge du processeur et l'activité des utilisateurs. Le but des prédicteurs de disponibilité est de déterminer les ressources de bonne qualité. Pour ce faire, ils utilisent les traces de disponibilité détaillées des ressources. La trace idéale contient les caractéristiques sur les ressources de calcul telles que les informations sur la disponibilité de l'hôte, la disponibilité du processeur, la disponibilité pour l'exécution et la durée des tâches exécutées. De façon globale, les prédicteurs de disponibilité doivent pouvoir [79, 83, 86] :

- Faire la prévision de disponibilité à court terme : il s'agit de déterminer la disponibilité d'un hôte spécifique dans un intervalle de temps $[t, t + p]$ appelé intervalle de prédiction. Notons que t est le moment de la prédiction et p la longueur de l'intervalle de prédiction. Un hôte considéré comme disponible dans l'intervalle $[t, t + p]$ s'il est complètement disponible sans interruptions dans cet intervalle ;
- Identifier les machines en panne : il s'agit de déterminer les machines qui sont en panne. Pour ce faire, nous distinguons les pannes transitoires et les pannes permanentes. Un nœud qui a une panne transitoire (due par exemple à une coupure d'électricité), peut reprendre l'exécution du calcul qui lui a été alloué dès son retour. Les pannes permanentes se produisent lorsque les nœuds de calcul deviennent indisponibles pendant une longue période. Cela se produit lorsque l'utilisateur éteint sa machine, lorsque la machine est en panne ou lorsque le volontaire quitte le système ;
- Déterminer les caractéristiques de l'ensemble du système et des sous-ensembles d'hôtes individuels pour guider la planification. Par exemple, il faut être à mesure de déterminer les performances du système, des hôtes et des sous-ensembles d'hôtes en fonction de leur disponibilité.
- Déterminer la durée de vie d'une ressource : le prédicteur doit pouvoir déterminer pour une ressource donnée, la période entre deux pannes consécutives. La distribution des durées de vie d'une classe de ressources est essentielle pour comprendre les types d'applications que cette classe peut desservir de manière fiable ;
- Déterminer la stabilité d'une ressource ou encore d'un groupe de ressources : il s'agit de définir la capacité d'une ressource ou d'un groupe de ressources à exécuter plusieurs travaux d'une durée donnée. La sélection d'une ressource à haute disponibilité et l'exécution de plusieurs tâches ayant les mêmes caractéristiques sont essentielles pour obtenir une faible surcharge.

Les prédicteurs de disponibilité présentés dans la littérature sont basés sur des règles, des statistiques ou de l'apprentissage automatique [79, 84] :

- Approches basées sur les règles : déterminent la prédiction en dérivant des règles utiles afin de rechercher les ressources pertinentes, puis en attribuant des tâches aux ressources disponibles. Par exemple, la théorie des ensembles approximatifs utilise une analyse d'ensembles approximatifs pour prédire le comportement d'un nœud.
- Approches basées sur les statistiques : les approches basées sur les statistiques calculent une valeur numérique utilisée par le planificateur pour sélectionner les ordinateurs sur lesquels exécuter les applications. Par exemple, l'analyse descriptive des données et l'utilisation de tests statistiques permettent de déterminer la disponibilité d'un hôte, d'un ensemble d'hôtes et l'association qui peut exister entre les différents composants de l'ordinateur. Les techniques d'ajustement de modèle paramétrique (méthode de Weilbull) et non paramétriques (méthode de ré-échantillonnage et méthode binomiale) permettant de prédire la durée de disponibilité d'une machine.
- Approches basées sur l'apprentissage automatique : par exemple, une approche basée sur l'apprentissage non supervisé effectue une analyse de regroupement sur les données historiques de l'utilisation des ressources afin de découvrir des modèles prototypes d'utilisation pour la prédiction de disponibilité. Les approches basées sur l'apprentissage automatique sont présentées plus en détail dans la section 4.2.

4.2 Approches de prévision de disponibilité des ressources basées sur l'Apprentissage Automatique

Un système de prédiction de disponibilité basé sur le ML utilise les méthodes d'apprentissage pour prédire la disponibilité d'une ressource ou d'un ensemble de ressources dans un intervalle de temps d'une longueur donnée en fonction des traces de disponibilité. Pour ce faire, un modèle prédictif est construit à l'aide de la trace de disponibilité des ressources. Ainsi, la section 4.2.1 présente l'apprentissage automatique, les sections 4.2.2 et 4.2.3 présentent respectivement les méthodes de classification et de régression permettant de faire la prédiction des ressources. Nous terminons par une application aux systèmes de calcul sur machines volontaires à la section 4.2.4.

4.2.1 Apprentissage automatique

L'apprentissage automatique utilise les approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir des données. Pour ce faire, une formule mathématique empirique est construite à partir des données historiques afin de pouvoir faire des prédictions pour de nouvelles données [87]. En effet, étant donné un problème, la première étape consiste à déterminer le modèle permettant de le résoudre. Ce modèle sera ensuite entraîné sur une grande quantité de données. Une fois entraîné, l'on obtient un modèle capable de faire les prédictions.

4.2.1.1 Les données

Les données utilisées dans l'apprentissage automatique consistent en une collection d'enregistrements. Chaque enregistrement est également appelé instance. L'ensemble des données est divisé en deux parties : un ensemble d'apprentissage utilisé pour construire le modèle prédictif ; un ensemble de test utilisé pour tester et évaluer le modèle construit.

1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1

TAB. 4.1 – *Exemple de trace de disponibilité d'une machine en un jour*

4.2.1.2 Différents types d'apprentissage

L'apprentissage automatique est divisé en deux principaux groupes : apprentissage supervisé et l'apprentissage non supervisé :

- On parle d'apprentissage non supervisé lorsque les données d'entraînement ne sont pas annotées. Le but de l'algorithme d'apprentissage est donc de déterminer la structure plus ou moins cachée de ces données en les regroupant de telle sorte que les données similaires soient dans le même groupe.
- On parle d'apprentissage supervisé lorsque les données d'apprentissage sont étiquetées et que l'objectif est de prédire l'étiquette de la nouvelle donnée. On parle de classification lorsque les étiquettes sont discrètes et de régression lorsque les étiquettes sont continues.

4.2.1.3 Différentes étapes d'un projet d'apprentissage automatique

Les principales étapes sont les suivantes [88] :

1. Acquisition de données : cette étape consiste en la collecte d'une grande quantité de données qui seront utilisées pour l'apprentissage ;
2. Prétraitement des données : en général, les données doivent être retouchées pour les mettre sous la forme exploitable par un algorithme d'apprentissage ;
3. Organisation des données : les données obtenues lors de la phase précédente sont divisées en données d'apprentissage pour entraîner le modèle et en données de test pour vérifier la performance des résultats ;
4. Construction du modèle : le modèle est construit à l'aide des données d'entraînement de l'étape précédente. Avant de choisir un modèle, différents modèles peuvent être testés. La construction du modèle se fait grâce à un algorithme d'apprentissage. Il prend les données et produit en sortie un modèle, qui à son tour permettra d'effectuer les prédictions sur de nouvelles données. Dans certains cas pour un même problème, plusieurs modèles peuvent être combinés. La sélection de modèles permet de choisir le modèle le plus approprié en fonction des données.
5. Évaluation du modèle : les données de test sont utilisées pour vérifier la pertinence du modèle ;
6. Utilisation : le modèle est utilisé pour résoudre le problème pour lequel il a été construit.

4.2.1.4 Problème de prédiction de disponibilité des ressources en utilisant le ML

La prédiction de disponibilité des ressources dans un intervalle de temps détermine son état (0 pour indisponible et 1 pour disponible) dans cet intervalle de temps. Le problème est donc de déterminer cet état dans un intervalle de temps donné. Pour ce faire, une partie de la trace de disponibilité de la ressource est extraite, formatée, découpée pour constituer un jeu de données. Le jeu de données est construit de telle sorte que la méthode d'apprentissage puisse déterminer la disponibilité dans un intervalle de temps donné.

Formellement, le problème de prédiction de disponibilité des ressources utilisant l'apprentissage automatique est défini comme suit : soit t le moment de prédiction, la disponibilité des ressources est déterminée dans un intervalle de prédiction allant de t à $t+pil$ en utilisant l'historique de disponibilité de la période d'entraînement de t-til à t et les méthodes d'apprentissage automatique. Le paramètre pil (*prediction interval length*) est la longueur de l'intervalle de prédiction exprimée en heures et til (*training interval length*) est la longueur de l'intervalle d'entraînement exprimée en nombre de jours. Le tableau 4.1 présente la trace de disponibilité d'une machine pour $pil = 1$ et $til = 1$.

4.2.2 Approches de classification

Dans un problème de classification, le but est de construire une fonction $f : X \to Y$ qui assigne à un vecteur d'attributs x_i, une étiquette ou une classe y_i avec l'erreur minimale [87]. Dans cette section, nous présentons deux exemples qui seront utilisés plus tard dans ce travail : la méthode naïve de Bayes et le perceptron multicouche.

4.2.2.1 Méthode naïve de Bayes

Le classifieur naïf de Bayes est basé sur l'application du théorème de Bayes avec l'hypothèse naïve d'indépendance entre chaque paire d'attributs [87]. Formellement, étant donné un vecteur d'attributs $x_i = (x_{i1},x_{i2},\ldots,x_{in})$, la valeur prédite y_i est celle qui maximise la probabilité a posteriori $P(y_i \mid x_{i1},\ldots,x_{ip})$.

$P(y_i \mid x_{i1},\ldots,x_{ip}) \propto P(y_i) \prod_{j=1}^{p} P(x_{ij} \mid y_i)$

$$y = \arg\max_{y_i} P(y_i) \prod_{j=1}^{p} P(x_{ij} \mid y_i) \tag{4.1}$$

$P(y_i)$ est la probabilité a priori pour les valeurs cibles et $P(x_{ij} \mid y_i)$ la probabilité conditionnelle pour chaque valeur cible y_i. Ces probabilités sont estimées à partir des données d'entraînement.

4.2.2.2 Perceptron multicouche

Le perceptron multicouche ou *Multi-Layer Perceptron* (MLP) est composé d'un assemblage interconnecté de nœuds (neurones) et de liens dirigés. Une couche est un ensemble de neurones qui n'ont aucune connexion entre eux [87]. Dans le cas des réseaux de neurones à propagation avant, les nœuds d'une couche sont connectés uniquement aux nœuds de la couche suivante. La couche d'entrée reçoit les données entrantes et la couche de sortie fournit le résultat. Une ou plusieurs couches intermédiaires ou couches cachées participent au traitement. L'objectif de MLP est de déterminer un ensemble de poids qui minimise la somme totale des erreurs quadratiques.

$$E(w) = \frac{1}{2} \sum_{i=1}^{p} \|y_i - \hat{y_i}\|^2 \tag{4.2}$$

Où :

- y_i sorties réelles
- $\hat{y_i}$ sorties souhaitées

Pour le cas du perception simple c'est-à-dire monocouche $\hat{y_i}$ est déterminé par la formule suivante :

$$\hat{y_i} = f(w_0 + w_1 x_{i1} + \ldots + w_d x_{id})$$

Où:

- $w_1, \ldots w_d$ sont les poids des liens ;
- $x_{i1} \ldots x_{id}$ sont les valeurs d'attribut d'entrée ;
- f est une fonction d'activation.

Bien que meilleur que le classifieur naïf de Bayes en termes de taux de prédiction, MLP prend beaucoup de temps et nécessite beaucoup de données pour l'entraînement.

4.2.3 Approches de régression

Dans un problème de régression linéaire, la cible prédite par la fonction de prédiction est une valeur numérique [89]. La relation entre la cible y_i et le vecteur d'attributs x_i est donnée par l'équation de régression linaire suivante :

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i \tag{4.3}$$

Les paramètres β_1,β_2,...,β_p sont des cœfficients de régression, β_0 est l'interception et ε_i est l'erreur d'approximation.

Sous la forme matricielle, la relation entre y_i et x_i est donnée par l'équation suivante.

$$y = X\beta + \varepsilon$$

$$\text{Où : y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_n \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

La tâche de régression linéaire consiste donc à estimer les cœfficients de régression qui minimisent le vecteur d'erreur ε sur la base des données d'entrée. Plusieurs méthodes de régression dont, la méthode de Lasso, existent. La régression *LASSO* [89, 90] (*Least Absolute Shrinkage and Selection Operator*) minimise le critère des moindres carrés en imposant une pénalité de l_1 aux cœfficients β ($\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$). L'estimateur *LASSO*, noté $\hat{\beta}_{Lasso}$ est :

$$\hat{\beta}_{Lasso} = \arg\min_{\beta \in \mathbb{R}^p} (\|y - X\beta\|^2 + \lambda \|\beta\|_1) \tag{4.4}$$

$\lambda \geq 0$ est un paramètre de réglage à déterminer.

La solution obtenue est dite parcimonieuse car elle comporte de nombreux cœfficients nuls. LASSO peut être utilisé pour sélectionner des variables dans un grand modèle.

4.2.4 Application aux systèmes de calcul sur machines volontaires

Les systèmes de prédiction de disponibilité des ressources utilisant les méthodes d'apprentissage automatique effectuent des prévisions en apprenant à partir des données fournies par le profil des volontaires. Pour ce faire, les algorithmes construisent un modèle basé sur les entrées, puis l'utilisent pour faire des prévisions. Dans cette section, nous présentons l'expérimentation de la prédiction de disponibilité des ressources dans un système de calcul sur machines volontaires en cinq étapes principales : acquisition, extraction, pré-traitement et segmentation des données, construction du modèle de prédiction et enfin la prédiction des ressources de calcul. A l'étape de la construction de modèles de prédiction, trois modèles à évaluer ont été choisis : le classifieur naïf de Bayes, le perceptron multicouche et la méthode de régression de Lasso. Nous commençons donc cette section en présentant l'acquisition, l'extraction, le pré-traitement et la segmentation des données. Ensuite nous présentons la prédiction en utilisant le classifieur naïf bayésien, le perceptron multicouche et la régression Lasso. Enfin nous faisons une comparaison des modèles expérimentés.

4.2.4.1 Acquisition des données

Nous avons utilisé les traces de disponibilité des machines volontaires du projet SETI@home. Ces traces contiennent les informations de disponibilité des nœuds (disponible ou non disponible) dynamiques à grande échelle sur Internet (environ 220000 nœuds pour les traces complètes). Elles sont composées d'une collection d'intervalles de disponibilité/indisponibilité sous la forme $\{t_{début}, t_{fin}, état\}$. De façon plus détaillé, si la valeur de l'état est 1, cela signifie que le nœud était en ligne entre le temps $t_{début}$ et t_{fin}, tandis que si l'état est égal à 0, cela signifie que le nœud était hors ligne [83]. Un extrait de la trace sur une période d'une journée est présenté par la tableau 4.2.

Temps de début	Temps de fin	État
1199132342	1199165424	1
1199165424	1199165712	0
1199165712	1199180458	1
1199180458	1199206512	0
1199206512	1199210588	1
1199210588	1199210730	0
1199210730	1199253658	1

TAB. 4.2 – *Exemple de trace extraite pour une ressource sur une période d'un jour*

4.2.4.2 Extraction des traces et pré-traitement des données

Temps de début	Temps de fin	État
Mardi, 1 janvier 2008 00:00:00	Mardi, 1 janvier 2008 00:59:59	1
Mardi, 1 janvier 2008 01:00:00	Mardi, 1 janvier 2008 01:59:59	1
Mardi, 1 janvier 2008 02:00:00	Mardi, 1 janvier 2008 02:59:59	1
Mardi, 1 janvier 2008 03:00:00	Mardi, 1 janvier 2008 03:59:59	1
Mardi, 1 janvier 2008 04:00:00	Mardi, 1 janvier 2008 04:59:59	1
Mardi, 1 janvier 2008 05:00:00	Mardi, 1 janvier 2008 05:59:59	0
Mardi, 1 janvier 2008 06:00:00	Mardi, 1 janvier 2008 06:59:59	1
Mardi, 1 janvier 2008 07:00:00	Mardi, 1 janvier 2008 07:59:59	1
Mardi, 1 janvier 2008 08:00:00	Mardi, 1 janvier 2008 08:59:59	1
Mardi, 1 janvier 2008 09:00:00	Mardi, 1 janvier 2008 09:59:59	0
Mardi, 1 janvier 2008 10:00:00	Mardi, 1 janvier 2008 10:59:59	0
Mardi, 1 janvier 2008 11:00:00	Mardi, 1 janvier 2008 11:59:59	0
Mardi, 1 janvier 2008 12:00:00	Mardi, 1 janvier 2008 12:59:59	0
Mardi, 1 janvier 2008 13:00:00	Mardi, 1 janvier 2008 13:59:59	0
Mardi, 1 janvier 2008 14:00:00	Mardi, 1 janvier 2008 14:59:59	0
Mardi, 1 janvier 2008 15:00:00	Mardi, 1 janvier 2008 15:59:59	0
Mardi, 1 janvier 2008 16:00:00	Mardi, 1 janvier 2008 16:59:59	0
Mardi, 1 janvier 2008 17:00:00	Mardi, 1 janvier 2008 17:59:59	1
Mardi, 1 janvier 2008 18:00:00	Mardi, 1 janvier 2008 18:59:59	0
Mardi, 1 janvier 2008 19:00:00	Mardi, 1 janvier 2008 19:59:59	1
Mardi, 1 janvier 2008 20:00:00	Mardi, 1 janvier 2008 20:59:59	1
Mardi, 1 janvier 2008 21:00:00	Mardi, 1 janvier 2008 21:59:59	1
Mardi, 1 janvier 2008 22:00:00	Mardi, 1 janvier 2008 22:59:59	1
Mardi, 1 janvier 2008 23:00:00	Mardi, 1 janvier 2008 23:59:59	1

TAB. 4.3 – *Traces découpées en intervalles horaires avec les temps sous une forme lisible*

Pour constituer notre jeu de données d'apprentissage, nous avons extrait les traces de 1000 ressources sélectionnées au hasard dans SETI@home, puis pour chaque ressource, nous avons extrait une sous-trace sur une période de quatre mois allant du 1er janvier 2008 au 1er mai 2008. Le tableau 4.2 montre un exemple de sous-trace d'une ressource extraite sur une période d'une journée. La ressource a changé d'état six fois au cours de cette journée.

Chaque trace est divisée en intervalles horaires, ce qui donne $24 \times 30 \times 4 = 2880$ intervalles. Par la suite, nous représentons la disponibilité de chaque ressource sous la forme d'un vecteur de 2880 bits. Chaque bit représente un intervalle horaire et sa valeur est de 1 si la ressource a été disponible pendant

cette heure et 0 dans le cas contraire. Le tableau 4.3 montre l'exemple d'un découpage en intervalles horaires d'une sous-trace pour une ressource au cours d'une journée. Le temps a été préalablement transformée de la forme *timestamp* à la forme compréhensible par les humains.

4.2.4.3 Segmentation des données

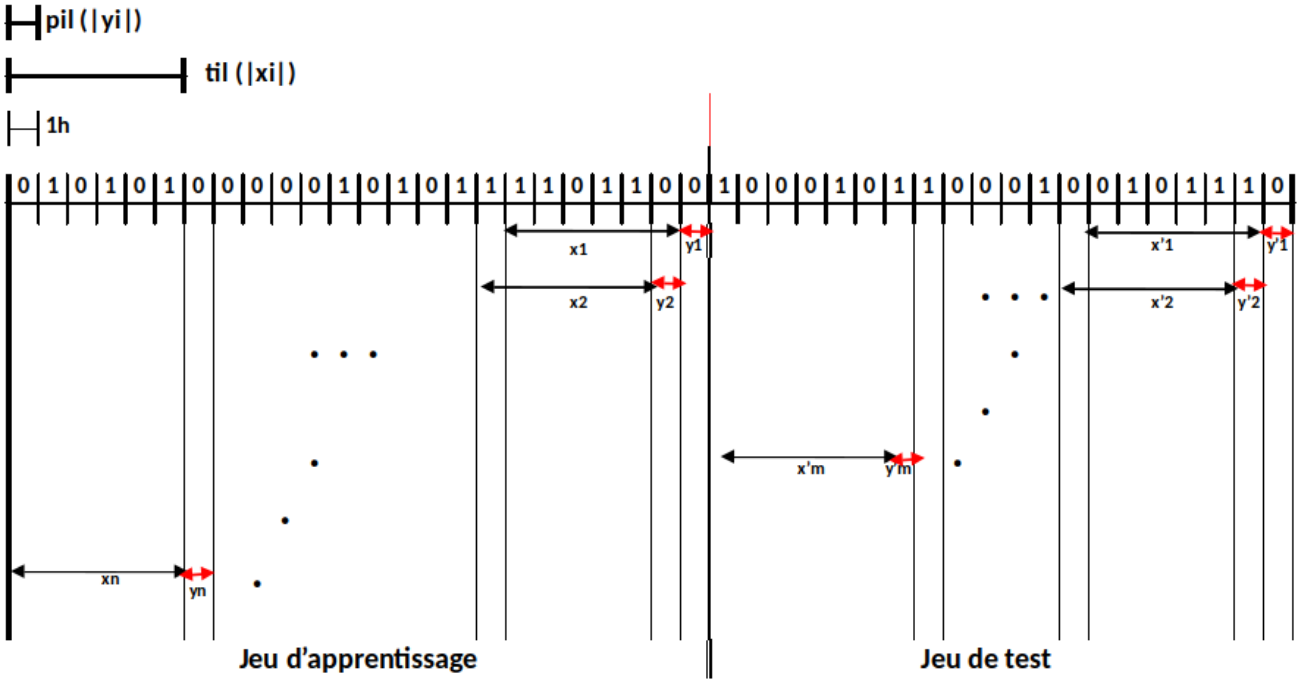

FIG. 4.1 – *Exemple de segmentation de données pour une ressource*

Pour constituer le jeu de données final, nous segmentons chaque chaîne binaire obtenue à l'étape de prétraitement en une série d'intervalles de décalage d'une heure se chevauchant partiellement (figure 4.1). À la fin, l'ensemble de données d'apprentissage de chaque ressource est constitué d'un ensemble de couples $\{(x_i, y_i), 1 \leq i \leq n\}$ où x_i est un segment de taille til, y_i est un segment de taille pil qui suit immédiatement x_i et n la taille de l'échantillon. La figure 4.1 montre une segmentation obtenue pour $til = 6$ *heures* et $pil = 3$ *heures* au cours d'une période d'un jour. On crée ainsi l'ensemble $\{(x_i, y_i), i = 1,2,\ldots,16\}$ où x_i, de taille 6, y_i de taille 3 et $n = 16$ (voir tableau 4.4).

	$til = 6$					
x_1	1	1	0	0	1	0
x_2	1	0	0	1	0	1
...	...					
x_{16}	0	0	1	0	1	0

	$pil = 3$		
y_1	0	1	1
y_2	1	1	1
...	...		
y_{16}	1	1	1

TAB. 4.4 – *Division des données en intervalles pour une ressource*

Les segments y_i de taille pil sont justifiés par le fait que nous voulons prédire la disponibilité des ressources dans l'intervalle $[t, t + pil]$. Par la suite, chaque segment y_i est transformé en 1 s'il ne contenait que des 1 et 0 sinon. En d'autres termes, y_i est 1 si la ressource a été entièrement disponible dans l'intervalle de temps $[t, t + pil]$ et 0 sinon. L'exemple du jeu de données final est présenté dans le tableau 4.5.

x_1	1	1	0	0	1	0
x_2	1	0	0	1	0	1
...	...					
x_{16}	0	0	1	0	1	0

y_1	0
y_2	1
...	...
y_{16}	1

TAB. 4.5 – *Sous-ensemble du jeu de données final pour une ressource*

4.2.4.4 Modèle de prédiction naïve de Bayes

Après avoir transformé les traces de disponibilité brutes en ensembles de tuples (x_i,y_i), nous allons utiliser cet ensemble de tuples pour construire les modèles de prédiction de disponibilité des ressources. Le modèle de prédiction que nous avons choisi d'expérimenter en premier est le classifieur naïf de Bayes car il est facile à mettre en place, permet d'obtenir rapidement les résultats et est le plus utilisé dans la littérature.

L'évaluation a été faite en termes de taux de prédiction, les valeurs du paramètre *pil* ont varié entre 1 et 5 heures et les valeurs de *til* ont été prises par multiples de 10 jours entre 10 et 50 et les multiples de 7 et 30 jours ont été ajoutés pour tenir compte des périodes habituelles du comportement humain, c'est-à-dire hebdomadaire et mensuelle.

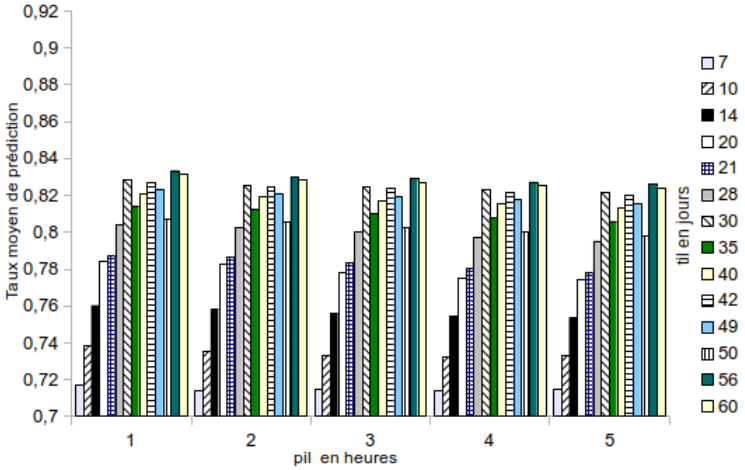

FIG. 4.2 – *Taux de prédiction en utilisant le classifieur naïf de Bayes*

La figure 4.2 résume les performances de prédiction obtenues avec le classifieur naïf de Bayes. L'on peut remarquer que le taux de prédiction croit pour les valeurs de *til* entre 7 et 30 jours. Il décroît à $til = 35$, croît à *til* entre 40 et 42, décroît à *til* entre 49 et 50 et remonte plus tard. Les meilleures performances sont obtenues pour $til = 56$, puis $til = 60$, puis $til = 30$. Ce résultat est cohérent avec le fait que la disponibilité des ressources est alignée sur le calendrier de travail qui a une fréquence mensuelle ou bimensuelle.

Les principaux avantages du classifieur naïf de Bayes : il nécessite relativement peu de données d'entraînement pour estimer les paramètres nécessaires à la classification ; l'apprentissage et l'exécution sont rapides. Notons cependant que lorsque il existe une corrélation élevée entre les caractéristiques, ce classifieur donnera de mauvaises performances car l'hypothèse d'indépendance des caractéristiques n'est pas vraie dans de nombreuses applications, ce qui réduit les performances de prédiction [30].

4.2.4.5 Perceptron multicouche

Nous avons effectué la prédiction de disponibilité en utilisant les mêmes conditions d'expérimentation qu'avec le classifieur naïf de Bayes. La figure 4.3 présente le résultat de l'expérience avec la

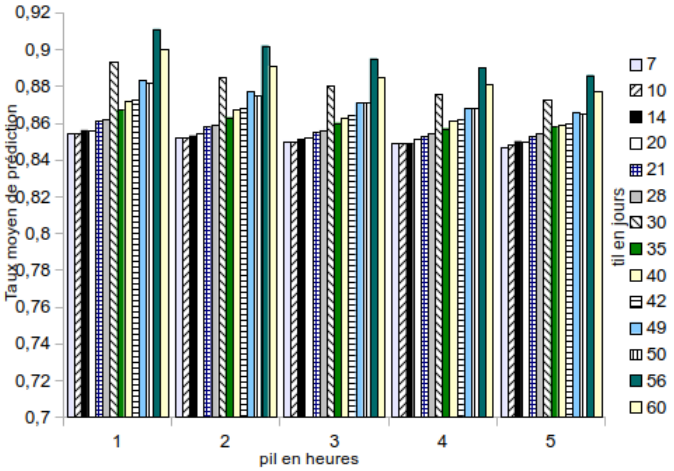

FIG. 4.3 – *Taux de prédiction en utilisant le perceptron multicouche*

méthode du perceptron multicouche. Cette figure nous permet de constater que l'on a les mêmes tendances qu'avec le classifieur naïf de Bayes, mais avec des taux de prédiction plus élevés. Les meilleurs résultats obtenus pour $til = 56$, puis $til = 60$ et $til = 30$. Contrairement au classifieur de Bayes, le temps pour entraîner le modèle avec le perceptron est beaucoup plus élevé.

4.2.4.6 Méthode de régression Lasso

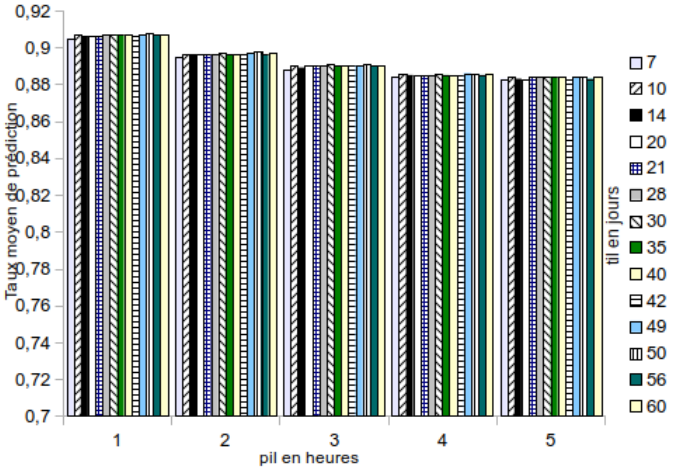

FIG. 4.4 – *Taux de prédiction en utilisant la régression Lasso*

Le modèle de prédiction de disponibilité des ressources est un classifieur binaire, la prédiction obtenue par la méthode de régression Lasso doit être transformée en valeur binaire. Pour ce faire, nous avons déterminé un seuil au-delà duquel la prédiction est fixée à 1 et est maintenue à 0 sinon. Ce seuil n'est pas le même pour chaque ressource, dans les expériences, il est choisi de manière à obtenir à chaque fois le meilleur taux de prédiction.

Le résultat des expérimentations est présenté par la figure 4.4. Nous constatons à travers cette figure que les performances sont homogènes pour une valeur de *pil* donnée. Ces performances diminuent légèrement avec l'augmentation de *pil*. Le taux de prédiction est meilleur que la méthode de Bayes pour toutes les valeurs de *til* et de *pil*. Quelles que soient les valeurs de *pil* et de *til*, les performances de prédiction ne changent pas énormément, contrairement au perceptron dont les performances croissent

au fur et à mesure que l'on a de données. Ce résultat obtenu avec la régression Lasso est dû au fait que lors de la prédiction, Lasso effectue une sélection des variables en éliminant les variables inutiles. Ainsi, la valeur de *til* n'a que peu d'influence sur les performances de la méthode.

4.2.4.7 Comparaison des modèles expérimentées

Le but des expériences étaient d'évaluer les prédicteurs d'abord individuellement, puis comparativement. Les tableaux 4.6, 4.7, 4.8, 4.9 et 4.10 présentent la comparaison des différentes méthodes en fonction des valeurs de *pil* et de *til*. Nous constatons que le classifieur naïf de Bayes a le pire taux de prédiction quelle que soit la valeur de *pil* et de *til*. Les performances de la méthode de régression Lasso ne changent pas beaucoup lorsque *pil* et *til* varient. Avec le perceptron, nous constatons que la quantité de données est alignée avec la qualité de prédiction. Ainsi, plus il y a de données, meilleure est la prédiction. À partir de *til* = 57 jours, le perceptron est meilleur que la méthode de régression Lasso.

En conclusion, la méthode de régression Lasso et la méthode MLP ne sont pas meilleures dans tous les aspects. Ainsi, un bon taux de prédiction ne sera obtenu qu'en combinant ces deux prédicteurs et en mettant en place une méthode de sélection du prédicteur approprié en fonction du profil de la ressource.

Méthode/til	7	10	14	20	21	28	30	35	40	42	49	50	56	60
Bayes	0,717	0,739	0,760	0,784	0,788	0,804	0,828	0,814	0,821	0,827	0,824	0,808	0,833	0,832
Lasso	**0,905**	**0,907**	**0,906**	**0,906**	**0,906**	**0,907**	**0,907**	**0,907**	**0,907**	**0,906**	**0,907**	**0,908**	0,907	**0,907**
MLP	0,854	0,854	0,856	0,856	0,861	0,862	0,893	0,867	0,872	0,873	0,883	0,882	**0,911**	0,900

TAB. 4.6 – *Comparaison du taux de prédiction de disponibilité des ressources pour pil* = 1

Méthode/til	7	10	14	20	21	28	30	35	40	42	49	50	56	60
Bayes	0,714	0,736	0,758	0,782	0,786	0,803	0,826	0,812	0,819	0,825	0,821	0,806	0,830	0,828
Lasso	**0,895**	**0,896**	**0,896**	**0,896**	**0,896**	**0,896**	**0,897**	**0,896**	**0,896**	**0,896**	**0,897**	**0,898**	0,896	**0,897**
MLP	0,852	0,852	0,853	0,854	0,858	0,859	0,885	0,863	0,867	0,868	0,877	0,875	**0,902**	0,891

TAB. 4.7 – *Comparaison du taux de prédiction de disponibilité des ressources pour pil* = 2

Méthode/til	7	10	14	20	21	28	30	35	40	42	49	50	56	60
Bayes	0,715	0,733	0,756	0,778	0,783	0,800	0,825	0,810	0,817	0,824	0,820	0,803	0,829	0,827
Lasso	**0,888**	**0,890**	**0,889**	**0,890**	**0,890**	**0,890**	**0,891**	**0,890**	**0,890**	**0,890**	**0,890**	**0,891**	0,890	**0,890**
MLP	0,850	0,850	0,851	0,852	0,855	0,856	0,880	0,860	0,863	0,864	0,871	0,871	**0,895**	0,885

TAB. 4.8 – *Comparaison du taux de prédiction de disponibilité des ressources pour pil* = 3

Méthode/til	7	10	14	20	21	28	30	35	40	42	49	50	56	60
Bayes	0,714	0,733	0,754	0,776	0,780	0,797	0,823	0,808	0,816	0,822	0,818	0,800	0,827	0,825
Lasso	**0,884**	**0,886**	**0,885**	**0,885**	**0,885**	**0,885**	**0,886**	**0,885**	**0,885**	**0,885**	**0,886**	**0,886**	0,885	**0,886**
MLP	0,849	0,849	0,849	0,851	0,853	0,854	0,876	0,857	0,861	0,862	0,868	0,868	**0,890**	0,881

TAB. 4.9 – *Comparaison du taux de prédiction de disponibilité des ressources pour pil* = 4

4.2.4.8 Travaux récents

Les méthodes de prédiction de disponibilité des ressources de calcul basées sur les méthodes d'apprentissage automatique ont été étudiées dans le contexte du calcul sur le cloud, les grilles et les systèmes de calcul sur machines volontaires [24]. Plusieurs techniques ont été proposées.

Méthode/til	7	10	14	20	21	28	30	35	40	42	49	50	56	60
Bayes	0,715	0,733	0,754	0,774	0,778	0,795	0,821	0,806	0,814	0,820	0,816	0,798	0,826	0,824
Lasso	**0,883**	**0,884**	**0,883**	**0,883**	**0,884**	**0,884**	**0,884**	**0,884**	**0,884**	**0,883**	**0,884**	**0,884**	0,883	**0,884**
MLP	0,847	0,848	0,850	0,850	0,853	0,854	0,873	0,858	0,859	0,860	0,866	0,865	**0,886**	0,877

TAB. 4.10 – *Comparaison du taux de prédiction de disponibilité des ressources pour pil* = 5

Les méthodes d'apprentissage supervisées considèrent deux classes : la classe disponible et la classe non disponible. Leur but est de déterminer pour un intervalle de temps donné, à quelle classe la machine appartient. Les techniques comme le classifieur naïf de Bayes [24, 30], la combinaison de l'indice de Jaccard à l'apprentissage paresseux [91], les K plus proches voisins [91], les séries chronologiques pour l'automatisation de la recherche de corrélation et de la sélection des attributs qui conduisent à une prédiction [79], les machines à vecteur de support sont généralement présentés dans la littérature.

Les méthodes d'apprentissage non supervisées construisent des clusters à partir des données pour former des groupes de modèles de disponibilité en fonction de l'utilisation commune des ressources. Ces informations sont par la suite utilisées par le classificateur, puis des prédictions de la disponibilité future des ressources sont faites en fournissant des données de test au classificateur [24, 79, 85, 92]. La modélisation semi-markovienne multi-états a été utilisée pour estimer la disponibilité d'un nœud de calcul pendant une certaine durée et les algorithmes de clustering hiérarchique ont été également utilisés [24].

4.3 Proposition d'une approche de prédiction de disponibilité des ressources

Les expérimentations présentées dans la section 4.2 nous ont permis de constater qu'aucun des modèles choisis pour la prédiction de disponibilité d'une ressource n'est meilleur que les autres quand on considère tous les différentes valeurs des paramètres *til* et *pil*. Ainsi, un bon taux de prédiction ne sera pas toujours obtenu à partir d'un modèle construit à l'aide d'une seule méthode. C'est pourquoi nous proposons dans cette section une approche de prédiction basée sur une combinaison de modèles et la mise en place d'un système permettant de sélectionner le modèle le plus approprié en fonction des paramètres *til* et *pil*. En nous inspirant des étapes parcourues durant les expérimentations, nous avons défini notre approche en six étapes principales (voir la figure 4.5) : acquisition de données, extraction de traces, pré-traitement des données, segmentation des données, définition de modèles de prédiction et prédiction de disponibilité.

Acquisition des données La phase d'acquisition de données consiste à collecter un ensemble de données sur chaque volontaire. Ces données contiennent une collection d'enregistrement d'intervalle de disponibilité/indisponibilité et d'autres informations telles que l'identifiant, la vitesse du processeur et la taille de la RAM de chaque ressource. Par la suite, ces données sont divisées en données d'entraînement et données de test. Les données d'entraînement permettront d'entraîner nos modèles et les données de test de déterminer pour chaque ressource et les paramètres *pil* et *til*, le modèle le plus approprié pour la prédiction.

Extraction des traces Généralement, les données du système sont stockées dans une base de données centrale. Au cours de la seconde phase, les données de chaque volontaire sont extraites. Pour chaque volontaire, le profil dynamique, déterminant sa période de disponibilité et d'indisponibilité est extrait. Pour ce faire, pour chaque enregistrement, nous considérons uniquement l'intervalle de temps et l'état de la ressource dans cet intervalle de temps. L'intervalle de temps se compose de l'heure

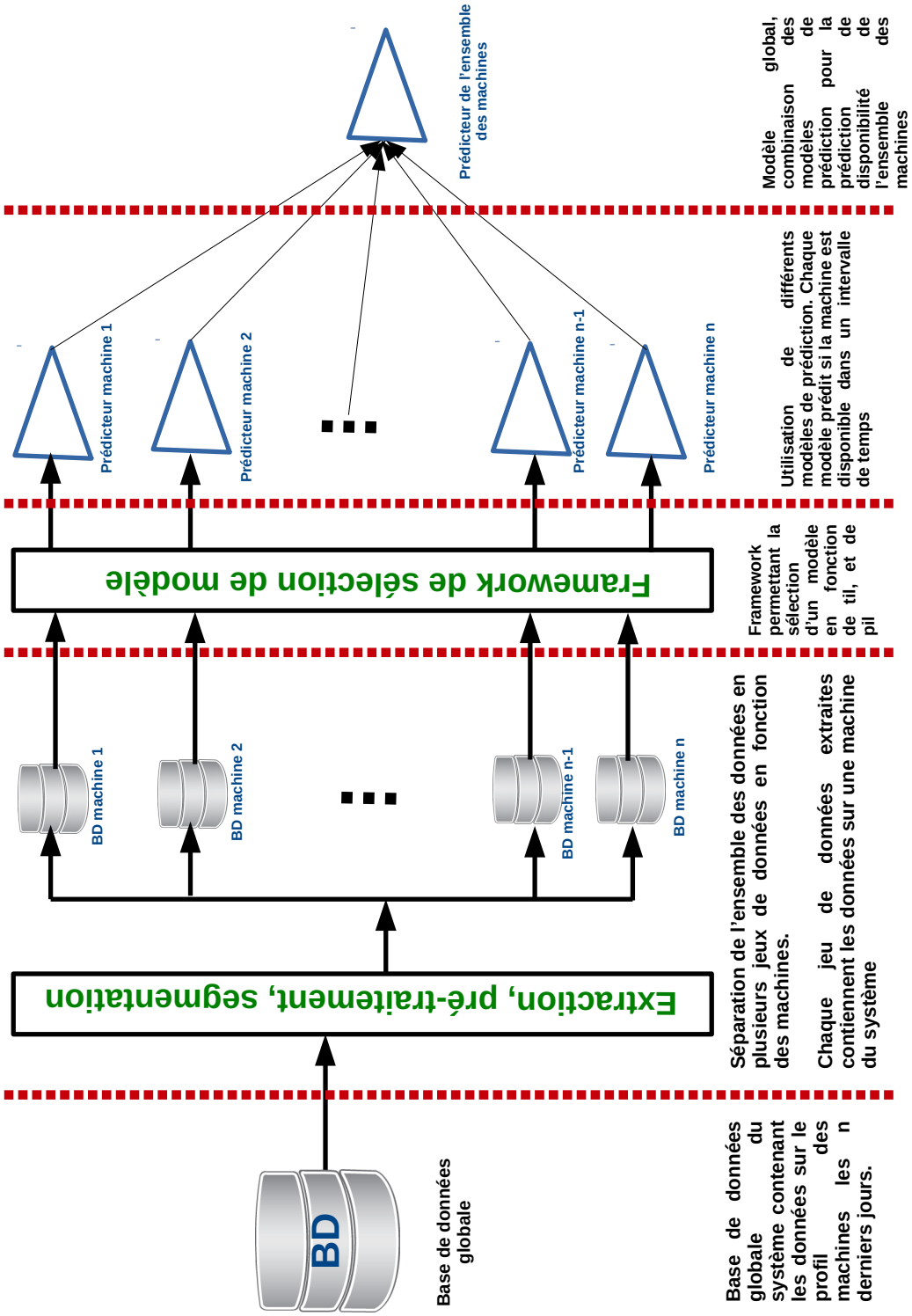

FIG. 4.5 – *Présentation de l'approche de prédiction de disponibilité des ressources de calcul*

de début et de l'heure de fin et l'état de la ressource est un binaire indiquant si la ressource était disponible ou non cet intervalle de temps.

Pré-traitement des données Le traitement du jeu de données implique la conversion de la trace de disponibilité d'une forme brute vers un format approprié pour les rendre beaucoup plus utilisable pour les tâches d'apprentissage automatique. Étant donné que le temps passé dans l'état disponible et indisponible est variable et que les variations peuvent aller de quelques secondes à plusieurs mois dépendant de l'utilisation de la ressource par son propriétaire et les éventuelles pannes, un pré-traitement est donc nécessaire. Ce pré-traitement permettra de transformer ces intervalles en intervalles de taille fixe. Dans un premier temps, les horodatages des intervalles sont transformés sous la forme lisible par l'homme. Par la suite, ces intervalles sont divisés en intervalles horaires. Pour chaque intervalle de temps obtenu, l'état de disponibilité est défini à 1 si la ressource a été disponible à 100% dans cet intervalle et à 0 dans le cas contraire. À la fin de cette phase, chaque ressource est représentée par un vecteur binaire.

Segmentation des données Pour former le jeu de données final, chaque chaîne binaire obtenue à l'étape de prétraitement doit être segmentée en une série d'intervalles de décalage d'une heure se chevauchant partiellement. À la fin, le jeu de données de chaque ressource est constitué d'un ensemble de couples $\{(x_i, y_i), 1 \leq i \leq n\}$ où x_i est un segment de taille *til* et y_i est un segment de taille *pil* qui suit immédiatement x_i et n la taille de l'échantillon.

Modèle de prédiction Nous proposons une approche de prévision combinant plusieurs méthodes d'apprentissage. Pour le définir, soit V l'ensemble des ressources et V_i le volontaire i ; M l'ensemble des modèles entraînés en utilisant les données dont nous avons présentées le traitement dans les paragraphes précédents ; P le vecteur de disponibilité, dont chaque cellule contient 0 lorsque la machine n'est pas disponible et 1 lorsque la machine est disponible ; $P_{i,j}$ la prédiction du volontaire V_i dans l'intervalle $pil = j$; et F la fonction de sélection de modèles. Alors, la formule 4.5 présente le modèle de prévision de disponibilité basé sur la combinaison de plusieurs modèles. La fonction F prend en entrée les données sur un volontaire et l'ensemble des modèles et retourne le modèle qui est le plus approprié pour la prévision de disponibilité. Elle permet de sélectionner le modèle le plus approprié pour la prévision de disponibilité d'un volontaire au fur et à mesure que les données sur celui-ci augmentent.

$$\begin{aligned} V = \{V_i = P_{i,j} | P_{i,j} = 0/1,\ i,j \in \mathbb{N}\} \\ M = \{m_k,\ k \in \mathbb{N}\} \\ F(V_i,M,til) = m_i \end{aligned} \tag{4.5}$$

En considérant les modèles (perceptron et Lasso) expérimentés dans la section 4.2, nous obtenons le modèle de l'équation 4.6.

$$\begin{aligned} V = \{V_i = P_{i,j} | P_{i,j} = 0/1,\ i,j \in \mathbb{N}\} \\ M = \{Lasso,\ MLP\} \\ F(V_i,M,til) = \begin{cases} Lasso & \text{si } til \leq 57 \\ MLP & \text{si } til > 57 \end{cases} \end{aligned} \tag{4.6}$$

En effet, nous avons effectué des expériences (voir section 4.2) en faisant varier les paramètres d'entrée (*til*,*pil*) dans le but de trouver pour chaque paire (*til*,*pil*), la méthode d'apprentissage qui prédit avec la meilleure précision. Nous avons donc constaté que le classifieur naïf de Bayes est la méthode la moins performante quelles que soient les valeurs des paramètres *pil* et *til*. Nous avons

également constaté que la méthode de prédiction de Lasso est très performante quelque soit la taille des données et que la courbe de performance de la méthode basée sur le perceptron devient meilleure que la méthode basée sur Lasso à partir de la valeur de $til = 57$. Le seul inconvénient avec le perceptron est la quantité de données qu'il faut le lui fournir pour l'apprentissage. La fonction de sélection F sélectionnera donc la méthode de Lasso pour chaque ressource ayant une valeur de $til \leq 57$ et le perceptron multicouche pour $til > 57$.

Prédiction de disponibilité La prédiction de disponibilité est la dernière phase de notre approche. Cette phase consiste à utiliser les modèles sélectionnés pour prédire la disponibilité de chaque ressource. La disponibilité de l'ensemble des ressources du système pour une valeur de *pil* donné est obtenue (voire formule 4.7) en combinant la prédiction de disponibilité de chaque ressource du système.

$$P = \sum_{i=1}^{n} P_i \tag{4.7}$$

4.4 Conclusion

Dans ce chapitre, nous avons présenté une solution au problème de la non-disponibilité permanente des ressources de calcul dans un système de calcul sur machines volontaires. Nous nous sommes particulièrement concentré sur les systèmes de prédiction basés sur l'apprentissage automatique. La plupart des solutions présentées dans la littérature sont basées sur la classification et la régression pour prédire la disponibilité d'une ressource ou encore d'un ensemble de ressources sur la base des données de disponibilité passées. Ayant constaté que l'utilisation d'une seule méthode d'apprentissage automatique ne permettait pas d'obtenir une bonne prédiction dans tous les cas, nous avons proposé une approche basée sur la sélection de modèles. Elle consiste à déterminer à partir du jeu de données en entrée, la méthode de prédiction la plus appropriée pour prédire la disponibilité d'une ressource de calcul. L'approche que nous avons proposée sera plus adaptée aux environnements pauvres en ressources car dans ces environnements, en plus de la volatilité des ressources de calcul, l'énergie électrique n'est généralement pas stable. Dans le chapitre 6, nous présentons une approche permettant de construire les systèmes de calcul haute performance basés sur les machines des volontaires et intégrant la prédiction de disponibilité.

Chapitre Cinq

Prédiction de disponibilité dans les systèmes de recommandation

Les sites de commerce électronique proposent divers produits (produits physiques, chansons, vidéos, pages web, etc.) aux utilisateurs en ligne. Le nombre toujours croissant de produits pose un problème de choix pour l'utilisateur. Les systèmes de recommandation répondent à ce problème en filtrant les informations pour ne présenter que les produits les plus susceptibles d'intéresser un utilisateur [93]. On distingue généralement les systèmes de recommandation basés sur le filtrage collaboratif (CF - *Collaborative Filtering*) [94] et ceux basés sur le contenu (CBF - *Content− Based Filtering*) [93, 95]. Différentes approches existent pour le calcul des recommandations. L'approche appelée *top-N* trie les produits pour chaque utilisateur dans l'ordre décroissant des préférences estimées et suggère une liste des N premiers [96, 97].

Les systèmes de recommandation traditionnels sont généralement basés sur des informations relatives aux caractéristiques des produits, d'une part, et aux préférences des utilisateurs, d'autre part. Cependant, dans de nombreux cas (par exemple, le système de recommandation des vêtements), il serait important de prendre en compte des informations contextuelles (heure, lieu, etc.). Il serait inapproprié de recommander des vêtements chauds pendant une période froide. Pour prendre en compte le contexte temporel, des systèmes de recommandation basés sur différentes considérations du temps ont été proposés. Ces nouveaux systèmes utilisent les informations relatives au moment où la recommandation est faite, soit pour faire varier l'importance des données en fonction du contexte temporel, soit pour faire décroître cette importance dans le temps [98, 99, 100]. L'idée de la prédiction de disponibilité des ressources peut être utilisée pour déterminer quels produits peuvent être achetés à une période donnée. L'historique d'achats est analysé et la prévision de disponibilité permet de donner un score représentant les chances qu'un produit soit acheté à un moment donné. Ce score peut être utilisé par la suite, soit pour pénaliser, soit pour renforcer le résultat d'une recommandation. Dans ce chapitre, nous expérimentons dans les systèmes de recommandation Top-N, la méthode utilisée dans le chapitre 4 pour la prévision de disponibilité des machines dans les systèmes de calcul sur machines volontaires. Nous commençons par les calculs statistiques sur la disponibilité des catégories de produits dans un système de recommandation en fonction de divers éléments du contexte temporel dans la section 5.1. Dans la section 5.2, l'expérimentation de la prédiction de disponibilité dans les systèmes de recommandation est effectuée. Et la section 5.3 conclut le chapitre.

5.1 Calculs statistiques sur la disponibilité des catégories en fonction du contexte temporel

Le contexte temporel est l'information qui caractérise le moment où une opération est effectuée dans un système de recommandation [101]. Il s'agit d'un élément essentiel pour la prédiction de disponibilité des catégories. Dans cette section, un ensemble de jeux de données est sélectionné et

ceux pour lesquels l'utilisation du contexte temporel est pertinente sont déterminés. Dans la suite, nous présentons d'abord les jeux de données et les contextes temporels considérés (section 5.1.1), puis les mesures d'homogénéité choisies et les résultats obtenus (section 5.1.2).

5.1.1 Contexte temporel

Cette section fournit une brève description des jeux de données sélectionnés et présente les contextes temporels considérés.

5.1.1.1 Jeux de données

Les jeux de données des systèmes de recommandation contiennent généralement des enregistrements des opérations associant les utilisateurs aux produits. Chaque opération contient l'identifiant de l'utilisateur, l'identifiant du produit, la catégorie ou le genre du produit, un horodatage et éventuellement d'autres informations. Un produit peut appartenir à plus d'une catégorie. Le tableau 5.1 montre un extrait d'un jeu de données d'un système de recommandation.

timestamp	**userId**	**itemId**	**genre**	**rating**
964982703	1	1	Adventure	4.0
964982703	1	1	Animation	4.0
964982703	1	1	Children	4.0
964982703	1	1	Comedy	4.0
964981247	1	3	Comedy	4.0
964981247	1	3	Romance	4.0
964982224	1	6	Action	4.0
964982224	1	6	Crime	4.0
964982224	1	6	Thriller	4.0

TAB. 5.1 – *Extrait du jeu de données MovieLens*

Un ensemble de cinq jeux de données parmi les plus populaires en ligne a été sélectionné et est présenté dans le tableau 5.2.

Nom	**Contenu**	**Catégorie**	**Nb catégorie**	**Nb produits**	**Durée**
Epinions [a]	produits divers	type de produit	27	296 277	4 326 jrs
Last.fm [b]	chansons	auteur de la chanson	83 982	961 416	3 150 jrs
Movielens-2K [c]	films	genre du film	92	10 109	104 167 jrs
Movielens-LS [d]	films	genre du film	26	9 724	8 214 jrs
Ponpare	produits divers	type de produit	41	19 367	359 jrs

TAB. 5.2 – *Jeux de données*

[a] https://www.epinions.com
[b] http://ocelma.net/MusicRecommendationDataset/
[c] https://grouplens.org/datasets/movielens/
[d] https://grouplens.org/datasets/movielens/

5.1.1.2 Définition du contexte temporel

Les contextes temporels que nous considérons sont de deux types : les contextes de base et les contextes composés. Le détail est donné dans le tableau 5.3 ci dessous :

Au total nous avons 16 types de contextes temporels et 212 contextes temporels considérés.

No	Nom du contexte	Nombre	Valeur
0	Moment de la journée	4	'night', 'morning', 'afternoon', 'evening'
1	Jour de la semaine	7	'monday', 'tuesday', ... , 'sunday'
2	Période de la semaine	2	'business', 'weekend'
3	Jour du mois	30	1, ..., 30+
4	Le mois	12	'january', 'february', ... ,'december'
5	Période du mois	3	'beginning', 'mid', 'end'
6	Saisons	4	'winter', 'spring', 'summer', 'autumn'
7	Moment de la journée & Jour de la semaine	28	
8	Moment de la journée & Période de la semaine	8	
9	Moment de la journée & Période du mois	12	
10	Moment de la journée & Saison	16	
11	Période de la semaine & Période du mois	6	
12	Période de la semaine & Mois	24	
13	Période de la semaine & Saison	8	
14	Période mois & Mois	36	
15	Période mois & Saison	12	

TAB. 5.3 – *Les contextes temporels choisis*

5.1.2 Mesure d'homogénéité

Après avoir présenté les jeux de données sélectionnés et les type de contextes temporels considérés, nous présentons les calculs effectués pour déterminer les jeux de données qui sont sensibles aux contextes temporels et les types de contextes auxquels ils sont sensibles.

5.1.2.1 Catégories pour lesquelles un type de contextes est pertinent

Pour déterminer les jeux de données pour lesquels l'utilisation du contexte temporel est bénéfique, nous utilisons des mesures d'homogénéité. Une mesure d'homogénéité détermine l'écart de la distribution d'une variable cible par rapport à la distribution uniforme. Un type de contexte temporel n'est pas utile pour une catégorie si les occurrences de cette catégorie sont uniformément réparties entre les éléments de ce type de contexte. Afin d'estimer l'importance d'un type de contexte temporel *tct* pour la prédiction d'une catégorie *cat*, nous devons nous rassurer que la valeur de la mesure d'homogénéité par rapport à *cat* pour le type de contexte *tct* est minimale. Supposons que *tct* prend ses valeurs dans l'ensemble à m éléments $\{tct_1,tct_2,\ldots,tct_m\}$ et que $f()$ est une fonction qui, étant donné une catégorie *cat* et un contexte temporel tct_i de type *tct*, retourne la probabilité que *cat* soit présent dans le contexte tct_i. Dans la suite, nous utilisons l'entropie, qui est une mesure d'homogénéité très utilisée.

L'Entropie pour la catégorie *cat* en fonction de *tct* est donnée par l'équation 5.1 suivante :

$$H(cat,tct) = -\sum_{i=1}^{m} f(cat,tct_i) \times log_b(f(cat,tct_i)) \tag{5.1}$$

$0 \leq H(cat,tct) \leq log_b(m)$, et $b = 2$ car *cat* est présent ou pas dans tct_i

Les courbes de la figure 5.1 présentent le pourcentage de catégories pour lesquelles le type de contextes temporels sur l'axe des abscisses est pertinent dans jeux de données du tableau 5.4 [Epinions, Last.fm, Movielens-2K, Movielens-LS et Ponpare].

Epinions

– Le type le plus pertinent de contextes temporels est 0 qui correspond à *(moment de la journée)* ;

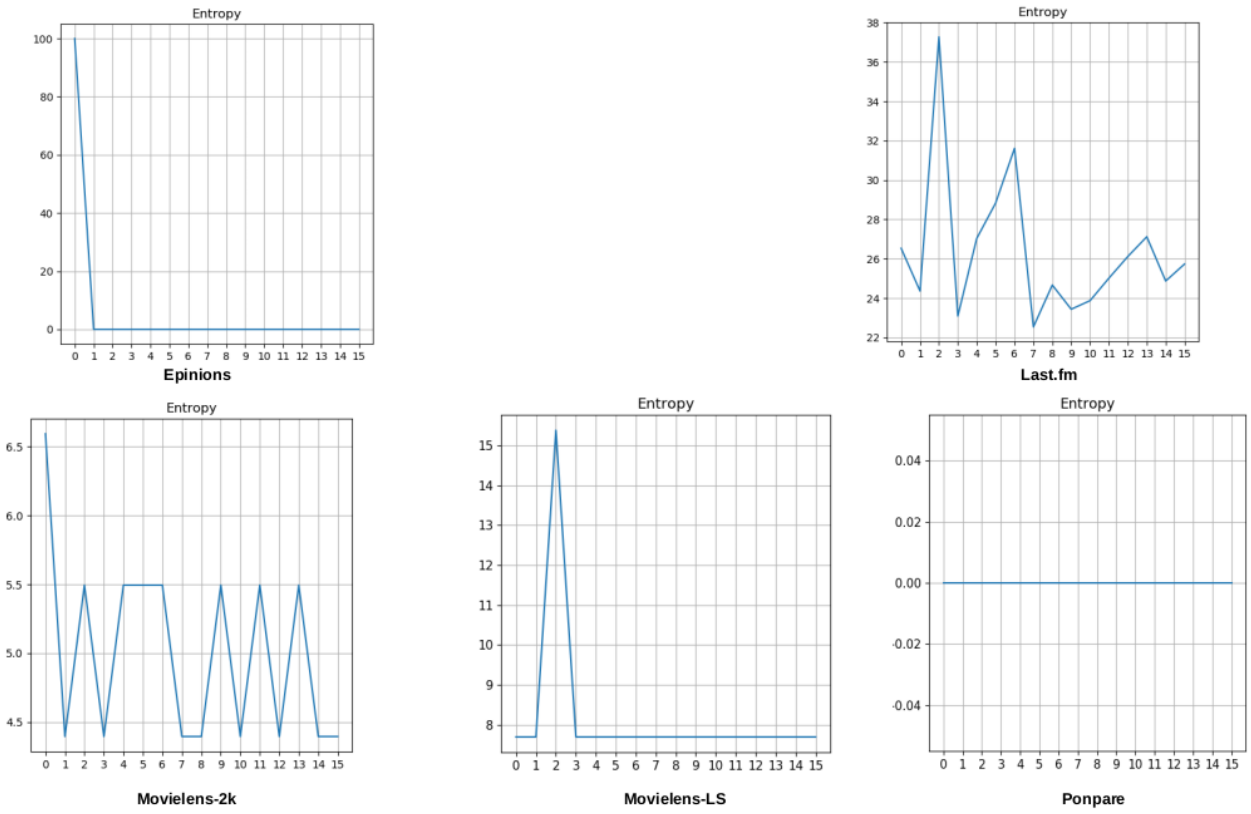

FIG. 5.1 – *Courbes d'homogénéité avec seuil 0,05*

– 100% des catégories sont influencées par le type de contextes temporels 0. Cela peut poser problème si tous les utilisateurs ne sont actifs qu'uniquement pendant une plage de la journée. Il est nécessaire d'observer la répartition des probabilités d'apparition des catégories en fonction de la valeur du contexte temporel.

Last.fm

– La courbe de ce jeu de données est particulière car elle varie beaucoup plus que les autres ;
– Chaque type de contextes temporels influence un pourcentage significatif de catégories (minimum 22% des catégories). Dans une telle situation, il est possible que certains types de contextes influencent exactement les mêmes catégories. D'où la nécessité d'observer les matrices des relations d'intersection des ensembles d'influence des types de contextes temporels ;
– Les 3 types de contextes les plus pertinents par ordre d'importance sont : 2 *(période de la semaine)*, 6 *(saison)* et 5 *(période du mois)*.

Movielens-2k

– Le type le plus pertinent de contextes temporels est 0 *(moment de la journée)*
– Même observation que dans Last.fm, chaque type de contextes temporels influence un pourcentage non négligeable de catégories (minimum de 4% de catégories)

Movielens-LS

– Le type de contextes temporels le plus pertinent est 2 *(période de la semaine)*
– Cependant, nous faisons le même constat que dans Last.fm et Movielens-2k, chaque type de contextes influence un pourcentage non négligeable de catégories (minimum de 7% de catégories)

Ponpare

- Les statistiques effectuées sur ce jeu de données sont presque nulles et inexploitables ;
- Face à de tels résultats, nous pensons que les types de contextes temporels ne sont pas utiles ;
- Ce jeu de données sera abandonné.

En dehors du jeu de données Ponpare pour lequel l'utilisation du contexte temporel n'est pas pertinente, il est nécessaire d'observer la répartition des probabilités d'apparition des catégories en fonction de la valeur du contexte temporel dans les autres jeux de données.

5.1.2.2 Probabilité d'apparition des catégories en fonction des valeurs des types de contextes pertinents

Pour chaque jeu de données, nous nous intéressons aux 3 types de contextes temporels les plus pertinents. Nous présentons les courbes des répartitions des probabilités d'apparition des catégories pour lesquelles le type de contextes est pertinent, en fonction de chacune des valeurs du type de contextes temporels. Lorsqu'un type de contextes temporels est pertinent pour plusieurs catégories, de telles courbes permettent de savoir si ces catégories sont présentes uniquement pour une seule valeur de ce type de contextes ou alors si les apparitions de ces catégories sont uniformément réparties entre les valeurs du type de contextes.

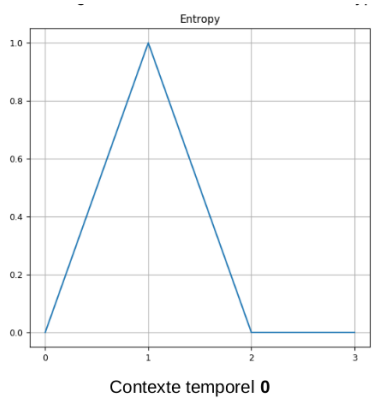

FIG. 5.2 – *Epinions - Probabilité d'apparition des catégories influencées par le type 0*

Epinions

- En observant la figure 5.2 nous constatons que la valeur 1 ("*Morning*") du type de contextes 0 *(moment de la journée)* contient toutes les apparitions des catégories influencées par ce type de contextes. En d'autres termes, tous les utilisateurs ne sont actifs que le matin (de 6h à 12h)
- Cela suggère que la prise en compte du contexte temporel n'est pas nécessaire dans Epinions

Last.fm

- En examinant la figure 5.3, nous constatons que chacune des valeurs de chacun des types de contextes pertinents a une probabilité significative d'apparition des catégories influencées
- Cette remarque laisse penser que la prise en compte du contexte temporel peut être bénéfique dans le jeu de données Last.fm.

Movielens2K

- La figure 5.4 montre que chacune des valeurs de chacun des types de contextes pertinents a une probabilité significative d'apparition des catégories influencées

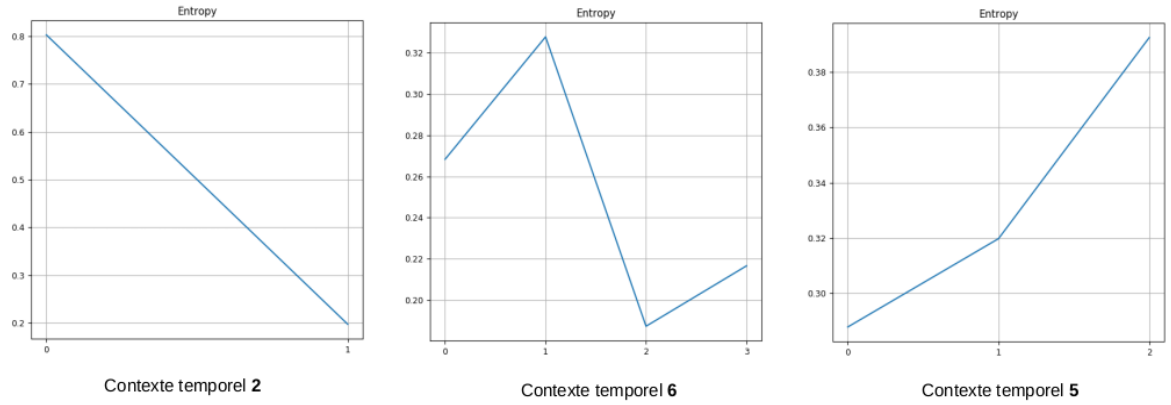

FIG. 5.3 – *Last.fm - Probabilité d'apparition des catégories influencées par le type 2*

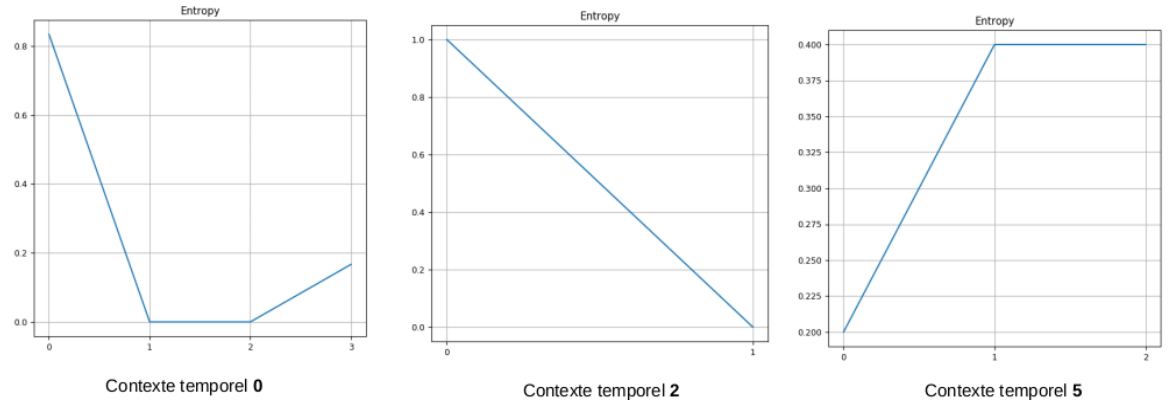

FIG. 5.4 – *Movielens2K - Probabilité d'apparition des catégories influencées par le type 0*

– Par contre dans la figure 5.4, la valeur 0 ("*Business*") du type de contextes 2 *(période de la semaine)* a toutes les apparitions des catégories influencées. Autrement dit, dans Movielens-2K les utilisateurs ne sont actifs que les jours ouvrables et non le week-end.
– Ces remarques signifient que la prise en compte du contexte temporel peut être bénéfique dans le jeu de données Movielens-2k. Et que nous pouvons ignorer le type 2 *(période de la semaine)*

MovielensLS

– La figure 5.5 montre qu'il y a un partage uniforme des apparitions des catégories influencées entre toutes les valeurs possibles du type de contextes le plus pertinent [2 *(période de la semaine)*]
– Cette remarque laisse penser que la prise en compte du contexte temporel peut être bénéfique dans Movielens-ls

5.1.2.3 Dépendance entre les types de contextes pertinents

Dans les cas où nous avons plusieurs types de contextes pertinents, il est possible qu'ils influencent les mêmes catégories ou que l'ensemble des catégories influencées par l'un soit inclus dans l'ensemble des catégories influencées par l'autre.

Supposons que I_i soit l'ensemble des catégories que le type de contextes i influence, la matrice mat_I est construite comme suit :

$$mat_I[i,j] = \frac{card(I_i \cap I_j)}{card(I_i)} \tag{5.2}$$

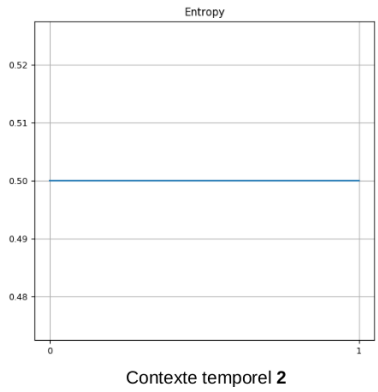

FIG. 5.5 – *MovielensLS - Probabilité d'apparition des catégories influencées par le type 2*

- Ainsi, $mat_I[i,j] == 1$ si toutes les catégories influencées par le types de contextes j sont également influencées par le type de contextes i
- Si $mat_I[i,j] == mat_I[j,i]$ alors i et j influencent le même nombre de catégories et exactement les mêmes catégories
- Si $mat_I[i,j] < 1$ alors il existe des catégories influencées par j qui ne sont pas influencées par i
- $mat_I[i,j] == 0$ si au moins l'un des ensembles I_i et I_j est vide

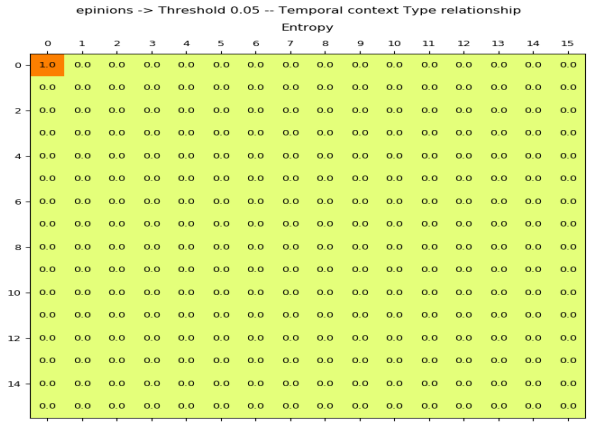

FIG. 5.6 – *Epinions - Relation de dépendance entre les types de contextes*

Epinions

- Dans la figure 5.6, nous constatons que $mat_I[0,0] == 1$ et le reste des cases sont à zéro ;
- Compte tenu des remarques faites pour Epinions dans la sous-section 5.1.2.2 nous pouvons à nouveau affirmer que le contexte temporel n'est pas utile pour ce jeu de données.

Last.fm

- En regardant la figure 5.7, nous constatons que très peu de cases ont une valeur égale 1 ou à 0. Cela signifie que, quel que soit le couple (i,j) de types de contextes, il est très probable que i influence des catégories que j n'influence pas et vice versa
- Ces remarques nous permettent d'affirmer que nous pouvons utiliser tous ou presque tous les types de contextes dans Last.fm

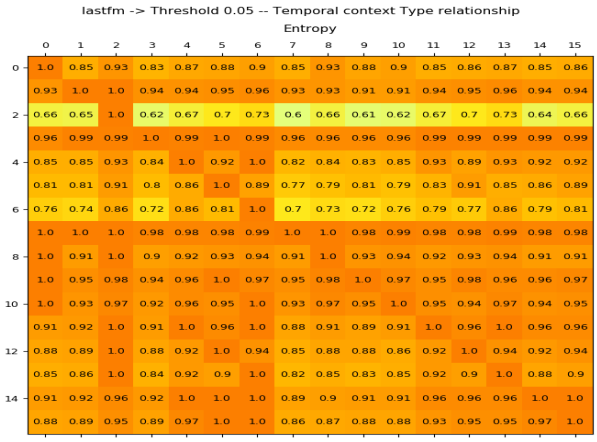

FIG. 5.7 – *Last.fm - Relation de dépendance entre les types de contextes*

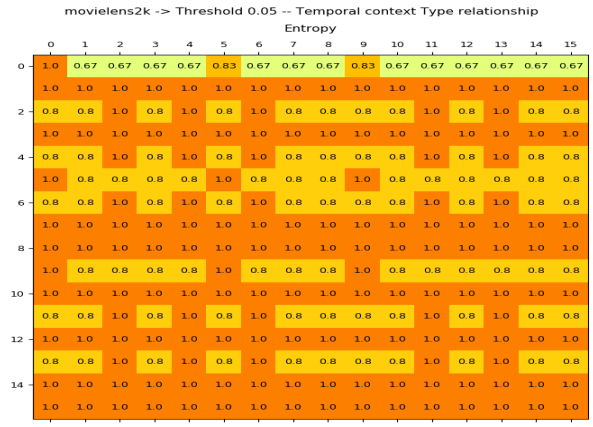

FIG. 5.8 – *Movielens2K - Relation de dépendance entre les types de contextes*

Movielens2K

- La figure 5.8 montre qu'il existe 4 classes des types de contextes qui influencent exactement les mêmes catégories : $classe_1 = \{0\}$, $classe_2 = \{1, 3, 7, 8, 10, 12, 14, 15\}$, $classe_3 = \{2, 4, 6, 11, 13\}$ et $classe_4 = \{5, 9\}$;
- Cela signifie que nous pouvons nous limiter à choisir un type de contextes dans chaque classe, ce qui correspond à 4 types de contextes à choisir ;
- En supposant les types contextes de base sont favorisés, nous choisissons les types de contextes suivants : $\{0, 1, 2, 5\}$

MovielensLS

- La figure 5.9 montre qu'il existe 2 classes de types de contextes temporels : $classe_1 = \{0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ et $classe_2 = \{2\}$;
- Comme dans Movielens2k, nous pouvons nous limiter à choisir un type de contextes dans chaque classe ;
- En supposant que nous privilégions les types de contextes de base, nous choisissons les types de contextes suivants : $\{0, 2\}$

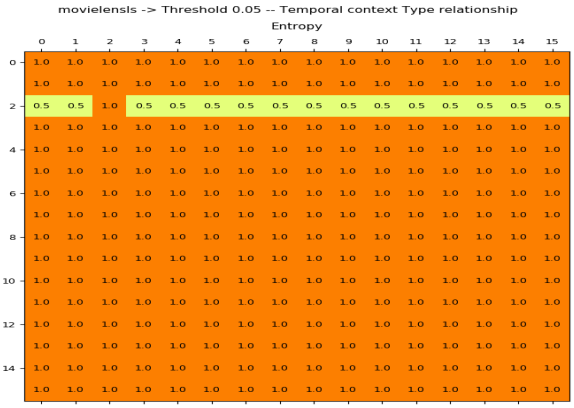

FIG. 5.9 – *MovielensLS - Relation de dépendance entre les types de contextes*

5.1.2.4 Bilan

Au terme de cette analyse, nous pouvons retenir les points suivants :

- Ignorer les jeux de données Epinions et Ponpare car la prise en compte du contexte temporel n'est pas pertinente ;
- Tous les types de contextes peuvent être utilisés dans le jeu de données Last.fm
- Concernant le jeu de données Movielens-2K, nous pouvons nous limiter aux types de contextes temporels de la liste {0, 1, 2, 5}
- Et enfin en ce qui concerne Movielens-LS, nous pouvons nous limiter aux types de contextes suivants : {0, 2}

5.2 Expérimentations et résultats

Dans cette section, nous présentons comment les recommandations sont calculées et évaluées en utilisant la prédiction de disponibilité des catégories de produits sur les jeux de données. Dans la suite, le traitement des données est présenté dans la section 5.2.1, puis l'expérimentation et les résultats dans la section 5.2.2.

5.2.1 Traitement des données

Pour calculer les recommandations en utilisant la prédiction de disponibilité de catégories, nous procédons comme suit :

- Fixer une date initiale t_i et t_f pour le début et la fin d'un jeu de données ;
- Découper en tranches de 6 heures représentant le contexte temporel *moment de la journée* (06h-12h, 12h-18h, 18h-00h, 00h-06h) ;
- Utiliser les 800 premières tranches pour l'entraînement ;
- Utiliser les 400 tranches suivantes pour les tests.

Le tableau 5.4 présente les informations essentielles sur les jeux de données utilisés dans les expérimentations : date de début (t_0) et date de fin, nombre de flots de liens ($L = \{(t_k,u_k,i_k)\}_{k=1..n}$ est le flot de liens extrait où (t_k,u_k,i_k) exprime le fait que l'utilisateur u_k a sélectionné le produit i_k à l'instant t_k), nombre d'utilisateurs et de produits distincts, nombre de catégories des produits et nombre de liens distincts utilisateur-produits ($Nb.(u,i)$) et produit-catégories ($Nb.(i,c)$) respectivement.

	Date début	Date Fin	$\|L\|$	$\|U\|$	$\|I\|$	$\|C\|$	Nb. (u,i)	Nb. (i, c)
Movielens-2k	2008-01-01	2008-12-31	507 280	1 186	7 676	67	136 380	24 280
Movielens-ls	2017-01-01	2017-12-31	23 589	58	3 300	19	8 161	8 706
Last.fm	2008-01-01	2008-12-31	535 073	768	10 804	40	145 223	10 804

TAB. 5.4 – *Statistiques sur les jeux de données extraits*

Pour déduire l'influence de la prédiction de disponibilité des catégories dans les systèmes de recommandation, nous considérons les notations suivantes :

- C est l'ensemble des catégories manipulées ;
- I est l'ensemble des produits avec au moins une catégorie $c \in C$;
- T est l'ensemble des tranches t_k du jeu de test. En d'autre termes, nous avons $|T|$ prédictions de disponibilité pour chaque catégorie $c \in C$;
- M_{ic} est la matrice qui contient les relations d'appartenance des produits aux catégories. $M_{ic}[i,c] = 1$ si l'item i appartient à la catégorie c et 0 sinon ;
- M_{ct} est la matrice qui contient les valeurs des prédictions de disponibilité des catégories. En d'autres termes, $M_{ct}[c,t_k] \in [0,1]$ est la probabilité d'apparition de la catégorie c dans la tranche de temps t_k ;
- M_{it} est la matrice qui contient l'influence des prédictions de disponibilités des catégories pour chacun des produits. $M_{it}[i,t_k]$ est la prédiction de disponibilité de l'item i dans la tranche de temps t_k en considérant les prédictions de disponibilité des catégories.

M_{it} est calculé avec l'équation 5.3, où M_{ic} est de taille $|I| \times |C|$ et M_{ct} de taille $|C| \times |T|$.

$$M_{it} = M_{ic} \cdot M_{ct} \tag{5.3}$$

5.2.1.1 Recommandations combinées aux prédictions de disponibilités

Dans un contexte de recommandation top-N, le système de recommandation propose à chaque utilisateur u une liste des N produits les plus susceptibles de l'intéresser. Pour ce faire, les produits sont classés par ordre décroissant de préférence. Cela suppose que pour toute paire utilisateur-produit (u,i), le système de recommandation est capable de donner une estimation de la préférence de l'utilisateur u pour l'item i. À cet effet, considérons que la matrice $M_{ui}^{t_k}$ contient toutes les estimations pour la tranche de temps t_k, des préférences des utilisateurs pour les différents produits. Autrement dit, $M_{ui}^{t_k}[u,i]$ quantifie la préférence de l'utilisateur u pour l'item i.

Combinaison des préférences des utilisateurs aux prédictions de disponibilités Pour intégrer les prédictions de disponibilités dans les systèmes de recommandation, nous avons considéré un paramètre $\beta \in [0,1]$ pour calibrer l'influence de la prédiction des disponibilités dans les systèmes de recommandation. En supposant que $R_{ui}^{t_k}$ contienne les valeurs finales des probabilités que l'utilisateur u sélectionne le produit i durant la tranche de temps t_k, alors nous utilisons l'équation 5.4 pour calculer ses valeurs :

$$R_{ui}^{t_k}[u,i] = M_{ui}^{t_k}[u,i] \cdot ((1-\beta) + \beta \cdot M_{it}[i,t_k]) \tag{5.4}$$

Ainsi, si $\beta = 0$ alors $R_{ui}^{t_k}[u,i] = M_{ui}^{t_k}[u,i]$. Et si $\beta = 1$ alors $R_{ui}^{t_k}[u,i] = M_{ui}^{t_k}[u,i] \cdot M_{it}[i,t_k]$
Dans la tranche de temps t_k, pour un utilisateur u, le système de recommandation top-N retourne les N premiers produits i classés dans l'ordre décroissant des valeurs de $R_{ui}^{t_k}[u,i]$.

5.2.1.2 Systèmes de recommandation

Pour avoir les combinaisons des systèmes de recommandation concernés, nous avons utilisé 02 aspects, à savoir :

- Le système de recommandation {Nul, Pop, Bip}
- Le mode de prise en compte de la prédiction de disponibilité des catégories {Nul, PCc, PCn, PC1}

En combinant les valeurs de ces deux aspects mentionnés ci-dessus, nous obtenons 12 cas possibles. Puisque le Nul-Nul est invalide, nous manipulons les 11 autres combinaisons. Les détails sur les mots-clés utilisés sont donnés ci-dessous.

Aspect système de recommandation La sémantique des mots clé Nul, Pop et Bip est :

- **Nul :** caractérise l'absence de système de recommandation. Dans ce cas, toutes les estimations des préférences des utilisateurs pour les différents produits sont nulles. Et donc $M_{ui}^{t_k}[u,i] = 0$ quel que soit u, i et t_k.
- **Pop :** correspond à la recommandation des produits les plus populaires que l'utilisateur cible n'a pas encore sélectionnés. Ainsi $M_{ui}^{t_k}[u,i]$ est égal au nombre de fois que l'item i a été sélectionné par les utilisateurs depuis l'instant initial $t0$, jusqu'à l'instant qui précède directement la tranche de temps t_k.
- **Bip :** correspond à la recommandation en exécutant *PageRank personnalisé* sur le graphe bipartite simple construit à partir de toutes les actions de sélection des produits par les utilisateurs. Dans ce cas, $M_{ui}^{t_k}[u,i]$ est égale à la valeur *PageRank personnalisé* du nœud de l'item i, lorsque l'injection des préférences est faite sur le nœud de l'utilisateur u. Cet algorithme de recommandation nécessite l'utilisation d'un paramètre $\alpha \in [0,1]$ qui permet de calibrer le taux de personnalisation. Comme notre travail n'est pas axé sur les performances du PageRank, la valeur de α a été fixée à 0.5.

Aspect prédiction de disponibilité des catégories Pour la prédiction de disponibilité des catégories, les mots-clés Nul, PCc, PCn et PC1 correspondent aux descriptions suivantes :

- **Nul :** les prédictions de disponibilité sont simplement ignorées ($\beta = 0$).
- **PCc :** les prédictions de disponibilité des catégories calculées sont prises en compte sans aucune forme de normalisation.
- **PCn :** les prédictions de disponibilité des catégories calculées avec prise en compte de la normalisation. Dans ce cas, la matrice M_{ic} qui met en relation les produits et les catégories est normalisée pour que $\sum M_{ic}[i,.] = 1$ afin de ne pas favoriser les produits qui sont reliés à plusieurs catégories.
- **PC1 :** correspond à un cas extrême où nous supposons que toutes les catégories sont toujours présentes dans toutes les tranches de temps. Il n'est donc pas nécessaire de recourir à la prédiction de disponibilité. Nous avons donc $M_{ct}[c,t_k] = 1,\forall c,\forall t_k$.

5.2.1.3 Évaluation des recommandations top-N

Pour l'évaluation des recommandations, nous avons utilisé 03 métriques qui estiment différents aspects des systèmes de recommandation à savoir le *Hit-ratio*(HR), le *Mean Average Precision*(MAP) et le *F1-score* [102].

Métriques d'évaluation top-N

- **Hit-Ratio :** proportion d'utilisateurs à qui le système de recommandation a fait au moins une

bonne recommandation. L'équation 5.5 présente le calcul de cette métrique.

$$HitRatio@N = \frac{\sum_{u \in U}(hit_N(u) > 0)}{|U|} \tag{5.5}$$

$hit_N(u)$ retourne le nombre de bonnes recommandation faites à u sur une liste de N produits recommandés.

- **Mean Average Precision :** proportion de produits pertinents en tenant compte de la position des produits parmi les N recommandés. Le calcul de la MAP est donné par l'équation 5.6.

$$MAP@N = \frac{\sum_{u \in U} AP_N(u)}{|U|} \tag{5.6}$$

$$AP_N(u) = \frac{\sum_{k=1}^{N} \frac{hit_k(u)}{k} \times h(k)}{hit_N(u)} \tag{5.7}$$

où $AP_N(u)$ désigne la précision moyenne des recommandations top-N proposées à u et $h(k) = 1$ si le produit à la position k est une bonne recommandation et 0 sinon.

- **F1-Score :** compromis entre la précision et le rappel afin que l'optimisation du score-F1 est plus robuste que l'optimisation de la précision ou du rappel. Pour un utilisateur u, $Precision = \frac{hit_N(u)}{N}$, $Recall = \frac{hit_N(u)}{I_{new}(u)}$
et $F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} = 2 \cdot \frac{hit_N(u)}{I_{new}(u)+N}$. On déduit que le calcul du score-F1 pour l'ensemble des utilisateurs est donné par l'équation 5.8.

$$F1@N = \frac{\sum_{u \in U} 2 \times hit_N(u)}{\sum_{u \in U}(I_{new}(u) + N)} \tag{5.8}$$

Calcul des valeurs finales des évaluations Moyenne pondérée des valeurs des métriques d'évaluation des 400 tranches de temps du jeu de test. Le poids de chaque valeur des métriques considérées est le dénominateur de la formule de la métrique. Par exemple, pour le Hit-Ratio (HR), le poids d'un HR est le nombre d'utilisateurs pour lesquels le système a fait des recommandations dans la tranche de temps concernée.

5.2.2 Résultats

Dans cette section, nous présentons les résultats des expérimentations faites sur les jeux de données Movielens-2k, Movielens-ls et Last.fm. Afin d'estimer les meilleurs résultats pour PCc, PCn et PC1, et sachant qu'il est impossible de tester toutes les valeurs du paramètre $\beta \in [0,1]$, dans les expérimentations β varie dans {0.1,0.25,0.5,0.75,1}. Seules les meilleures performances sont conservées. A cet effet, les 03 sous-tableaux du tableau 5.5 contiennent les résultats obtenus respectivement avec ces jeux de données. Pour chacun des tableaux, les meilleures performances en fonction du type de système de recommandation et de la métrique d'évaluation sont en gras. Chaque valeur est donnée en pourcentage (%). Dans les 03 tableaux, la combinaison de la prédiction des disponibilités aux systèmes de recommandation est représenté par l'ensemble des mots clés {PCc, PCn}. Ainsi, lorsque PCc ou PCn est meilleur cela confirme l'importance de la combinaison proposée. Par ailleurs, dans le premier bloc horizontal, {PCc, PCn} est comparé à PC1, et dans les deux blocs suivants, Pop-{PCc, PCn} et Bip-{PCc, PCn} sont comparés à Pop-{Nul, PC1} et Bip-{Nul, PC1}.

Table 1 (Movielens-2k) Le groupe {PCc, PCn} produit une amélioration 08 fois sur 27. Les pourcentages d'amélioration dans ces 08 cas sont :

- **Amélioration dans le bloc 1 :** Hit-ratio Top-10 (01.445 à 01.822 (26.1%)); MAP Top-10 (00.483 à 00.509 (05.4%)); F1-score Top-10 (00.233 à 00.271 (16.3%)).

Table 1	Hit-Ratio			MAP			F1-Score		
Movielens-2k	@5	@10	@50	@5	@10	@50	@5	@10	@50
Nul-PCc	00.754	**01.822**	06.807	00.367	**00.509**	00.661	00.160	**00.271**	00.361
Nul-PCn	00.042	00.272	01.843	00.027	00.052	00.101	00.013	00.042	00.084
Nul-Pc1	**01.089**	01.445	**09.782**	**00.436**	00.483	**00.682**	**00.257**	00.233	**00.529**
Pop-Nul	08.630	11.772	20.214	04.974	05.201	04.982	02.540	02.715	**02.180**
Pop-PCc	09.238	**11.919**	20.151	05.451	05.603	05.351	02.762	**02.814**	02.157
Pop-PCn	08.693	11.730	**20.235**	05.002	05.245	04.982	02.525	02.717	02.169
Pop-PC1	**09.636**	11.877	19.941	**05.902**	**06.086**	**05.818**	**02.866**	02.767	02.111
Bip-Nul	10.557	**14.181**	25.995	05.879	06.181	05.963	03.370	**03.660**	02.716
Bip-PCc	10.494	14.076	25.974	06.144	06.417	06.083	03.363	03.631	02.673
Bip-PCn	10.515	**14.181**	**26.121**	05.872	06.180	05.931	03.342	03.657	**02.727**
Bip-PC1	**10.725**	14.076	25.744	**06.476**	**06.706**	**06.369**	**03.425**	03.657	02.667

Table 2	Hit-Ratio			MAP			F1-Score		
Movielens-ls	@5	@10	@50	@5	@10	@50	@5	@10	@50
Nul-PCc	00.000	00.000	09.091	00.000	00.000	00.574	00.000	00.000	**00.775**
Nul-PCn	00.000	**01.818**	07.273	00.000	00.227	00.523	00.000	**00.221**	00.387
Nul-Pc1	**01.818**	**01.818**	**10.909**	**00.909**	**00.909**	**00.950**	**00.318**	**00.221**	00.709
Pop-Nul	03.636	07.273	18.182	02.727	03.005	02.489	01.600	02.004	02.404
Pop-PCc	**07.273**	07.273	20.000	**05.455**	05.455	**04.745**	**02.871**	**03.333**	02.532
Pop-PCn	**07.273**	**09.091**	**21.818**	04.101	04.060	03.300	02.560	02.667	02.469
Pop-PC1	05.455	**09.091**	18.182	05.250	**05.532**	03.951	02.560	02.673	**02.666**
Bip-Nul	09.091	14.545	**34.545**	03.697	04.149	04.663	02.233	03.348	**03.251**
Bip-PCc	09.091	14.545	**34.545**	05.818	05.576	05.019	03.200	03.778	03.184
Bip-PCn	09.091	**16.364**	**34.545**	04.313	05.404	04.625	02.233	03.348	03.119
Bip-PC1	**10.909**	14.545	30.909	**07.290**	**07.383**	**06.499**	**03.852**	**04.027**	03.054

Table 3	Hit-Ratio			MAP			F1-Score		
Last.fm	@5	@10	@50	@5	@10	@50	@5	@10	@50
Nul-PCc	00.019	**00.097**	00.595	6.6e-3	**00.017**	00.035	3.7e-3	**00.013**	00.022
Nul-PCn	00.019	**00.097**	00.595	6.6e-3	**00.017**	00.035	3.7e-3	**00.013**	00.022
Nul-Pc1	**00.039**	00.054	**00.676**	**00.014**	00.016	**00.043**	**7.5e-3**	7.1e-3	**00.026**
Pop-Nul	**00.766**	01.217	04.941	**00.408**	**00.460**	**00.581**	**00.223**	**00.255**	**00.272**
Pop-PCc	00.739	**01.259**	**04.976**	00.375	00.434	00.553	00.214	00.253	00.270
Pop-PCn	00.739	**01.259**	**04.976**	00.375	00.434	00.553	00.214	00.253	00.270
Pop-PC1	**00.766**	01.217	04.941	**00.408**	**00.460**	**00.581**	**00.223**	**00.255**	**00.272**
Bip-Nul	01.633	02.565	**08.326**	**00.908**	**00.997**	**01.133**	00.466	00.545	**00.547**
Bip-PCc	**01.664**	**02.604**	08.276	00.896	00.984	01.121	**00.472**	**00.546**	00.541
Bip-PCn	**01.664**	**02.604**	08.276	00.896	00.984	01.121	**00.472**	**00.546**	00.541
Bip-PC1	01.633	02.565	**08.326**	**00.908**	**00.997**	**01.133**	00.466	00.545	**00.547**

TAB. 5.5 – *Résultats pour les jeux de données Movielens-2k, Movielens-ls et Last.fm*

- **Amélioration dans le bloc 2 :** Hit-ratio Top-10 (11.877 à 11.919 (0.35%)); Hit-ratio Top-50 (20.214 à 20.235 (0.01%)); F1-score Top-10 (02.767 à 02.814 (01.7%)).
- **Amélioration dans le bloc 3 :** Hit-ratio Top-50 (25.995 à 26.121 (0.48%)); F1-score Top-50 (02.716 à 02.727 (0.41%)).

Table 2 (Movielens-ls) Le groupe {PCc, PCn} produit une amélioration 08/27 fois. Les pourcentages d'amélioration dans ces 08 cas sont :

- **Amélioration dans le bloc 1 :** F1-score Top-5 (00.709 à 00.775 (9.3%)).
- **Amélioration dans le bloc 2 :** Hit-ratio Top-5 (05.455 à 07.273 (33.3%)); Hit-ratio Top-50 (18.182 à 21.818 (20.0%)); MAP Top-5 (05.250 à 05.455 (3.9%)); MAP Top-50 (03.951 à 04.745 (20.1%)); F1-score Top-5 (02.560 à 02.871 (12.1%)); F1-score Top-10 (02.673 à 03.333 (24.7%)).
- **Amélioration dans le bloc 3 :** Hit-ratio Top-10 (14.545 à 16.364 (12.5%)).

Table 3 (Last.fm) Le groupe {PCc, PCn} produit une amélioration 09/27 fois. Les pourcentages d'amélioration dans ces 09 cas sont :

- **Amélioration dans le bloc 1 :** Hit-ratio Top-10 (00.054 à 00.097 (79.6%)); MAP Top-10 (00.016 à 00.017 (6.3%)); F1-score Top-10 (7e-03 à 0.013 (85.7%)).
- **Amélioration dans le bloc 2 :** Hit-ratio Top-10 (01.217 à 01.259 (3.5%)); Hit-ratio Top-50 (04.941 à 04.976 (0.71%)).
- **Amélioration dans le bloc 3 :** Hit-ratio Top-5 (01.633 à 01.664 (01.9%)); Hit-ratio Top-10 (02.565 à 02.604 (01.5%)); F1-score Top-5 (00.466 à 00.472 (01.3%)); F1-score Top-10 (00.545 à 00.546 (00.18%)).

5.2.2.1 Métriques et Top-N qui favorisent l'amélioration

Le tableau 5.6 illustre la répartition des 25 amélioration du groupe {PCc, PCn} en fonction des métriques et Top-N.

	Hit-ratio	MAP	F1-score	**TOTAL**
Top-5	02	01	02	05
Top-10	06	02	05	**13**
Top-50	04	01	02	07
TOTAL	**12**	04	09	**25**

TAB. 5.6 – *Répartition des cas d'amélioration de performance*

La métrique qui valorise le mieux notre travail est Hit-ratio et le Top-N qui valorise le mieux notre travail est Top-10. Lorsqu'on s'intéresse uniquement à la métrique combinée Hit-ratio Top-10, notre contribution permet des améliorations 06/09 fois et les 03/09 restantes, notre contribution ne fait pas baisser les résultats mais y sont égaux.

5.2.2.2 Meilleures valeurs du paramètre β

Dans cette sous-section, nous nous attardons sur les valeurs du paramètre β pour lesquelles les combinaisons des systèmes de recommandation avec PCc, PCn et PC1 produisent leurs meilleurs performances. Le tableau 5.7, présente ces valeurs, et chaque fois que plusieurs valeurs conduisent au meilleur résultat, seule la plus petite est retenue. Le mot clé *all* est utilisé si toutes les valeurs conduisent exactement à la même performance.

Dans Last.fm, les combinaisons avec PC1 constituent un cas particulier. En effet, quelque soit la valeur de β, pour chacune des métriques on a exactement la même performance. Une telle situation laisse penser que le processus d'intégration de PC1 est inutile.

Le tableau 5.8 présente la répartition des meilleurs valeurs de β en fonction des métriques d'évaluation. La valeur de β la plus récurrente est 0.1, suivi de 0.25, ce qui laisse penser qu'il vaut mieux fixer $\beta < 1/3$ lorsqu'on n'a pas la possibilité d'explorer plus de valeurs.

Si on choisit Hit-ratio Top-10 comme métrique d'évaluation, car cette dernière valorise mieux notre contribution, alors, la meilleure valeur de β est 0.25 dans 09/16 cas si on ignore PC1 dans Last.fm et 11/18 cas sinon.

TAB. 5.7 – *Meilleures valeurs du paramètre β pour les combinaisons avec PCc, PCn et PC1.*

Movielens-2k	H@5	H@10	H@50	M@5	M@10	M@50	F@5	F@10	F@50
Pop-PCc	0.1	0.25	0.1	0.25	0.25	0.25	0.1	0.25	0.1
Pop-PCn	0.1	0.25	0.25	0.25	0.25	0.1	0.1	0.25	0.25
Pop-PC1	0.25	0.25	0.1	0.5	0.5	0.5	0.5	0.1	0.1
Bip-PCc	0.1	0.1	0.1	0.25	0.25	0.25	0.1	0.1	0.1
Bip-PCn	0.1	0.25	0.1	0.1	0.1	0.1	0.1	0.25	0.1
Bip-PC1	0.1	0.1	0.1	0.75	0.75	0.75	0.1	0.1	0.25
Movielens-ls	H@5	H@10	H@50	M@5	M@10	M@50	F@5	F@10	F@50
Pop-PCc	0.5	0.1	0.5	0.75	0.75	0.75	0.5	0.5	0.25
Pop-PCn	0.75	0.25	0.75	0.75	0.75	0.75	0.75	0.75	0.1
Pop-PC1	all	0.75	0.5	0.5	0.75	0.5	0.25	0.1	1.0
Bip-PCc	0.1	0.1	0.1	0.75	0.75	0.25	0.5	0.75	0.25
Bip-PCn	0.1	0.75	0.1	1.0	0.75	0.1	0.1	0.1	0.1
Bip-PC1	0.25	0.1	0.1	0.75	0.75	1.0	0.25	1.0	0.25
Last.fm	H@5	H@10	H@50	M@5	M@10	M@50	F@5	F@10	F@50
Pop-PCc	0.1	0.25	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pop-PCn	0.1	0.25	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pop-PC1	all	all	all	all	all	all	all	all	all
Bip-PCc	0.1	0.25	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Bip-PCn	0.1	0.25	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Bip-PC1	all	all	all	all	all	all	all	all	all

TAB. 5.8 – *Répartition des meilleures valeurs de β en fonction des métriques d'évaluation.*

β	H@5	H@10	H@50	M@5	M@10	M@50	F@5	F@10	F@50	**TOTAL**
0.1	**11**	05	**12**	**05**	05	**07**	**10**	**09**	**10**	**74**
0.25	02	**09**	01	03	03	03	02	03	05	31
0.5	01	00	02	02	01	02	03	01	00	12
0.75	01	02	01	**05**	**07**	03	01	02	00	22
1.0	00	00	00	01	00	01	00	01	1.0	04
all	03	02	02	02	02	02	02	02	02	19

5.2.2.3 Justification de l'allure des résultats de Last.fm

Voici quelques aspects qui justifient l'égalité des lignes (PCc et PCn) et (Nul et PC1) dans la Table 3.

– **Égalité entre PCc et PCn :** Il y a égalité entre PCc et PCn dans Last.fm car dans ce jeu de données particulièrement, on a des chansons et des auteurs. Et dû au grand volume de ce jeu de donnée sur 01 an, nous avons restreint le nombre d'auteurs considérés à 40 (les plus populaires). Cette action a probablement eu pour conséquence qu'il n' y a pas de featuring entre ces auteurs pour les chansons considérées.
Ainsi, chacun des produits de ce jeu de données a une seule catégorie (auteur). Et donc, la normalisation des poids des catégories ne change rien dans Last.fm. Contrairement à Movielens

où un film a facilement plusieurs genres. Par exemple, un film policier peut aussi être un film d'action.

– **Égalité entre Nul et PC1 :** La source de l'égalité entre Nul et PC1 est la même que celle de PCc et PCn. Dans Last.fm chaque produit a une seule catégorie (auteur) et PC1 considère que chacune des catégories est présente tout le temps. Ces deux précédents aspects implique que quelque soit la valeur de β le paramètre qui permet de calibrer l'influence de la prédiction de disponibilité des catégories dans l'équation 5.4, il n'y a pas d'incidence sur le classement des produits. Plus précisément, la même quantité est ajouté à tous les produits et donc le classement de Nul reste le même avec PC1.

5.3 Conclusion

Le but de ce chapitre était d'expérimenter la prise en compte du contexte temporel dans les systèmes de recommandation. Le concept de prévision de disponibilité des ressources dans les systèmes de calcul sur machines volontaires décrit au chapitre 4 a été utilisé pour le calcul des prévisions de disponibilité des catégories de produits dans les systèmes de recommandation.

Les calculs statistiques effectués sur cinq jeux de données sélectionnés nous ont permis d'identifier ceux pour lesquels l'utilisation du contexte temporel est bénéfique. Ces statistiques ont également permis de choisir le type de contextes temporels *moment de la journée* (période de durée 06h à savoir : 06h-12h, 12h-18h, 18h-00h, 00h-06h). Les expérimentations menées sur trois jeux de données des systèmes de recommandation (MovieLens-2k, MovieLens-ls et Last.fm) ont montré que la prédiction de disponibilité des catégories utilisée avec la métrique *Hit-Ratio* combinée à la recommandation *Top-10* et le paramètre de calibrage de l'influence de la prévision des disponibilités $\beta < 1/3$ permet d'améliorer les résultats de la recommandation.

Chapitre Six

Approche de calcul haute performance basée sur les machines volontaires

L'accès aux ressources de calcul est un défi pour les scientifiques, les ingénieurs et les entreprises dans les environnements à ressources limitées. Cependant, dans de nombreux pays, les populations ont accès à Internet, les ordinateurs personnels sont nombreux et appartiennent à des individus. Par exemple, au département d'informatique de l'Université de Yaoundé I au Cameroun, nous avons dénombré un minimum de 1000 ordinateurs appartenant au département, étudiants et enseignants. Habituellement, ces ordinateurs ne sont pas utilisés à plein temps. Par exemple, lorsque les étudiants sont en classe ou lorsqu'ils se reposent, leurs ordinateurs sont inutilisés. Ces ordinateurs peuvent donc être utilisés pour construire une plate-forme de calcul. L'objectif de ce chapitre est la mise en place d'une approche, d'une infrastructure matérielle et logicielle permettant de fournir une puissance de calcul aux scientifiques et ingénieurs des environnements à ressources limitées. Tout d'abord, nous décrivons ce que nous appelons les environnements à ressources limitées (section 6.1). Par la suite, nous présenterons l'approche (section 6.2), l'implémentation de l'approche (section 6.3) et la conclusion du chapitre (section 6.4).

6.1 Environnements à ressources limitées

Aujourd'hui, tous les pays sont confrontés à des problèmes qui nécessitent une grande puissance de calcul. Cependant, certains environnements ne disposent pas d'une puissance de calcul suffisante. Dans la suite, nous commençons par présenter les principales caractéristiques de ces environnements à la section 6.1.1, puis, nous présentons les problèmes de calcul qu'ils rencontrent à la section 6.1.2 et enfin, les solutions généralement utilisées à la section 6.1.3

6.1.1 Description

Les environnements à ressources limitées peuvent être vus sous plusieurs angles : absence d'infrastructures de calcul haute performance, alimentation électrique et Internet instable. Une des principales caractéristiques des environnements à ressources limitées est le manque d'infrastructures de calcul haute performance. Au Cameroun, par exemple, il n'existe pas de système de calcul haute performance. Généralement, les chercheurs et les ingénieurs se rabattent vers leurs ordinateurs personnels. Nous avons collecté les données sur la capacité des machines détenus par les étudiants et les enseignants au sein de l'Université de Yaoundé I (la fig 6.1 présente la fréquence d'horloge et la taille des mémoires de ces machines) et avons constaté que ces machines sont peu performantes pour être utilisées pour du calcul haute performance.

Une autre caractéristique observée dans les environnements à ressource limitées est l'instabilité de l'énergie électrique. Une étude datant de 2018 montre que dans les pays comme le Kenya, l'Inde, le Nigéria et l'Argentine, les coupures d'électricité sont régulières, la capacité de production est limitée

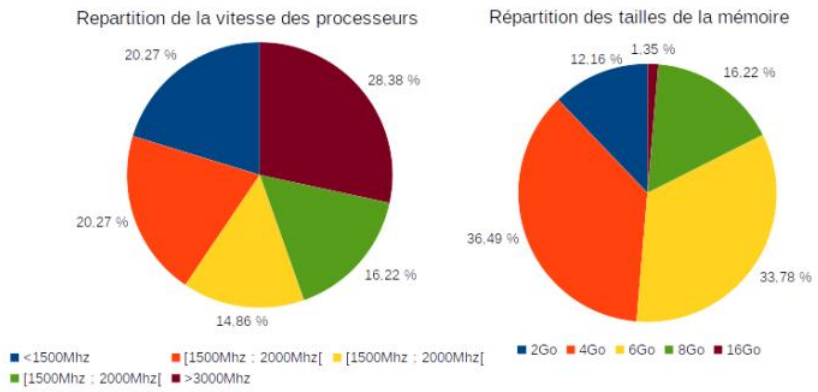

FIG. 6.1 – *Répartition des fréquences de processeur et des tailles de mémoire*

par les caractéristiques géographiques [103]. Au Cameroun, l'approvisionnement en électricité est fréquemment perturbé par des coupures dont la durée totale est en moyenne de 35h/semaine [104]. Un ensemble de données [1] sur la planification des coupures d'électricité dans la ville de Yaoundé collectées sur une période de 10 semaines (du 17/07/2020 au 13/09/2020) nous a permis de constater que toute la ville peut être l'objet de coupures :

- ces coupures étant répartis par quartier, par période ;
- au total, 276 coupures ont été planifiées, leur durée variait entre 59 min et 16 h 10 avec une moyenne de 9 h 41 ;
- la plupart des coupures sont planifiées dans les plages $6h - 16h$, $00h - 06h$ et $23h - 23h59$;
- 91% de coupures sont effectuées en fin de semaine (vendredi, samedi et dimanche) ;
- les coupures ont concerné 132 quartiers, le nombre de coupures par quartier et par semaine variant en entre 1 et 8.

Le tableau 6.1 présente un extrait d'un planning des coupures et le tableau 6.2 présente le planning des coupures par quartier et par semaine dans la ville de Yaoundé. Chaque case contient le nombre de quartiers impactés dans un certain nombre de coupures par semaine. À la semaine 30, par exemple, une coupure est prévue dans 54 quartiers et deux coupures dans 5 quartiers.

Une dernière caractéristique identifiée dans les environnements à ressources limitées est l'instabilité de la connexion Internet. De façon générale, les pannes de l'Internet sont inévitables et fréquentes dans tous les pays [105]. En fonction de leur ampleur, elles peuvent empêcher les utilisateurs d'avoir accès au réseau pendant une courte ou longue période. Selon notre observation au Cameroun, les coupures d'Internet sont fréquentes chez tous les fournisseurs d'accès. Ces coupures peuvent durer d'une minute à une semaine entière. Les raisons souvent évoquées sont les problèmes de maintenance des équipements ou l'accès à la fibre optique.

6.1.2 Les problèmes de calcul

De nombreux pays Africains sont en proie aux catastrophes naturelles comme les inondations, les éruptions volcaniques, les épidémies. Pour apporter une solution à ces problèmes, des modèles mathématiques peuvent être mis en place et les simulations utilisées pour remplacer l'expérimentation. Ces simulations permettrons de réaliser ces phénomènes difficiles voir impossible à appréhender par les expériences classiques. Par exemple, l'analyse des grandes masses de données peut permettre de rendre possible les prévisions météorologiques plus précises, permettant aux pouvoirs publics de mettre en place un système d'anticipation des inondations et des maladies qui en découlent souvent. Notons que la taille de certains problèmes (propagation des épidémies, prévision météorologiques, etc.) ne permet

1. `https://my.eneocameroon.cm/programme-coupure`

Date	Heure début	Heure fin	Localité ou Quartier
01-08-2020	06H00	16H00	SOFAVINC Raccordement d'une nouvelle ligne au réseau
01-08-2020	07H00	16H00	ODZA LIEU DIT PLAQUE L&B Travaux de renforcement de la ligne électrique
01-08-2020	23H50	15H00	MANGUIERS Travaux de maintenance dans un poste
01-08-2020	23H50	16H00	NGONA Travaux de maintenance dans un poste
02-08-2020	06H30	17H00	ODZA Implantation de poteaux bétons
02-08-2020	06H30	17H00	MVAN Implantation de poteaux bétons

TAB. 6.1 – *Extrait d'un calendrier de coupures dans la ville de Yaoundé*

Nb_coupure/Semaine	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**
1	46	54	19	6	2	7	24	12	11	12
2	6	5	10				1			
4									1	
6				1						
7			1	2						
8					1					
Total	58	64	46	34	2	7	26	12	15	12

TAB. 6.2 – *Planification des coupures d'électricité par quartier et par semaine*

pas de les résoudre à l'aide de machines ordinaires. Ces problèmes impliquent des calculs complexes à grande échelle qui, sans système de calcul haute performance, prendraient des jours, des semaines, voire des années à calculer. Dans le cas par exemple de la prévision météorologique pour mettre en place les mesures d'évacuation ou de prise en charge de la population, des résultats tardifs impliquent souvent des résultats inutiles.

Nous avons collecté un ensemble de données au niveau des départements de mathématiques et d'informatique de l'Université de Yaoundé 1 permettant d'identifier les problèmes de calcul haute performance auxquels sont généralement confrontés les chercheurs et les étudiants :

- Au département de mathématiques, un des axes de recherche est l'étude théorique et numérique des problèmes modélisés par les équations aux dérivées partielles ou ordinaires, linéaires ou non. Ces équations proviennent de divers domaines tels que la physique, la biologie, l'épidémiologie et la science de données. Des méthodes de calcul scientifique sont utilisées pour la simulation numérique de ces problèmes. L'une des grandes difficultés pour la résolution de ces équations est l'accès à une ressource de calcul haute performance.
- Au département d'informatique, les chercheurs travaillent dans des domaines variés, certains nécessitant un calcul haute performance et d'autres non. L'analyse des graphes dans les réseaux sociaux [106] et l'apprentissage des modèles neuronaux pour l'extraction de connaissances à partie les grands volumes de données [107] sont des exemples de problèmes qui nécessitent un calcul haute performance.

6.1.3 Solutions généralement utilisées

Les solutions adoptées pour le calcul haute performance dans les environnements à ressources limitées varient en fonction du contexte. Dans de nombreux cas, les chercheurs utilisent leurs ordinateurs pour les simulations ou l'entraînement des modèles d'apprentissage automatique. Cependant, dans de nombreux cas, les calculs peuvent être très longs (une semaine ou plus) et être interrompus avant la fin à cause d'une coupure d'électricité. Selon un enseignant chercheur au département de physique, la simulation de la caractérisation de la résonance stochastique a pris une semaine sur une machine core i5 avec 8 Go de RAM. Avant la fin de la simulation, de nombreuses coupures de courant avaient été enregistrées.

Une autre solution généralement utilisée est celle des grilles de calcul. En effet, la collaboration entre chercheurs des universités du Nord et du Sud permet souvent aux étudiants des universités du Sud d'accéder aux grilles de calcul déployées dans les pays du Nord. Au niveau du département d'informatique par exemple, plusieurs étudiants ont eu accès à la grille française Grid'5000 pendant leur travaux. Notons que l'accès à cette grille nécessite que l'étudiant soit inscrit en co-tutelle ou en co-direction à l'Université du Nord, avec un enseignant jouant le rôle de garant. De plus, les calculs doivent souvent être planifiés à l'avance pour y avoir accès. Cette solution n'est donc pas facilement accessible à tous.

Dans certains pays Africains, les gouvernements ont opté pour l'acquisition d'une infrastructure de calcul haute performance. C'est le cas du Zimbabwe [57] et de l'Afrique du Sud [58] qui ont acquis une grappe de calcul. L'objectif de ces grappes est de leur permettre de réaliser des simulations dans le cadre des projets scientifiques ambitieux. Cependant les solutions adoptées par le Zimbabwe et l'Afrique du Sud sont coûteuses et difficiles à obtenir pour les autres pays.

Une autre solution consiste à payer du temps de calcul sur les grappes ou le cloud. Au niveau du cloud, Amazon vend plusieurs types d'instances de machine qui incluent des nœuds de cluster [2]. Elles se distinguent par des capacités de calcul et de réseau différentes. L'instance la plus haut de gammes, appelée *Cluster Compute Eight Extra Large* est l'instance destinée aux applications de calcul haute performance. Ces systèmes sont beaucoup moins chers que la solution de mise en place d'une infrastructure de calcul haute performance. Cependant, elle est difficilement accessible dans les environnements à ressources limitées à cause des coûts. Par exemple, un projet demandant 100 TFlops avec une grappe de calcul peut coûter jusqu'à 200 000 $ par an. Les instances d'*Amazon Elastic Computing Cloud* fournissent 2 GFlops et coûtent 2,40$ par jour. Pour atteindre 100 TFlops, il faudrait 50 000 instances, ce qui coûterait 43,8 million$ par an . Les expériences de Marathe et al. [56] ont dû être arrêtées sur EC2 d'Amazon car chaque expérience leur coûtait 15$. En plus, les temps de calcul pour mener à bien une expérimentation peut être difficiles à estimer et un budget fixe est quasi impossible à respecter [56].

6.2 Approche de calcul sur machines volontaires

Dans la section 6.1, nous avons présenté les environnements à ressources limitées comme ceux ne disposant pas d'infrastructures de calcul haute performance et dont l'énergie électrique et l'Internet sont instables. Dans cette section, nous proposons une approche de mise en place des systèmes de calcul haute performance basés sur les machines des volontaires et en prenant en compte le contexte des environnements à ressources limitées. Cette approche, résumée par la figure 6.2 est divisée en trois activités principales : l'activité de gestion présentée à la section 6.2.1, l'activité de calcul présentée à la section 6.2.2 et l'activité de support présentée à la section 6.2.3.

2. `https://aws.amazon.com/fr/ec2/pricing/on-demand/`

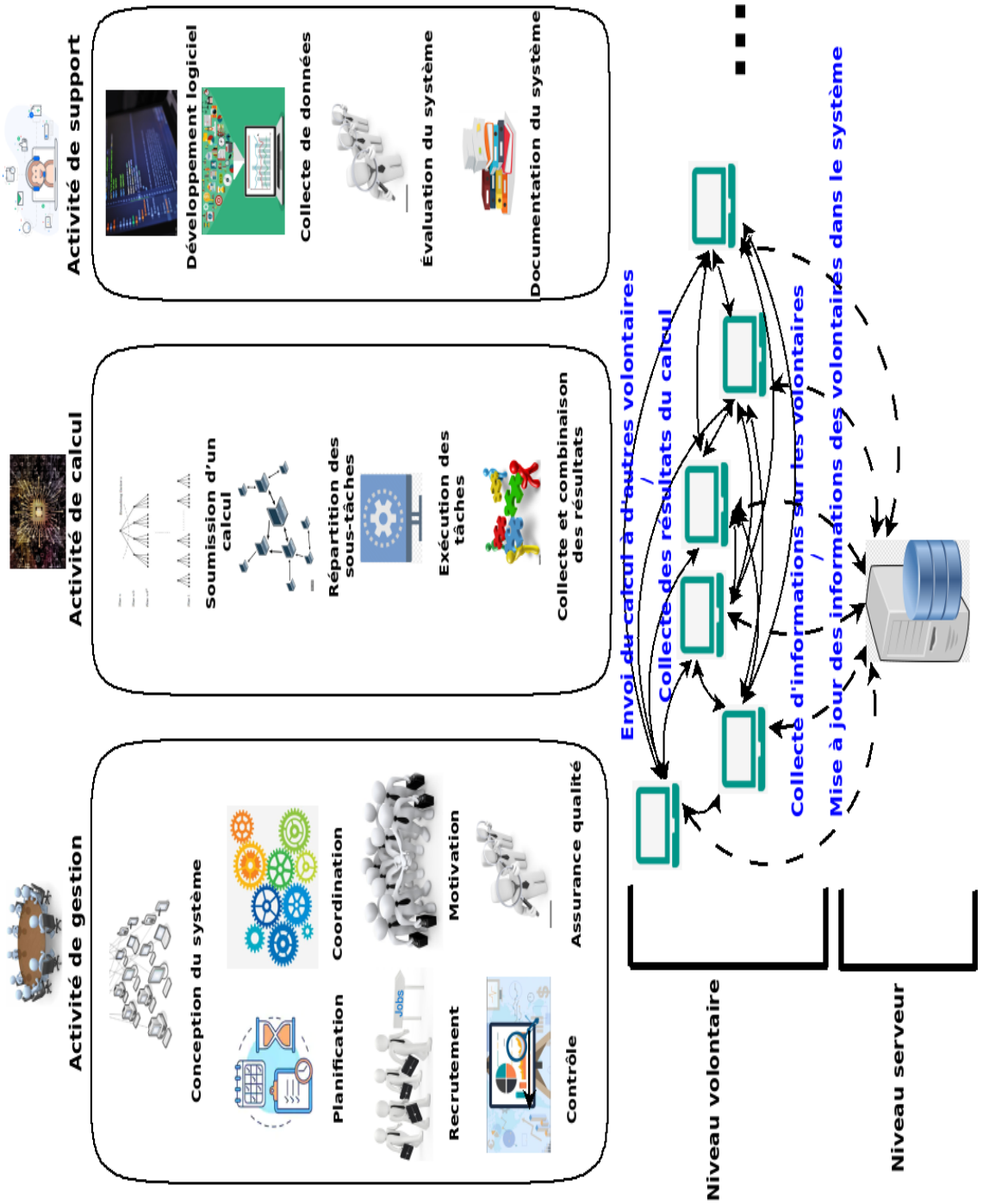

FIG. 6.2 – *Une approche basée sur les machines volontaires pour le calcul haute performance*

6.2.1 Activité de gestion

L'activité de gestion est l'activité principale du système. Pendant cette activité, les informations sur le comportement au quotidien des volontaires (par exemple, la performance du système et de chaque volontaire) sont utilisées pour superviser le système et améliorer ses performances. Les informations sur le profil des volontaires, les types de tâches soumises par les clients, le nom de chaque tâche, la durée d'exécution des tâches, le succès ou l'échec de l'exécution des tâches et les dates d'exécution permettront de considérer le contexte des environnements à ressources limitées. L'activité de gestion est composée des sous-activités suivantes : conception du système, planification, coordination, recrutement des volontaires, motivation, contrôle, et assurance qualité.

6.2.1.1 Conception du système

L'approche que nous proposons est basée sur une architecture hybride dans lequel le serveur joue le rôle de répertoire global d'information et un ensemble de machines jouant les rôles de volontaires et de clients. Le serveur collecte et stocke les informations sur les volontaires (CPU, RAM, disque dur, période de disponibilité), sur le système (puissance globale du système) afin de mettre à jour la base de données locale des volontaires. Les volontaires sont les machines qui vont récupérer les calculs chez les clients et les exécuter. Les clients sont les volontaires qui soumettent les calculs. Leur base de données locale, qui n'est rien d'autre qu'une copie de la base de données du serveur, leur permet de déterminer pour un calcul donné, l'ensemble des machines capables de le résoudre dans un temps raisonnable. L'un des principaux avantages de cette approche est qu'elle ne nécessite aucun investissement supplémentaire en matériel, mais repose sur les machines volontaires déjà détenues par les individus. Il permet donc de garantir l'autonomie et la quasi-gratuité. Notons cependant que le partage de l'ordinateur des volontaires avec leurs propriétaires et la nature instable de l'électricité et d'Internet posent le problème de disponibilité des ressources de calcul. En effet, pour un calcul donné, trouver l'ensemble des machines pouvant l'exécuter dans un temps raisonnable est un défi. Il est nécessaire d'analyser les historiques de disponibilité des différentes machines pour trouver celles qui sont susceptibles d'être disponibles en continu pendant tout l'intervalle de temps estimé pour le calcul en question. Nous avons présenté dans le chapitre 4.3 une solution basée sur l'apprentissage artificiel.

La première et principale activité est la mise en place du système. Dans notre approche, les volontaires sont d'une part des utilisateurs souvent connectés à un réseau, d'autre part les personnes qui ont besoin du calcul haute performance et enfin leurs amis ou les membres de leurs familles. Le serveur a pour rôle de centraliser toutes les informations sur les volontaires. Ces informations sont mises à jour par les volontaires. En cas de panne du serveur (coupure d'électricité ou tout autre panne), l'algorithme d'élection d'un serveur temporaire [108, 109] est utilisé pour choisir le volontaire qui remplacera le serveur en attendant que le problème soit résolu. Dans notre cas, il s'agira de choisir le volontaire qui se connecte le plus à Internet. De manière générale, le système que nous proposons est défini par l'équation 6.1 :

$$\begin{aligned} VC &= [S, (V_1,P_1), ..., (V_i,P_i)], \\ P_i &= [cpu\_speed, num\_core, mem\_size, \\ &\quad disk\_size, os\_name, location, avail]. \end{aligned} \tag{6.1}$$

– S est le serveur. Il est utilisé pour centraliser toutes les informations sur les volontaires afin de les restaurer chaque fois qu'un client en a besoin. Les informations stockées sont les informations sur le système, les performances du système et de chaque volontaire, les logs sur les calculs effectués, le type de tâches déjà réalisé, les temps d'exécution et les tâches en attente d'exécution. Ces informations sont essentielles pour une gestion efficace du système. Le serveur utilise le feedback

sur l'exécution des calculs pour estimer le temps d'exécution d'une tâche, évaluer les volontaires et déterminer les plus fiables.

- V_i un ordinateur volontaire. Il peut jouer deux rôles : le rôle de volontaire ou le rôle de client.
  - Le rôle de volontaire : dans ce cas, il reçoit les tâches, les exécute et renvoie les résultats au client qui a soumis ;
  - Le rôle de client : il synchronise sa base de données locale avec le serveur pour identifier les volontaires pouvant exécuter les tâches, leur envoie le calcul et informe le serveur. Lorsque les volontaires renvoient le résultat, il donne un feedback au serveur.
- P_i le profil du volontaire V_i. Le profil P_i contient les informations sur les éléments qui peuvent être utilisés pour déterminer la capacité de calcul du volontaire V_i. Cette capacité de calcul est obtenue en utilisant la puissance de calcul de l'ordinateur et sa disponibilité. Ces informations sont : la vitesse du processeur (*cpu_speed*), le nombre de cœurs (*num_core*), la taille de la mémoire (*mem_size*), la taille du disque (*disk_size*), le système d'exploitation utilisé (*os_name*), la localisation géographique et la disponibilité (obtenue en analysant le profil d'activité). Ces informations sont utilisées par les clients pour déterminer à quels volontaires envoyer les calculs, les types de calcul à envoyer et les meilleurs moments pour leur envoyer.

Pour fonctionner le système utilise un logiciel que nous avons nommé *vcSoftware* déployé pour des raisons de sécurité dans un conteneur (par exemple docker) et composé de trois modules : le premier nommé *serverManager* est installé au niveau du serveur, le second nommé *volunteerManager* est activé par défaut au niveau des machines volontaires et le troisième nommé *clientManager* est activé par le volontaire pour soumettre un calcul.

serverManager Le module *serverManager* est installé par l'ordinateur désigné comme serveur. Le serveur est un ordinateur dédié, connecté en permanence à un réseau (Intranet/Internet). Il permet de collecter périodiquement les informations sur le profil de chaque volontaire et enregistrer un journal des travaux effectués par le système. Ces informations seront utilisées par la suite pour mettre à jour la base de données des clients, mais surtout, recommander un profil à un volontaire (par exemple, temps de calcul prenant en compte la période d'inactivité de l'ordinateur), de connaître les performances de chaque volontaire et les performances de l'ensemble du système et de faire prédiction de performances. Lorsque le serveur tombe en panne, un serveur secondaire avec une copie de la base de données du serveur prend le relais.

volunteerManager Ce module est utilisé au niveau du volontaire. Pour faire partir du système, chaque volontaire télécharge, installe le logiciel, active le module *volunteerManager* et rempli éventuellement le formulaire fourni pour préciser les périodes de disponibilité de la machine. Ce module est utilisé pour collecter les données de chaque volontaire et envoyer au serveur ; recevoir les tâches des clients, les exécuter et leur renvoyer les résultats. Lors de la configuration du module, le volontaire peut choisir de contribuer uniquement pendant les périodes de disponibilité choisies.

clientManager Lorsqu'un volontaire veut soumettre un calcul, il active le module *clientManager*. Ce module lui permet de soumettre son calcul sous la forme d'un ensemble de tâches qui peuvent être exécutées en parallèle, de déterminer l'ensemble des machines qui peuvent être disponibles jusqu'à la fin du calcul, d'attribuer les tâches aux volontaires, de récupérer les résultats et les journaux des volontaires, de fusionner les résultats pour obtenir le résultat final et d'envoyer le feedback du calcul au serveur.

6.2.1.2 Planification

L'activité de planification est la base de l'activité de gestion. Elle permet d'anticiper et de précéder toutes les autres fonctions de l'activité de gestion et de relever les défis des changements environnementaux tels que l'arrivée et départ de volontaires, panne de serveur et coupure de courant/Internet. Cette activité se situe à deux niveaux : au niveau du serveur et au niveau du volontaire.

Planification au niveau du serveur Pour planifier au niveau du serveur, toutes les informations collectées sur les volontaires pendant l'activité de support (voir section 6.2.3) sont utilisées. Ces informations permettent au serveur de connaître le temps d'inactivité de chaque ordinateur volontaire, de savoir le moment où chacun se connecte à Internet, d'évaluer la performance de chaque volontaire et la performance de l'ensemble du système. Cette planification implique :

- Une évaluation continue des forces et des faiblesses du système : cette évaluation est basée sur les informations sur le profil des volontaires et sur les feedbacks des calculs.
- Identification des actions effectuées pour accomplir les tâches : les informations sur l'exécution des tâches permettrons de déterminer le temps d'exécution et la puissance de calcul nécessaire pour exécuter une tâche dans le but d'améliorer l'exécution future des tâches similaires. Les informations sur une tâche déjà exécutées par le système peuvent être utilisées pour l'exécuter plus efficacement la prochaine fois.

Planification au niveau du volontaire Les informations des volontaires sont stockées sur le serveur et mises à jour de façon fréquente. Elles permettront aux clients de déterminer les volontaires à qui envoyer des tâches en tenant compte de la puissance de calcul offerte, du profil du volontaire et de la prédiction de disponibilité de chaque volontaire. Les principales questions qui vont guider le processus sont : quelle tâche envoyer ? à quel moment la tâche doit-elle être envoyée ? à quel volontaire envoyer la tâche ? à quel moment de la journée envoyer la tâche ? quelle est la durée d'exécution de chaque tâche envoyée ? et quelle est la durée d'exécution de l'ensemble des tâches ? Une bonne planification aidera à relever les défis d'un environnement dynamique.

6.2.1.3 Coordination

Tout comme l'activité de planification, la coordination se fait à deux niveaux : au niveau du serveur et au niveau des volontaires/clients.

Coordination au niveau du serveur Le serveur identifie les volontaires disponibles pour le calcul, la puissance de calcul qu'ils offrent et la durée pendant laquelle ils restent connectés au réseau. Le serveur stocke les informations sur les différentes tâches déjà effectuées. Ces informations seront utilisées pour prédire le temps et les ressources nécessaires pour terminer une tâche. Toute modification des informations sur les volontaires devrait entraîner des changements organisationnels. Les informations sur le recrutement ou la démission des volontaires doivent être prises en compte.

Coordination au niveau des volontaires/clients : A ce niveau, l'activité de coordination consiste à organiser les tâches à réaliser, ainsi que le temps et les ressources nécessaires pour le faire. Les informations obtenues du serveur sont utilisées pour identifier les ressources nécessaires pour réaliser une tâche donnée.

6.2.1.4 Recrutement

Cette activité consiste à recruter les volontaires, à sélectionner en fonction de la puissance de calcul et de la disponibilité le groupe de volontaires pour effectuer un calcul et à évaluer les volontaires

inscrits dans le système. Nous recommandons de commencer par les personnes qui ont besoin de calcul haute performance et qui ne disposent pas de ressources financières suffisantes telles les étudiants et enseignants ; ensuite, nous leur recommandons d'inviter les membres de leur communauté (amis, familles et connaissances) à s'inscrire. Une fois les volontaires dans le système, l'analyse de leurs informations permettra de classer les profils en fonction de la disponibilité et de la puissance de calcul.

6.2.1.5 Motivation

Au cours de cette activité est définit les moyens de motiver les volontaires pour qu'ils s'inscrivent et ceux du système pour qu'ils participent davantage. Les volontaires n'étant pas rémunérés, un système de motivation est nécessaire. Dans notre cas, la possibilité de faire du calcul haute performance à moindre coût est une bonne motivation pour les étudiants en cycle recherche, les enseignants et les chercheurs. Les données collectées sur l'utilisation du système par les volontaires doivent être mises à la disposition du public, principalement les étudiants, les ingénieurs et les chercheurs, afin de les encourager à rejoindre le système. Les données sur les performances du système, les domaines dans lesquels le système a été utilisé, les contributions des volontaires, etc. sont mises à la disposition des volontaires pour les encourager et motiver les membres de leur communauté à participer aux calculs.

6.2.1.6 Activité de contrôle

L'activité de contrôle vise à garantir que les tâches planifiées seront exécutées comme prévu. Au cours de cette activité, l'évaluation de la performance de chaque volontaire et de l'ensemble du système est effectuée afin d'identifier les faiblesses et les forces du système. L'activité de planification est la base de l'activité de contrôle. Elle est basée sur les tâches exécutées et les résultats de ces tâches. Elle joue un rôle important pour garantir l'efficience et l'efficacité du système. Les informations collectées lors de l'utilisation du système permettront de mesurer la performance de chaque volontaire, mais aussi d'identifier ses forces et ses faiblesses. Au niveau du serveur, la comparaison entre les performances planifiées et réelles est analysée pour identifier les écarts et les raisons de ces écarts. Au niveau des volontaires, chaque résultat d'une tâche recueillie auprès des volontaires est utilisé pour l'évaluer. Les informations en temps réel permettront un contrôle rapide, ce qui réduira les coûts des erreurs de planification.

6.2.1.7 Assurance qualité

L'activité d'assurance qualité garantit que la qualité de chaque calcul est satisfaisante, c'est-à-dire que l'exécution a eu lieu pendant un temps raisonnable et a donné des résultats corrects.

6.2.2 Activité de calcul

Les calculs exécutés par le système sont constitués des tâches pouvant s'exécuter en parallèle. Ainsi, chaque tâche est envoyée aux volontaires, qui retournent les résultats et les logs après exécution. Globalement, l'activité de calcul est composée des sous-activités suivantes : soumission du calcul, distribution des tâches aux volontaires, exécution des tâches par les volontaires, retour des résultats et des logs et combinaison des résultats.

6.2.2.1 Soumission d'un calcul

Dans cette étape, un calcul est soumis au système par un client. Soit T un calcul donné, $T = \{t_1,t_2,...,t_n\}$ où n représente le nombre de tâches indépendantes qui composent T. Les t_i étant indépendantes et peuvent être exécutées en parallèle par différents volontaires. Chaque tâche t_i est définie par un ou plusieurs paramètres (fichier(s) ou valeur(s) en entrée), le code de la tâche et produit un résultat (fichier(s) ou valeur(s) en sortie).

6.2.2.2 Répartition des tâches aux volontaires

En tenant compte des données obtenues du serveur sur le profil de chaque volontaire, le module *clientManager* identifiera les volontaires pouvant participer au calcul en faisant une prédiction de disponibilité. Ces volontaires sont les ordinateurs susceptibles d'être disponibles jusqu'à la fin du calcul. Par la suite, les tâches seront envoyé aux volontaires choisis.

Le client qui a démarré un calcul informe le serveur des volontaires impliqués dans le calcul. Le serveur informera à son tour tous les autres clients afin qu'ils puissent prendre compte les autres calculs en cours dans le système.

6.2.2.3 Exécution des tâches

Le module *volunteerManager* sur la machine des volontaires reçoit une tâche, identifie la période de disponibilité spécifiée dans le profil du volontaire et exécute la tâche pendant cette période. Si pendant l'exécution, le système détecte une activité de l'utilisateur de la machine, la tâche est arrêtée et relancée à la prochaine période de disponibilité ou tout simplement suspendue complètement. Si plusieurs tâches ont été attribuées à un volontaire, le *volunteerManager* les exécutera en utilisant l'algorithme premier entré, premier sorti (*First In First Out*). À la fin de l'exécution, un log (heures auxquelles les calculs se sont déroulés, les incidents potentiels, etc.) est enregistré. Notons que *volunteerManager* exécute les tâches en fonction des périodes de disponibilité définies par le propriétaire. Les résultats sont retournés aux clients dès que possible en fonction de la disponibilité du réseau.

6.2.2.4 Collecte des résultats

Une fois les taches terminées, le *clientManager* collecte les résultats et les logs. Si certaines tâches ont échoué, les volontaires les plus efficaces sont identifiés et ces tâches leur sont envoyées. Tous les logs venant des volontaires sont compilés à la fin et renvoyés au serveur.

6.2.2.5 Combinaison des résultats

La dernière étape de l'activité de calcul est la combinaison des résultats obtenus auprès des volontaires par le module *clientManager* pour obtenir la solution finale.

6.2.3 Activité de support

L'activité de support est composée d'une série de sous-activités réalisées en même temps que les activités de gestion et de calcul. Son objectif est de faciliter la gestion et le calcul en fournissant tous les outils et informations nécessaires au système. Le logiciel *vcSoftware* est développé au cours de cette activité. Globalement, cette activité comprend : le développement de logiciels, la collecte de données sur le système et sur chaque ordinateur volontaire, l'évaluation et la documentation du système.

6.2.3.1 Développement logiciel

Au cours de cette activité, le développement ou la mise à jour du logiciel *vcSoftware*, qui sera utilisé par le serveur, les volontaires et les clients est effectué. Cet outil est composé de trois modules principaux :

Le module *serverManager* a pour objectif de collecter et de stocker les données envoyées par les volontaires au serveur. Ces données sont celles fournies par le volontaire lors de son inscription, mais également d'autres informations sur le temps d'inactivité de l'ordinateur du volontaire les heures de connexion sur Internet. Les informations permettront au serveur de définir les profils de chaque volontaire ; faire des suggestions pour la période de disponibilité pour le calcul, sachant que le volontaire peut invalider. Par exemple, pendant les vacances, la période d'inactivité n'est pas nécessairement

la même que pendant les périodes de travail. Les données collectées peuvent permettre au serveur d'identifier ces périodes creuses et demander au concerné de mettre à jour son profil.

Le module *volunteerManager* est le module qui, du côté des volontaires, a pour but de : collecter périodiquement les informations sur le volontaire et les envoyer au serveur ; recevoir les tâches, les ordonnancer en fonction d'autres tâches reçues et les exécuter ; envoyer au client les résultats obtenus après exécution des tâches et les logs contenant les informations comme la performance du volontaire ayant exécuté la tâche, le taux d'échec lors de l'exécution, etc.

Le module *clientManager* va permettre de soumettre les tâches, d'identifier les ordinateurs volontaires capables d'exécuter chaque tâche jusqu'à la fin et d'envoyer les tâches aux volontaires. Il permet également de collecter les résultats et d'envoyer les logs sur l'exécution des tâches et la performance de chaque volontaire au serveur.

6.2.3.2 Collecte de données

L'activité de collecte de données se déroule à la fois au niveau du serveur et des volontaires. Au niveau du serveur, les données collectées sont utilisées pour évaluer la performance du système et celle de chaque volontaire. La performance théorique de l'ensemble du système est calculée en fonction du nombre de volontaires inscrits dans le système. La performance effective est calculée en fonction de la puissance fournie pour exécuter les tâches.

Au niveau du volontaire, les données sont collectées en deux étapes : d'abord, le volontaire remplit le formulaire d'inscription dans lequel il fournit des informations sur son ordinateur et sa disponibilité pour le calcul. Par la suite, il installe le module *volunteerManager* et une collecte automatique se met en place. La collecte automatique peut avoir lieu toutes les heures ou même tous les jours. Les données sont ensuite utilisées pour suggérer des mises à jour de profil aux volontaires et pour évaluer chaque volontaire. Chaque fois que le serveur reçoit les mises à jour d'un volontaire, ces informations sont agrégées avec les informations existantes.

Au niveau du client : les données sont collectées sur l'exécution des calculs et la performance des volontaires.

6.2.3.3 Évaluation du système

L'évaluation du système implique l'évaluation du serveur et de chacun des volontaires. En effet, les données collectées lors des calculs permettront au serveur de savoir si le système est efficace et s'il fournit des résultats corrects. À cet effet, lors de l'exécution de chaque tâche, le nom, le type, l'heure d'exécution de la tâche, la date d'exécution, le nombre de fois où l'exécution de la tâche a échoué et le statut (échoué ou non) sont enregistrés dans un fichier journal et envoyé au serveur.

6.2.3.4 Documentation du système

La documentation est une activité vitale pour le système. En effet, au cours de cette activité, les documents sont produits pour aider le gestionnaire, les volontaires et les clients à mieux utiliser le système. Il fournit toutes les informations sur les caractéristiques du système et de chaque volontaire. Il aide les utilisateurs à comprendre le système et ses principales fonctions. La documentation est mise à jour à chaque fois qu'il y a changement dans le système. Elle peut se voir à deux niveaux : au niveau du gestionnaire de la plate-forme de calcul et au niveau des volontaires et des clients.

Au niveau du gestionnaire, les documents permettront d'installer et de configurer le serveur. Il fournira également des orientations : au serveur sur la façon de motiver les volontaires à s'inscrire, à contribuer et à maximiser les performances du système ; sur la personnalisation des outils pour qu'ils fonctionnent au mieux pour chaque volontaire. Le gestionnaire peut également rédiger les questions fréquemment posées (FAQ) pour aider les volontaires.

Au niveau du volontaire/client, les supports sont utilisés pour informer le volontaire sur le système et décrire ce qu'il est censé faire, comment il fonctionne. Dans l'ensemble, il explique aux volontaires comment installer le logiciel afin de contribuer/utiliser le système et comment soumettre le calcul et collecter les résultats. Pour faciliter l'accès aux volontaires non-informaticiens, des vidéos peuvent être faite sur la façon d'installer le logiciel, soumettre les tâches et collecter les résultats. Un espace libre consultation permet aux volontaires de voir les contributions des uns et des autres.

6.3 VC_UY1 : un système de calcul basé sur les machines volontaires à l'Université de Yaoundé I

Après avoir défini notre approche à la section 6.2, nous avons travaillé à la mise en place d'un système au sein de l'Université de Yaoundé I. Ce système est appelé *VC_UY1* (*Volunteer Computing at the University of Yaounde I*). Le travail effectué pour sa mise en place a été essentiellement consacré au recrutement des volontaires et au développement de l'outil. Dans cette section, nous présentons les différentes activités qui nous ont permis de mettre en place ce système. Ainsi, la section 6.3.1 présente le recrutement des volontaires, les sections 6.3.2 et 6.3.3 présentent les spécifications et la conception de la plate-forme de calcul *vcSoftware*. Nous terminons avec une brève présentation de *VC_UY1* à la section 6.3.4.

6.3.1 Recrutement des volontaires

Pour recruter les différents volontaires, nous avons conçu un formulaire à l'aide de l'outil en ligne *Google Form*[3]. *Google Form* est un outil développé par Google qui permet de réaliser des questionnaires et de mener des enquêtes. Le formulaire que nous avons conçu pour le recrutement des volontaires a été organisé en trois parties principales :

- Partie informative : la partie informative du formulaire contient une brève description du projet, les valeurs du système à mettre en place, son objectif social, le type de contribution que le bénévole peut apporter et le fait que la contribution est sans compensation financière ;
- Partie enregistrement des informations : dans cette partie, les volontaires enregistrent leurs informations personnelles telles que le nom, filière, niveau, contact téléphonique, adresse e-mail et les caractéristiques de leurs machines telles que la vitesse du processeur, la taille de la RAM et du disque dur. Notons que seule l'enregistrement de l'adresse émail du volontaire était obligatoire. En effet, beaucoup de volontaires peuvent ne pas avoir les compétences nécessaires pour donner les caractéristiques de leurs machines ;
- Informations sur la disponibilité du volontaire : contrairement aux approches existantes présentées au chapitre 4 dans lesquelles le volontaire n'a pas besoin d'indiquer sa disponibilité lors de son enrôlement, dans notre approche, lors de son inscription, le volontaire peut donner les périodes pendant lesquelles il estime que sa machine est libre et peut être utilisée pour le calcul.

Une fois mis en place, le recrutement des volontaires s'est déroulé en deux étapes principales.

Étape 1 : diffusion du formulaire Au cours de la première phase, le formulaire a été diffusé aux enseignants des départements de mathématiques et d'informatique et aux étudiants du département d'informatique en utilisant les groupes de diffusion Yahoo, Gmail et les groupes Whatsapp. Cette phase a débuté le 20 juillet 2020 et s'est achevée le 20 août 2020. Un total de 60 volontaires se sont inscrits dont 5% d'enseignants et 95% d'étudiants. Le tableau 6.3 présente le nombre d'inscrits par cycle (Licence, Master et Doctorat), par filière (Informatique, Physique et Autres).

3. https://docs.google.com/forms

Filière/Cycle	L	M	D	Total
Informatique	10	45	2	57
Physique	1	0	0	1
Autre	0	0	2	2
Total	11	45	4	**60**

TAB. 6.3 – *Nombre d'inscrits dans la première phase de recrutement des volontaires*

Étape 2 : porte à porte Ayant constaté que les personnes contactées pour être volontaires (en particulier les enseignants sur lesquels nous comptions le plus) ne s'étaient pas inscrites, nous les avons approchés pour les inviter à participer au système. Lors de la deuxième phase qui a duré 8 jours (du 01 septembre 2020 au 08 septembre 2020), nous avons utilisé une approche de contact pour encourager les volontaires à s'inscrire.

- Invitation des enseignants : nous avons rencontré des enseignants des départements d'informatique et de mathématiques pour leur expliquer la pertinence du projet et comprendre pourquoi ils ne s'étaient pas inscrits. La majorité d'entre eux ont évoqué le manque de temps comme raison, d'autres ont dit qu'ils ne comprenaient pas le bien fondé du projet. Une fois les mérites du système ré-expliqués, la grande majorité a procédé à l'enregistrement de leur machine. Pour ceux qui n'avaient pas le temps de remplir le formulaire, nous avons enregistré leurs machines pour eux.
- Invitation des étudiants : nous avons contacté et rencontré les délégués des étudiants, qui ont relancés leurs camarades. Nous avons également rencontré personnellement les étudiants des cycles Master et recherche pour mieux leur expliquer le projet. Une fois le projet bien compris, beaucoup n'ont pas hésité à s'inscrire.
- Rencontre avec le chef de département d'informatique et du directeur du CUTI (Centre Universitaire des Technologies de l'Information) : pour prendre en compte les machines des salles de TPs et les machines du CUTI, les responsables de ces deux structures ont été rencontrés. Une fois le projet bien compris, ils ont mis à notre disposition les machines de leurs services et les machines des salles de TPs du département.

Au terme de la deuxième phase, il y a eu 104 nouveaux volontaires inscrits dans le système dont 60 (57,70%) étudiants, 16 (15,38%) enseignants, 20 (19,23%) salle de TP et 8 (7,69%) CUTI. Le tableau 6.4 présente les détails.

	Étudiant			Enseignant	Salle TP	CUTI	Autres	Total
Filère/Cycle	L	M	D					
Informatique	23	19	4	10				
Mathématique	0	1	0	5				
Physique	0	0	3	1				
Total	23	20	7	16	20	8	10	**104**

TAB. 6.4 – *Nombre d'inscrits dans la deuxième phase de recrutement des volontaires*

Bilan Au terme du recrutement des volontaires, nous pouvons dire que l'intérêt pour la recherche peut motiver les étudiants et les enseignants à offrir du temps libre de leur machine pour mettre en place un système de calcul haute performance. En effet, la principale raison pour laquelle les utilisateurs nous ont dit s'être enregistrés était de pouvoir permettre à l'Université de disposer d'un système de calcul haute performance. Certains enseignants ont dit être prêt à relancer certaines recherches abandonnées

à cause des besoins de calculs dont ils avaient (par exemple, la simulation de résonance stochastique sur plusieurs dimensions). Beaucoup veulent savoir ce que l'on peut obtenir avec un tel système.

De façon globale, nous avons enregistré 164 volontaires dont 117 (71,34%) étudiants, 19 (11,58%) enseignants, 20 (12,19%) salles de TP et 8 (4,87%) CUTI. Les machines des enseignants, des salles de TP et du CUTI (28,65%) étaient presque toujours disponibles en soirée, entre 18h-06h.

Nous avons constaté que la sensibilisation personne à personne était plus efficace pour le recrutement des volontaires. Notons cependant que c'était la première fois que ce travail était réalisé au sein de l'Université. Notons aussi que les étudiants posaient le problème du support de l'Internet, étant donné beaucoup n'ont pas les moyens de pouvoir s'offrir la connexion.

6.3.2 Spécifications de la plate-forme de calcul vcSoftware

La seconde activité effectuée lors de la mise en place de *VC_UY1* est la spécification de la plate-forme de calcul *vcSoftware*. En s'inspirant des systèmes existants comme Condor [110], I-Cluster [67] et SETI@home [111] et en tenant compte des spécificités des environnements pauvres en ressources présentées à la section 6.1, de l'approche de mise en place des systèmes de calcul basées sur les machines volontaires dans les environnements pauvres en ressources présentée à la section 6.2, nous avons déterminé les objectifs de *vcSoftware*, identifié les acteurs, les cas d'utilisation et les principales contraintes du système.

Objectif de l'application La finalité de l'application *vcSoftware* est de permettre aux clients de pouvoir soumettre et exécuter les applications parallèles, en prédisant et en exploitant les périodes d'inactivités des machines volontaires enrôlées. Pour atteindre cet objectif :

- l'application utilisée par le nœud serveur doit permettre de collecter les données sur les volontaires, les clients, les tâches exécutées et leurs performances, de compiler ces données pour fournir toutes les informations nécessaires sur le système ;
- l'application au niveau du client doit lui permettre de soumettre un calcul en tâches, déterminer les machines volontaires pouvant exécuter les tâches jusqu'à la fin, distribuer les tâches aux volontaires, collecter les résultats, les fusionner pour obtenir la solution finale au problème et faire un feedback de l'exécution du calcul et la contribution des volontaires au serveur ;
- et l'application au niveau du volontaire doit lui permettre de collecter les données sur son profil, recevoir les tâches, les exécuter sans perturber l'utilisateur de la machine, renvoyer les résultats et les logs sur l'exécution au client.

Acteur	**Description**
Serveur	Est la machine qui centralise toutes les informations sur le système. Ces informations sont fournies aux volontaires et aux clients quand ils en ont besoin.
Volontaire	Est le nœud de calcul. Son rôle principal est d'exécuter les tâches qui lui ont été allouées. Pour ce faire, les données sur les performances et la disponibilité de ces machines sont collectées et stockées sur le serveur. Chaque volontaire va garder un résumé des informations qui lui sont nécessaires pour fonctionner.
Client	Le client est un volontaire qui possède des tâches parallèles à exécuter. Il va utiliser les données en sa disposition pour identifier pour un calcul donné, l'ensemble des volontaires susceptibles de l'exécuter jusqu'à la fin.

TAB. 6.5 – *Tableau présentant l'ensemble des acteurs de vcSoftware*

Acteurs et cas d'utilisation Les acteurs sont les utilisateurs de la plate-forme. Ils utilisent l'application et exécutent un ou plusieurs cas d'utilisation. Le tableau 6.5 présente les principaux acteurs, la figure 6.3 présente le diagramme des cas d'utilisation, contenant la liste des cas d'utilisation avec

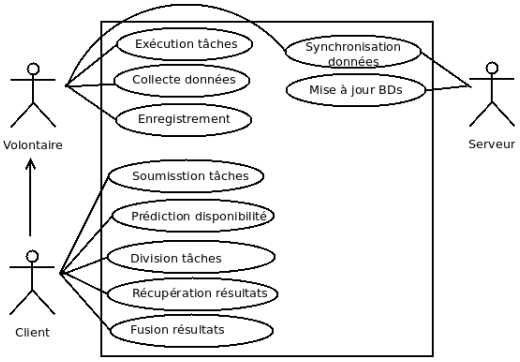

FIG. 6.3 – *Diagramme de cas d'utilisation de vcSoftware*

la précision de l'acteur qui exécute chaque cas d'utilisation. Enfin, le tableau 6.6 présente la liste des cas d'utilisation du système et leur brèves descriptions.

Comme l'utilisateur partage sa machine, un certain nombre de contraintes doivent être vérifiées pour le bon fonctionnement du système :

- Les applications invitées ne doivent pas perturber l'utilisateur : le système doit pouvoir détecter le retour de l'utilisation pendant l'exécution d'une tâche et arrêter immédiatement l'exécution pour ne pas perturber celui-ci; l'utilisateur doit pouvoir trouver sa machine dans le même état dans lequel il l'a laissé ; l'application invitée doit être invisible à l'utilisateur. Pour ce faire, il faut assurer une bonne isolation entre le mode utilisateur et le mode calcul ;
- Déploiement automatique : pour faciliter l'intégration de nouveaux volontaires, le déploiement de l'application doit être le plus évident possible. Pour garder le système hautement autonome, l'installation doit être réduite à sa plus simple expression : exécution du programme, qui procède à l'installation de tous les composants. Une fois installé, l'utilisateur peut intégrer des informations supplémentaires comme son contact et sa disponibilité ;
- Tenir compte des environnements à ressources limités : lors de la prédiction de disponibilité des machines volontaires, le système doit tenir compte de la disponibilité de l'énergie électrique pendant la période de disponibilité du volontaire ;
- Sécurité : les données de l'utilisateur, ses applications et sessions doivent être protégées contre toute faute ou malveillance du système de calcul; les applications de calcul doivent aussi être protégées contre les intrusions des utilisateurs ;
- Un système permettant de gel/reprise d'une tâche en cours d'exécution (*checkpointing/restart*) doit être mis en place pour mettre l'application en pause pour la relancer plus tard en cas de retour prématuré de l'utilisateur ou d'une panne.

Cas d'utilisation	Description
Enregistrement	Le système doit permettre à tout utilisateur de la plate-forme de s'enregistrer en fournissant les informations comme son nom, son lieu de résidence, son téléphone, son adresse émail. Les informations comme le nom, le numéro de téléphone ne sont pas obligatoires. Par contre, le lieu de résidence doit être connu pour déterminer les périodes pendant lesquels il n'y a pas par exemple d'électricité. Il peut aussi fournir, si possible, les informations sur les caractéristiques de sa machine. Contrairement aux systèmes existants, l'utilisateur lors de son enregistrement peut indiquer les périodes de la semaine pendant lesquelles sa machine sera disponible.
Collecte de données	Le système doit fournir un mécanisme permettant d'enregistrer toutes les activités de la machine dans les intervalles précises. Les informations collectées sont : la date et heure de l'enregistrement, la charge du processeur, la charge de la mémoire, la dernière activité du clavier et de la souris, la présence ou l'absence du courant électrique et de connexion au réseau, intervalle de connexion à Internet, intervalle de disponibilité, intervalle d'indisponibilité. Les informations sur l'exécution des calculs : informations sur date et heure de l'enregistrement, le déroulement de l'exécution de la tâche (les échecs, etc.), la puissance de calcul nécessaire, etc. Ces informations sont utilisées pour constituer le profil de la machine enregistrée et le profil des calculs effectués par le système.
Mise à jour de la base de données	Le système doit permettre la mise à jour des machines serveurs et volontaires à chaque fois qu'il y a une opération comme l'enregistrement d'un nouveau volontaire, une mise à jour de profil, une demande de calcul, une réception de calcul, un envoi de résultat, etc.
Soumission du calcul	Le système doit fournir une interface permettant aux utilisateurs de pouvoir diviser leur calcul en tâches.
Prédiction de disponibilité	Le système doit permettre de déterminer à un moment donné l'ensemble des machines qui seront disponibles. Notons que dans les environnements à ressources limités, lorsque la machine est disponible, le système doit également tenir compte de la disponibilité de l'énergie électrique et de l'Internet.
Allocation des tâches	Le système doit permettre aux clients de pouvoir allouer des tâches à exécuter aux machines volontaires.
Exécution des tâches	Le système doit permettre l'exécution des tâches soumises aux volontaires. Lors de l'exécution des tâches, le système doit prendre en compte l'état de la machine : machine libre, présence d'électricité et présence des tâches à exécuter. Le système doit arrêter l'exécution d'une tâche si une activité de l'utilisateur est détectée ou si la période de disponibilité prédite est achevée.
Retour des résultats	Une fois une tâche exécutée, le système doit permettre à la machine qui a exécutée la tâche de retourner les résultats et les logs sur le déroulement du calcul au client qui l'a soumis.
Récupération des résultats des calculs	Le système doit permettre au client de recevoir tous les résultats des calculs qui ont été soumis aux volontaires.
Fusion des résultats	Le système doit fournir un mécanisme permettant au client de fusionner les résultats des calculs.
Feedback des calculs au serveur	Le système doit fournir un mécanisme permettant au volontaire de collecter les données sur le déroulement du calcul et au client de constituer un ensemble de rapports pour faire un feedback au serveur.
Statistiques	Le système doit pouvoir faire les statistiques générales. Ces statistiques vont permettre de connaître la puissance globale, la puissance de chaque volontaire, la contribution de chaque volontaire, les calculs effectués sur le système, etc.

TAB. 6.6 – Tableau présentant les cas d'utilisation du système

6.3.3 Conception de vcSoftware

Les spécifications nous ont permis de définir l'ensemble des exigences à satisfaire par le logiciel qui sera utilisé par le système *VC_UY1*. Les architectures que nous présentons dans cette section décrivent l'organisation du système en différents modules et la communication entre ces modules. Dans la suite, nous présentons l'architecture physique, l'architecture logique et le diagramme de classe persistant.

6.3.3.1 Architecture physique

Compte tenu du contexte des environnements à ressources limités, l'architecture que nous avons adopté pour la mise en place de *VC_UY1* est une architecture hybride présentée par la figure 6.4. Cette architecture est composée d'un serveur utilisé comme outil de stockage qui contient la description de toutes les ressources du système et de l'état d'exécution de tous les calculs ; d'un ensemble de volontaires, qui peuvent être à la fois nœud de calcul, nœud client ou nœud serveur. Les volontaires vont donc conserver les informations partielles sur les ressources et l'état d'exécution des tâches. Ces informations seront utilisées pour la planification lorsqu'ils ont un calcul à soumettre. La figure 6.4 a été annotée comme suit :

- (1) Enregistrement : lorsque un nouveau volontaire rejoint le système, il s'enregistre chez le serveur ;
- (2) Synchronisation de base de données : lorsqu'il y a une activité dans le système (enregistrement de volontaire, exécution de tâches, changement de profil d'un volontaire, etc.), les bases de données des serveurs et des volontaires sont mises à jour pour la prise en compte des nouvelles informations ;
- (3) Soumission du calcul : le client soumet un calcul sous la formes d'un ensemble de tâches. Dans la figure 6.4, nous prenons l'exemple d'une équation aux dérivées partielles, qui discrétisée et mise sous forme matricielle sera par la suite divisée en tâches;
- (4) Prédiction de disponibilité : une fois les tâches obtenues, les clients doivent utiliser les données stockées en local pour construire des groupes de volontaires en fonction de leurs capacités, de leur disponibilité, de leur réputation, de leur confiance et déterminer l'ensemble des machines susceptibles d'exécuter toutes les tâches jusqu'à la fin dans un délai raisonnable ;
- (5) Allocation des tâches : les taches obtenues à l'étape (3) sont combinées aux données nécessaires à leur exécution et allouées aux différentes machines volontaires identifiées à l'étape (4) ;
- (6) Exécution des tâches : les tâches reçues sur les machines volontaires sont exécutées par celles-ci. Les informations sur le déroulement de l'exécution sont enregistrées dans un fichier de logs ;
- (7) Retour des résultats : les résultats et les logs obtenus à l'étape (6) sont envoyés au client qui a soumis la tâche;
- (8) Fusion des résultats : les résultats des calculs sont fusionnés pour obtenir le résultat du calcul global ;
- (9) Feedback au serveur : le feedback sur l'exécution du calcul contenant les informations sur le temps d'exécution des tâches, la contribution des volontaires en terme de puissance de calcul sont envoyées au serveur. Le serveur va compiler pour aider les planificateurs à prédire la disponibilité des machines lorsqu'il y aura de nouveaux calculs.

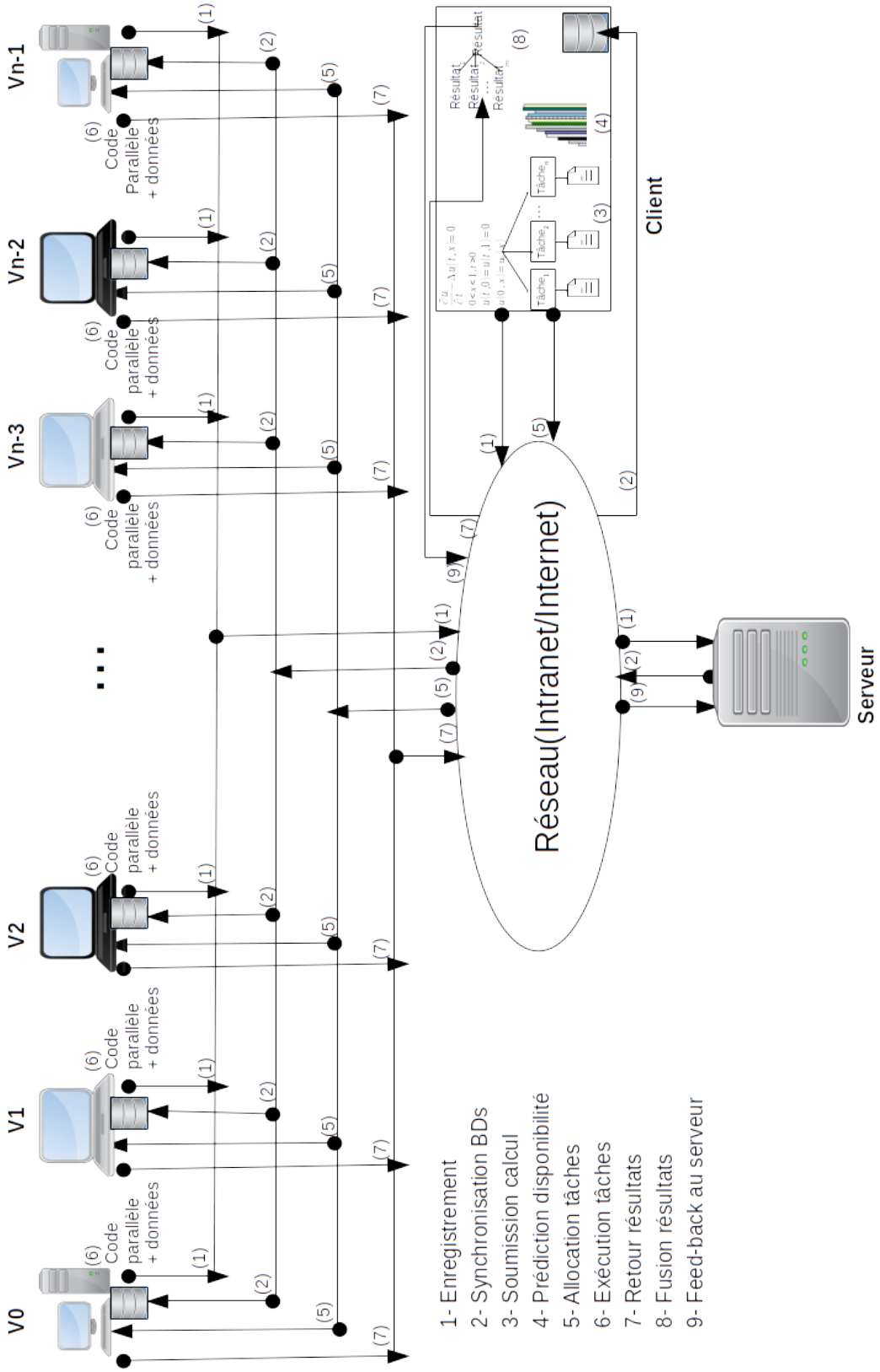

FIG. 6.4 – *Architecture physique*

Nous venons de présenter l'architecture physique de *VC_UY1*. Cette architecture fonctionne en utilisant le logiciel *vcSoftware* dont les différents composants seront présentées dans la suite.

6.3.3.2 Architectures logiques et diagramme de classe

Nous avons développé une architecture permettant de rendre l'application invisible à l'utilisateur de la machine. Pour ce faire, nous avons utilisé le conteneur Docker [4]. Le conteneur permet d'exécuter du code sur une machine en disposant de fonctionnalités données. Il protège la machine cible ainsi que les données qui y sont stockées contre toute attaque de sécurité qui pourrait provenir de code malicieux ou déficient chargé par le conteneur. Avec cette technologie, plusieurs processus et applications sont séparés les uns des autres afin d'optimiser l'utilisation de l'infrastructure tout en bénéficiant du même niveau de sécurité que celui des systèmes distincts. Ainsi elle permet d'héberger des services sur un même serveur physique tout en les isolant les uns des autres. Les conteneurs se partagent le même noyau du système d'exploitation et isolent les processus de l'application du reste du système [112].

Docker plus particulièrement est une technologie permettant d'automatiser le déploiement des applications (ou encore un ensemble de processus combinés qui forment une application) au sein d'un environnement de conteneurs. Il utilise un modèle de déploiement basé sur une image. Une *image container* est un ensemble de processus logiciels légers et indépendants, regroupant tous les fichiers nécessaires à l'exécution des processus : code, runtime, outils système, bibliothèques et paramètres. Il est donc plus simple à partager une application ou un ensemble de services, avec toutes leurs dépendances, entre plusieurs environnements. Docker permet d'exécuter les conteneurs, de simplifier leur conception et leur construction, l'envoie d'images, le contrôle des versions d'images, etc. Il permet d'empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quelle machine. Cette approche permet d'accroître la flexibilité et la portabilité d'exécution d'une application, laquelle va pouvoir tourner de façon fiable et prévisible sur une grande variété de machines hôtes. Cela permet aux nœuds d'être déployés au fur et à mesure que les ressources sont disponibles, offrant un déploiement transparent et similaire pour les systèmes distribués. Il utilise l'isolation des ressources comme le processeur, la mémoire, les entrées/sortes et les connexions réseau, ainsi que les espaces de noms séparés pour isoler le système d'exploitation. Il permet de simplifier la mise en œuvre des systèmes distribués en permettant à de multiples applications, tâches de fond et autres processus de s'exécuter de façon autonome sur une seule machine physique ou encore à travers un éventail de machines isolées. Ceci permet de déployer des nœuds en tant que ressources sur besoin, fournissant une plate-forme de déploiement scalable.

Avec cette architecture, les données de l'utilisateur, ses applications et ses sessions sont protégées contre toute faute ou malveillance du système de calcul que nous ajoutons dans sa machine. Il permet aussi aux applications de calcul d'être protégées contre les intrusions des utilisateurs.

Le conteneur dans lequel l'application utilisée est déployée va permettre d'activer les modules en fonction des machines, faire passer en mode calcul lors de l'arrivée du calcul, passer du mode normal en mode calcul. Cette information sera envoyée au serveur afin que les autres machines clientes puissent programmer les calculs en tenant compte des machines déjà occupées par les calculs. A la fin du calcul, le conteneur va passer en mode normal afin que l'utilisateur trouve sa machine dans le même état que lorsqu'il l'avait quittée. Dans le cas où l'utilisateur revient sur sa machine plutôt que prévu, l'activité utilisateur est détectée et la machine est remise en état normal, en abandonnant les éventuels travaux en cours.

Nous utilisons un mécanisme permettant de changer d'état de la machine entre son contexte utilisateur normal et un contexte d'exécution de calculs. Ainsi, le comportement du volontaire est observé et ses périodes d'inactivités sont identifiées. Une fois identifiées, ces périodes sont proposées aux utilisateurs. Ainsi, dans notre cas, ce sont les volontaires qui proposent leurs périodes d'inactivités ou qui valident une période d'inactivité proposée par le système. Mais, si pendant la période d'inactivité,

4. https://www.docker.com/

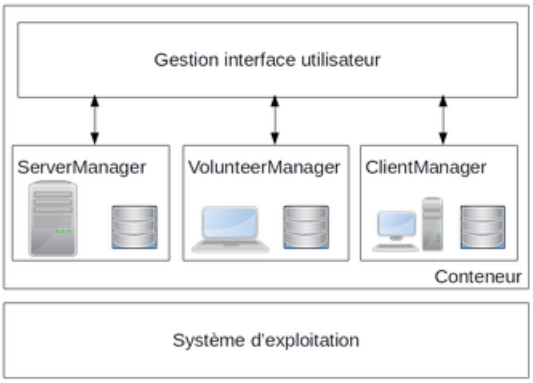

FIG. 6.5 – *Architecture globale du système*

le volontaire revient sur sa machine, le calcul est mis en pause. C'est pourquoi nous avons mis en place un système de capture et d'analyse du profil d'utilisation de la machine, qui nous permet de déterminer les périodes de disponibilité.

De façon globale, l'architecture logique de la plate-forme *vcSoftware* utilisée par notre système et présentée par la figure 6.5 est composée de :

- *clientManager* : le module *clientManager* est utilisé pour découper le calcul en tâches parallèles, soumettre, déterminer les machines susceptibles d'exécuter le calcul dans un délai raisonnable, distribuer le calcul aux volontaires, récupérer et fusionner les résultats pour obtenir le résultat du calcul et faire un feedback au serveur. Pour faire une meilleure prédiction de ressources, le *clientManager* synchronise périodiquement les données stockées localement avec les données stockées sur le serveur.
  Lorsqu'un volontaire veut soumettre un calcul, il active ce module et devient un client. Une fois ce module activé, une requête est envoyé au serveur afin d'obtenir la version la plus à jour des données sur les volontaires. Le volontaire va donc pouvoir lancer l'exécution de son calcul et informer le serveur ;
- *serverManager* : le module *serverManager* est le répertoire d'information du système. Il permet la collecte de données et la production des statistiques sur le fonctionnement du système et la synchronisation des données avec les bases de données des volontaires.
  Le module serveur est activé sur la machine désignée comme serveur. En cas de panne du serveur, le volontaire désigné pour le remplacer le temps qu'il soit réparé active tout simplement ce module.
- *volunteerManager* : ce module collecte les informations sur le profil du volontaire, permet la récupération des tâches au niveau du client, leur exécution en collectant les données sur cette exécution et le retour des résultats. Il assure aussi la mise à jour du profil du volontaire auprès du serveur.
  Un nouveau volontaire pour être dans le système doit tout simplement installer ce module.

Les figures 6.6, 6.7, et 6.8 présentent les architectures logiques du *clientManager*, du *serverManager* et du *volunteerManager*. Ces modules sont composés comme suit :

- Gestion de l'interface utilisateur : l'interface utilisateur est la partie de l'application qui assure le contact entre les utilisateur et la plate-forme. L'interface de *vcSoftware* permet à l'utilisateur de s'enregistrer, activer le module sur lequel il veut utiliser sa machine, déterminer ou visualiser ses périodes de disponibilité, avoir toutes les informations et les statistiques sur le système, visualiser la disponibilité des volontaires en fonction du temps, soumettre un calcul, allouer les tâches aux volontaires, visualiser la distribution de l'exécution du calcul par les volontaires et l'évolution de l'exécution globale.

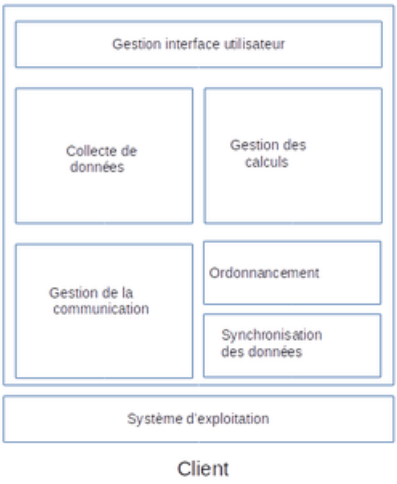

FIG. 6.6 – *Architecture du client*

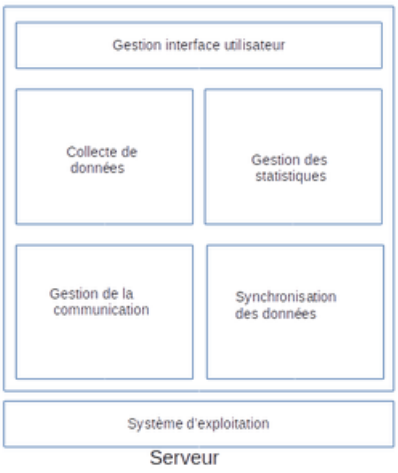

FIG. 6.7 – *Architecture du serveur*

– Gestion de la collecte de données : l'objectif de ce module est de collecter les données sur les caractéristiques des machines, leur performance et leur disponibilité afin de constituer une banque de données sur le comportement des machines. Il s'agira de mettre en place des outils qui vont enregistrer les caractéristiques de la machine, les activités de l'utilisateur, surveiller et enregistrer les différents évènements pouvant conduire à l'arrêt ou à la dégradation des performances d'un calcul sur une machine. Ces évènements peuvent être de différents types :
  – propriétaire : activités du clavier et de la souris,
  – matériel: charge du processeur et de la mémoire,
  – logiciel : disfonctionnement quelconque au niveau logiciel,
  – réseau : activités au niveau du réseau, présence ou non d'une connexion internet,
  – électricité : disponibilité ou non d'électricité.

  Pour ce faire, ce module de collecte de données est composé des processus qui sont actifs en permanence et dialoguent entre eux afin de maintenir à jour la base de données pour refléter au mieux l'état des machines du système.
– Gestion des calculs : en fonction de l'utilisateur, ce module permet la soumission du calcul, la distribution des tâches, le retour des résultats, la récupération et la fusion des résultats.
  Lorsque l'utilisateur est un client, ce module lui donne la possibilité de soumettre le calcul sous

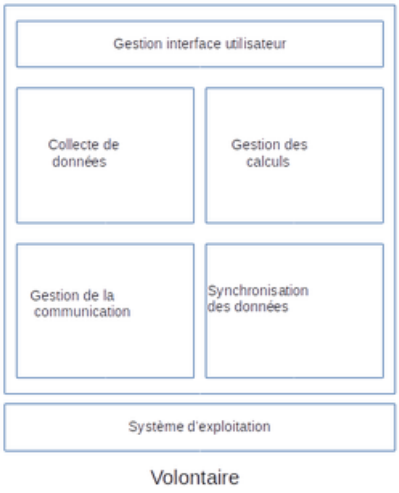

FIG. 6.8 – *Architecture du volontaire*

la forme d'un ensemble de tâches. Par la suite, ce module va récupérer la base de données des volontaires auprès du serveur et ensuite faire de la prédiction pour avoir la disponibilité des machines. Le module de prédiction de disponibilité va en fonction des machines disponibles, de leurs performances individuelles et des caractéristiques du calcul à lancer, déterminer l'ensemble des machines à allouer à ce calcul afin de pouvoir garantir sa terminaison dans un temps raisonnable. Chaque tâche se voit attribuer une date limite (délai), si le résultat n'est pas reçu après cette date, la tâche est réaffectée à un autre volontaire. Le module permet au client de soumettre les tâches aux volontaires en spécifiant le code de la tâche, ainsi les données nécessaires et où doivent être placées les fichiers de sortie des résultats. Une fois la liste des machines identifiée, les tâches leur sont envoyées. Un calcul est terminé lorsque toutes les tâches ont été soumises, exécutées, les sous-résultats récupérés et combinés pour obtenir le résultat final.

Au niveau des volontaires, ce module va permettre de récupérer les tâches chez un client, d'exécuter les tâches et de renvoyer les résultats au client. Pendant l'exécution de la tâche, la détection d'une activité utilisateur ou encore la fin de la période de disponibilité du volontaire provoque systématiquement la suspension de la tâche. Si le calcul est terminé, les résultats sont sauvegardés, sinon, le calcul est mis en pause ou abandonné. A la fin d'un calcul, les machines qui avaient été allouées se remettent en état disponible pour d'autres calculs. Le démarrage d'une tâche peut être planifié ou non. L'instance est dite planifiée lorsque les heures de début et de fin sont définies à l'avance et non planifiée lorsqu'elle démarre et se termine automatiquement en fonction de l'état de la machine. Une instance planifiée peut être ponctuelle (s'exécute une seule fois) ou répétitive. Pour qu'une tâche s'exécute, les conditions suivantes doivent être remplies : machine à l'état libre et présence d'électricité. Conditions d'arrêt d'une tâche : heure de fin, changement d'état de la machine, coupure d'électricité, arrêt de la machine et fin des tâches à exécuter.

Pendant l'exécution les traces d'exécution des tâches sont enregistrées dans un fichier différent toutes les heures. Ces traces contiennent les résultats partiels ainsi les temps d'exécution.

Une fois les calculs terminés, l'utilisateur doit pouvoir trouver sa machine dans le même état dans lequel il l'a laissé avant l'utilisation pour le calcul. Ce module doit aussi rendre l'application invisible à l'utilisateur de la machine du volontaire

– Gestion de la communication : le module de la communication assure les échanges entre les volontaires et les clients et le serveur. Les principaux échanges sont : envoie d'une mise à jour de profil, envoie d'un calcul, réception du calcul, envoi de résultats et envoie traces d'exécution. La communication est déclenchée à la demande d'autres modules et en présence d'une connexion

réseau. Sinon, la communication est retardée jusqu'à ce que la connexion soit disponible.

- Gestion de l'ordonnancement : ce module fonctionne étroitement avec le module de gestion de calcul. Il contient un outil permettant de prédire la disponibilité d'un ensemble de volontaires ; un sélecteur de ressources, qui va émettre des choix en termes de combinaisons de ressources disponibles pour un calcul donné ; un planificateur, qui va crée un ordonnancement conforme à une combinaison donnée de ressources. Le choix des machines pour les tâches tiennent compte de l'état de la machine, le choix du propriétaire de la machine et la présence des tâches à exécuter. Pour ce faire, la base de données au niveau de la machine cliente qui veut soumettre le calcul est utilisée ;
- Gestion de la synchronisation des données : chaque machine du système possède une base de données contenant les informations sur les autres machines. La base de données du serveur contient les informations brutes sur toutes les machines ; les bases de données locales sur les machines volontaires contiennent un résumé des informations stockées sur le serveur et pouvant être utilisée pour la prédiction de disponibilité. A intervalle régulier, les machines volontaires vont envoyer leurs informations au serveur. Ces informations seront compilées et les bases de données des machines clientes seront mises à jour.
- Production des statistiques : le module des statistiques est utilisé au niveau du serveur pour produire toutes les statistiques nécessaires à l'administrateur, au client et aux volontaires. Les principales statistiques sont la puissance maximale du système, l'ensemble des calculs déjà exécuté par le système classé par domaine, le taux de contribution de chaque volontaire, d'où viennent les volontaires et quel est le profil des meilleurs volontaires. Ces statistiques sont importantes pour motiver les volontaires.

Pour garder le système hautement autonome, l'installation est réduite à sa plus simple expression : exécution du programme, qui procède à l'installation de tous les composants. Quelques réglages permettent au client et serveur d'activer le mode client/serveur. Par défaut, l'application une fois installé va se mettre au mode volontaire. Le volontaire peut spécifier ses plages de disponibilités. S'il ne le fait pas, alors, le système construit son profil d'activité et détermine automatiquement en utilisant les méthodes présentées dans le chapitre 4 ses périodes de disponibilité.

Diagramme de classe Toutes les spécifications nous ont permis de construire le diagramme de classes persistent présenté à la figure 6.9. Ce diagramme de classe contient l'ensemble des classes qui seront persistées dans la base de données et les relations entre ces classes.

6.3.4 Présentation générale de VC_UY1

VC_UY1 est un système de calcul sur machines volontaires mis en place à l'Université de Yaoundé I en septembre 2020. Il est basé sur une architecture hybride composé d'un serveur, de volontaires et des clients :

- Le serveur : installé sur une machine équipée d'un processeur Intel Core i5 avec 4 cœurs cadensés à 2 Ghz, d'une mémoire de 4 Go et du système d'exploitation Linux Ubuntu 18.04. Elle a une disponibilité et une connexion Internet permanente ;
- Les volontaires : le système est construit à l'aide d'un ensemble de machines volontaires utilisées comme nœuds de calculs. Pour faire partir du système, le volontaire active tout simplement le module volontaire du logiciel *vcSoftware*. Par la suite, il peut éventuellement remplir un formulaire pour indiquer sa période de disponibilité. Ces informations sont utilisées pour initialiser la base de données de profil. S'il ne le fait pas, alors, le système va collecter un ensemble de données permettant de déterminer sa disponibilité. L'on distingue deux types de volontaires :
  - Machines dédiées : parmi les volontaires enregistrés, il y a 47 de machines dédiées. Ces machines sont les machines des salles de travaux pratiques, les ordinateurs dans les bureaux

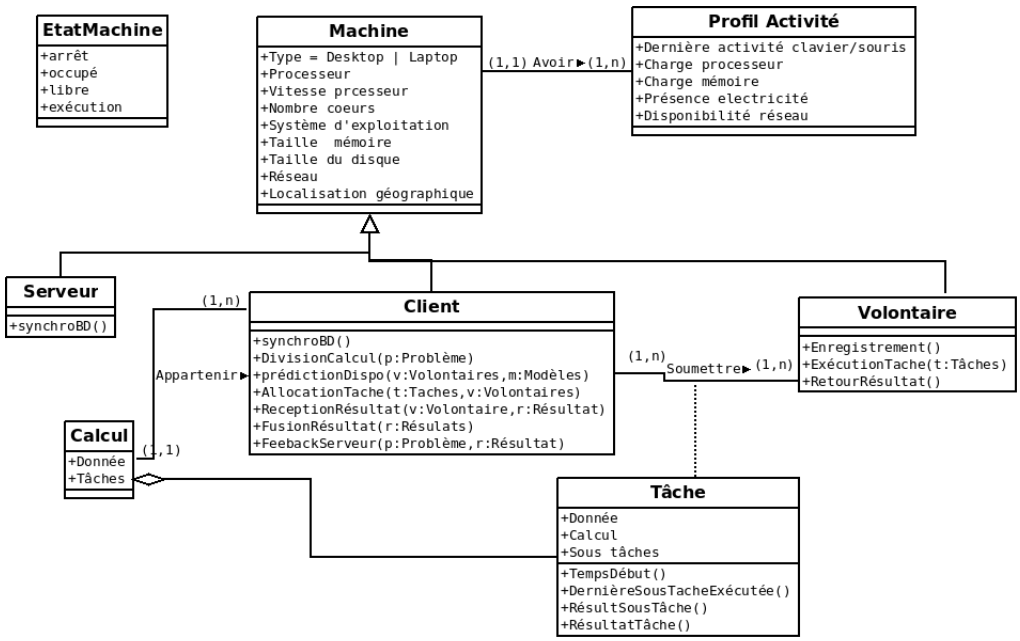

FIG. 6.9 – *Diagramme de classes*

du département et les ordinateurs au CUTI. D'après leurs utilisateurs, ces machines sont généralement libres pendant la soirée (18h-6h), les week-ends et pendant les congés ;

- Machines non dédiées : le système est également composé de 117 de machines non dédiées. Ces machines sont les ordinateurs portables détenus par les enseignants et les étudiants des départements d'informatique et de mathématiques.

– Le client : les clients sont les volontaires qui ont besoin d'utiliser le système.

Le tableau 6.7 présente la répartition des volontaires en fonction de type d'ordinateurs et de systèmes installés.

Type/Système	Windows		Linux		Total	
	Machine	Cœur	Machine	Cœur	Machine	Cœur
Desktop	40	5x4 + 35x2 = 90	10	10x2 = 20	50	110
Laptop	89	2x8 + 33x4 + 54x2 = 256	25	11x4 + 14x2 = 72	114	328
Total	**129**	346	**35**	92	**164**	**438**

TAB. 6.7 – *Répartition des caractéristiques (type, système et nombre de cœurs) volontaires enregistrés*

L'environnement logiciel de *VC_UY1* est constitué d'un logiciel appelé *vcSoftware*. Ce logiciel a été implémenté en utilisant le langage Python et la bibliothèque IPython. Plusieurs versions ont été développées en fonction du système d'exploitation (Linux et Windows) et la version de Python (Python2.x et Python3.x).Ce logiciel est composé de trois principaux modules :

- *serverManager* : activé au niveau du serveur, permet de faire la collecte de données sur les machines volontaires. Ces données sont traitées et la partie permettant aux volontaires de faire la prévision de disponibilité sont copiées sur les machines volontaires ;
- *volunteerManager* : activé au niveau du volontaire, ce module permet au volontaire de recevoir les calculs, les exécuter, renvoyer les résultats au client, faire la collecte de données et envoyer ces données au serveur;
- *clientManager* : activé par le client, il permet de découper le calcul, le distribuer aux volontaires, récupérer les résultats, les fusionner et envoyer le feedback au serveur.

Heure/Jour	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
06h-08h	8	7	7	8	6	12	16
08h-10h	9	6	8	7	5	13	17
10h-12h	9	10	10	10	8	12	16
12h-14h	8	9	9	10	7	12	14
14h-16h	9	8	11	10	10	12	14
16h-18h	8	7	6	7	8	12	15
18h-20h	9	8	7	9	6	12	14
20h-22h	9	9	6	9	6	12	14
22h-00h	10	8	8	8	6	11	14
00h-02h	17	14	17	14	12	18	22
02h-04h	16	15	16	14	12	18	22
04h-06h	17	14	16	14	12	18	22

TAB. 6.8 – *Disponibilité des machines en fonction du jour de la semaine et de l'heure de la journée*

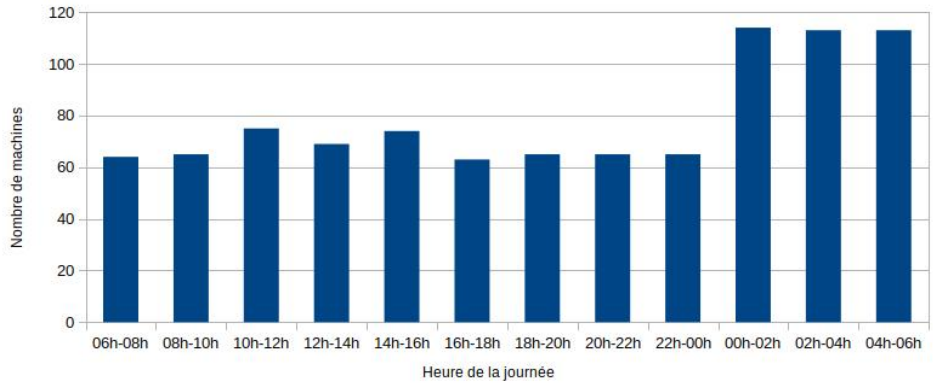

FIG. 6.10 – *Disponibilité des machines en fonction de l'heure de la journée*

Le tableau 6.8 et la figure 6.10 présentent la disponibilité des machines en fonction des heures de la journée et jour de la semaine. Pour une période de 2 heures, nous avons constaté que le nombre moyen de machines disponibles dans le système est de 11, avec un minimum de 5 et un maximum de 22. En plus les ordinateurs dans les salles de TP et les bureaux des enseignants sont les ressources les plus disponibles quotidiennement car elles sont complètement disponibles les soirs, les week-ends et les congés.

Notons que toutes les informations de disponibilité fournies dans cette section sont les informations fournies par les utilisateurs. Ainsi, avant le mettre le système en service, l'on dispose déjà une période de disponibilité. Contrairement aux approches proposés dans la littérature dans laquelle les données doivent être collectées sur le volontaire pendant un certain temps. En effet, les utilisateurs vont mettre leurs informations de disponibilité lors de leur enregistrement dans le système.

Les principales propriétés de *VC_UY1* sont :

- Exécution des tâches parallèles : le système permet d'identifier les machines disponibles pour du calcul et de résoudre les problèmes dans les meilleurs délais possibles.
- Résilience : le système tient compte de son environnement. Dans les environnements dégradés, plusieurs facteurs supplémentaires tels que la disponibilité de l'énergie électrique et de l'Internet permettent de déterminer si une machine est disponible. Dans notre cas, les lieux où il n'y a pas d'électricité sont considérés lors de la prédiction de disponibilité ;
- Scalabilité : le système peut passer très facilement à l'échelle. En effet, pour qu'un nœud de

calcul fasse partie du système, il suffit qu'il installe le logiciel *vcSoftware*, qui va directement activé le module volontaire. Si le volontaire fournit ses périodes de disponibilité, alors, les calculs peuvent lui être envoyés. Sinon, les données sont collectées en vue de faire de la prédiction de disponibilité ;

- S'organise et s'adapte automatiquement à la disponibilité et à la volatilité des machines, à la connectivité réseau variable, à un grand nombre de machines participantes ;
- Supporte facilement l'arrivée dynamique de nouveaux volontaires.

Dans la version actuelle de *VC_UY1* :

- un client pour soumettre un calcul au système, doit lui même diviser le calcul en tâches, rencontrer le responsable du système et lui soumettre son calcul. Le responsable du système va donc se charger de lancer les calculs à partir d'une machine cliente ;
- permet uniquement d'exécuter les applications avec les tâches parallèles indépendantes ;

6.4 Conclusion

Ce chapitre avait pour but de proposer une approche pour la mise en place de systèmes de calcul haute performance basés sur les machines des volontaires dans les environnements pauvres en ressources. A son terme, nous avons présenté ce qu'est un environnement pauvre en ressources. Nous avons vu que dans ces environnements, les ingénieurs et les scientifiques ont généralement besoin des systèmes de calcul haute performance pour effectuer leurs simulations. Étant donné que les solutions généralement utilisées par ceux-ci ne sont pas entièrement satisfaisantes, nous avons proposé une approche basée sur l'utilisation des machines des volontaires pour le calcul haute performance. Cette approche est bassée sur une architecture hybride composée d'un serveur, de volontaires et de clients. Le serveur est utilisé comme répertoire global d'informations sur le système. Les volontaires sont les machines des particuliers qui sont responsables de l'exécution des tâches. Les clients sont les machines qui ont besoin du calcul. Les clients vont diviser le calcul en tâches, les tâches seront soumises aux volontaires, qui vont les exécuter et renvoyer les résultats aux clients. Les clients vont agréger les résultats et feront un feedback sur l'exécution du calcul au serveur. Par la suite, nous avons présenté le processus qui nous a permis de recruter les volontaires et quelques statistiques sur les volontaires recrutés ; nous avons présenté les spécifications et la conception de la plate-forme de calcul *vcSoftware*, la plate-forme permettant au système de fonctionner. Dans le chapitre 7, nous présenterons l'expérimentation de notre système.

Chapitre Sept

Expérimentation de VC_UY1

L'approche que nous avons proposée dans cette thèse étant destinée aux applications nécessitant une grande puissance de calcul, il est indispensable de les valider expérimentalement sur ce type d'applications. En effet, pour que le système soit adopté et utilisé, les machines volontaires doivent fournir une puissance de calcul suffisante pour résoudre les problèmes informatiques associés. Dans ce chapitre, nous présentons donc l'expérimentation sur la multiplication de matrices de grande taille et sur la caractérisation de la résonance stochastique. Dans la suite de ce chapitre, nous commencerons par présenter les différents problèmes expérimentés à la section 7.1 et le protocole expérimental à la section 7.2. Par la suite, nous présentons l'expérimentation, la discussion et conclusion aux sections 7.3 et 7.4 respectivement.

7.1 Problèmes expérimentés

Deux problèmes ont été utilisés pour expérimenter la plate-forme *VC_UY1* : le problème mathématique de la multiplication des matrices de grandes tailles présenté à la section 7.1.1 et le problème physique de la caractérisation de la résonance stochastique présenté à la section 7.1.2.

7.1.1 Multiplication des matrices de grandes tailles

La multiplication des matrices [113, 114, 115, 116] est une opération qui effectue un produit matriciel à partir de deux matrices d'entrée. Chaque entrée du produit matriciel est le produit scalaire d'une ligne dans la première matrice et d'une colonne dans la seconde (voir l'équation 7.1). La multiplication matricielle a de nombreuses applications. C'est le noyau de base de calcul pour de nombreux algorithmes des systèmes d'apprentissage automatique et des systèmes de recommandation. La multiplication matrice-vecteur par exemple est l'élément central de l'algorithme PageRank [20, 115]. La multiplication des matrices est un problème en informatique car les matrices sont généralement de grandes dimensions, atteignant des centaines, des milliers et même des millions [113, 114, 115, 116].

Trois algorithmes populaires ont été proposés dans la littérature : l'algorithme itératif, l'algorithme récursif et l'algorithme de Strassen. Pour illustrer ces algorithmes, considérons deux matrices carrées A et B d'ordre n ci-dessous :

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{pmatrix}$$

La multiplication de la matrice A par la matrice B donne la matrice C obtenue en utilisant l'équation 7.1.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{7.1}$$

À partir de l'équation 7.1, un algorithme itératif simple peut être construit en utilisant des boucles sur les indices i, j de 1 à n. Cet algorithme est d'ordre $O(n^3)$. En utilisant l'approche diviser pour régner, les matrices A et B sont partitionnées en blocs d'ordre $n/2$. La multiplication donne donc:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

L'approche diviser pour régner fonctionne pour toutes les matrices carrées dont les dimensions sont des puissances de deux. Pour les matrices qui ne respectent pas cette condition, l'on rempli les lignes et les colonnes manquantes par des zéros. La complexité de l'approche diviser pour régner est la même que celle de l'algorithme itératif, c'est à dire est de l'ordre de n^3. Dans notre exemple, cette approche nécessite 8 multiplications de blocs pour calculer la matrice de produit, qui nécessite toujours n^3 de temps d'exécution [113, 114, 116]. Notons toutefois que l'avantage de cette approche par rapport à l'approche itérative est qu'elle peut être plus facilement exécutée sur un système de calcul parallèle.

Une autre approche de multiplication des matrices est appelée algorithme de Strassen [113, 117]. Cet algorithme est une approche récursive diviser pour régner dans laquelle pour chaque appel récursif, les matrices d'entrées sont divisées en 4 blocs, mais seulement 7 multiplications de blocs sont nécessaires. Comparée à l'approche itérative et à l'approche diviser pour régner, l'approche de Strassen n'a besoin que de 7 multiplications matricielles par blocs, ce qui donne une complexité de $n^{2.807}$ [113, 114, 116]. Lorsque les matrices sont de grandes tailles, le temps d'exécution de la multiplication matricielle peut être très long. Comme la multiplication matricielle peut être divisée en multiplication sous-matricielle, cette tâche peut être parallélisée.

La multiplication des matrices de grande taille est assez caractéristique des applications de calcul scientifiques d'aujourd'hui car demande une grande puissance de calcul. Dans la suite, nous présenterons la résonance stochastique qui est le second problème sur lequel à porté nos expérimentions.

7.1.2 Caractérisation de la résonance stochastique

La résonance stochastique (*stochastic resonance*, SR) est un phénomène non linéaire dans lequel la transmission d'un signal utile peut être améliorée par l'ajout du bruit au système [118, 119]. La caractérisation de la résonance stochastique étudie les effets potentiels bénéfiques du bruit dans le traitement de l'information. Cette étude a été mise en évidence dans plusieurs domaines dont l'électronique, l'économie financière et la médecine.

La deuxième expérimentation du système *VC_UY1* portera sur le travail du Prof. Djuidjé [120] effectué sur la caractérisation de la résonance stochastique dans un système à N particules couplées via des ressorts non linéaires se déplaçant unidirectionnellement dans un potentiel sinusoïdal périodique $U(x) = -u_0 sin(x)$ dans un milieu à coefficient de frottement λ. Chaque particule affiche un comportement particulier et est entraînée par une force périodique externe $F(t) = F_0 cos(\omega t)$ de fréquence ω. La variation du paramètre de couplage ainsi que le nombre de particules affectent l'apparition du phénomène de la résonance stochastique dans le système notamment la première et la dernière particule. L'énergie de la résonance $< \overline{W} >$ d'une particule est obtenue à partir de l'énergie d'entrée $< W >$ en faisant d'abord une moyenne sur les périodes puis sur les positions initiales. Cette énergie est donnée par l'équation 7.2 :

$$< \overline{W} > = < \int_0^{(N_1+1)\tau} x(t)\omega A_0 sin(\omega t) dt) > \tag{7.2}$$

Les positions $x(t)$ des différentes particules sont déterminées en résolvant le système des équations de Langevin décrit dans [120] par la méthode de Runge-Kutta de quatrième ordre pour les processus stochastiques [121].

Soit $x(t)$ est la solution de l'équation différentielle 7.3.

$$\dot{x} = F(x,t) + \xi(t) \tag{7.3}$$

Alors $x(t)$ peut être approchée au temps t_{k+1} par le système d'équations 7.4 :

$$\begin{cases} x_{k+1} = x_k + \alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \alpha_4 k_4 \\ k_1 = hF(x_k,t_k) + h(Dq_1)^{1/2} r_1 \\ k_j = hF(x_k + \sum_{i=1}^{j-1} \alpha_{j,i} k_i, t_k + hc_j) + h(Dq_j)^{1/2} r_j \end{cases} \tag{7.4}$$

Le système est initialisé avec les particules placées à la séparation uniforme et leurs vitesses initiales sont nulles $\dot{x}_i(0) = 0$. Dans le calcul approché de $x(t)$, le temps maximal t_{k+1} est pris de telle sorte que le système atteigne l'ergodicité. Le système est dit ergodique s'il satisfait la propriété qui stipule qu'à l'équilibre, la valeur moyenne de l'énergie est sensiblement égale ou égale à la moyenne calculée sur une longue période de temps. L'hypothèse ergodique est donc fondamentale pour un bon rapprochement entre la théorie et l'expérience.

D'après Prof. Djuidjé, l'atteinte de l'ergodicté demande une puissance de calcul considérable. Les expérimentations qu'elle a effectuées sur une machine équipée d'un processeur Intel core i3 et d'une mémoire de 4 Go ont duré une trentaine de jours environ. Durant les expérimentations, plusieurs coupures d'électricité ont été enregistré l'obligeant de relancer à chaque fois les calculs pour récupérer les résultats perdus. La machine sur laquelle les expérimentations ont été initialement lancées est tombée en panne avant la fin, ceci en raison du fait que le processeur a fonctionné trop longtemps sans s'arrêter et de l'absence d'un bon système de ventilation et d'aération. Dans la suite nous présentons le protocole expérimental utilisé pour résoudre les problèmes mentionnés ci-dessus.

7.2 Protocole expérimental

Pour valider notre hypothèse selon laquelle des problèmes de calcul complexes et volumineux peuvent être résolus à moindre coût et dans un délai raisonnable, nous présentons dans cette section la description des conditions et du déroulement des expériences menées. Dans ce qui suit, nous commençons par présenter les volontaires qui ont été sélectionnés pour l'expérimentation dans la section 7.2.1. Par la suite, nous présentons l'infrastructure et la description de l'expérimentation dans les sections 7.2.2 et 7.2.3 respectivement.

7.2.1 Sélection des volontaires

Pour expérimenter notre approche, un sous-ensemble de 74 volontaires a été sélectionné. Pour ce faire, nous les avons rencontrés physiquement ou par téléphone. De façon globale, les volontaires sélectionnés pour l'expérimentation peuvent être classés en 39 machines dédiées (52,7%) et 35 machines non dédiées (47,3%):

- **Machines dédiées** : ce sont les machines des salles de travaux pratiques du département d'informatique, les machines du CUTI et les ordinateurs dans bureau des enseignants. Lorsque les étudiants sont en période d'examen, ces machines sont disponibles 24 heures sur 24 (salles de TP) et d'autres sont disponibles toute la nuit de 18 heures à 8 heures (minimum 12 heures par jour).
- **Machines non dédiées** : ce sont les volontaires (étudiants et enseignants) avec des ordinateurs portables. Les volontaires qui étaient prêts à se rendre à l'Université ont été retenus. Ceci parce que nous ne disposions pas assez d'argent pour rembourser leur connexion Internet.

7.2.2 Infrastructure d'expérimentation

Après avoir sélectionné les volontaires à utiliser pendant l'expérimentation, nous présentons l'environnement matériel, l'environnement logiciel et le processus de communication utilisés pendant l'expérimentation.

7.2.2.1 Matériel

Le matériel utilisé pour l'expérimentation a été organisé en trois groupes : les serveurs, les machines volontaires et les clients.

- Serveur : pour faciliter l'expérimentation, nous avons mis en place un serveur au niveau du CUTI et un serveur au niveau du département d'informatique pour le stockage des informations produites par les machines connectées au réseau local du CUTI et celui du département. Le serveur du département était équipé d'un processeur GenuineIntel avec quatre cœurs cadencés à 3 Ghz, d'une mémoire de 8 Go et du système d'exploitation Linux Ubuntu 18.04. Le serveur du CUTI était équipé d'un processeur Intel(R) Core(TM) i5-4590T avec 4 cœurs fonctionnant à 2 Ghz, d'une mémoire de 4 Go et du système d'exploitation Linux Ubuntu 18.04.
- Volontaires : les machines volontaires étaient constituées de 20 machines de salle de travaux pratiques du département d'informatique, 11 machines d'enseignants, 8 machines du CUTI et 35 machines d'étudiants. Ces machines avaient des processeurs de différentes familles, chacune disposant d'un nombre variable de cœurs compris entre 2, 4 et 8 avec pour fréquence variant entre 800 Mhz et 3600 Mhz. Les tailles des mémoires vont de 2 Go à 16 Go. Le tableau 7.1 présente une description de l'environnement matériel des machines volontaires utilisées.

Type/Système	Windows		Linux		Total	
	Machine	Cœur	Machine	Cœur	Machine	Cœur
Desktop	28	1x4 + 27x2 = 58	11	11x2 = 22	39	80
Laptop	25	3x4 + 22x2 = 56	10	8x4 + 2x2 = 36	35	92
Total	53	114	21	58	74	172

TAB. 7.1 – *Description de l'environnement matériel des machines volontaires*

- Client : durant l'expérimentation, la machine serveur a également été utilisée comme machine client.

7.2.2.2 Logiciel

Selon les résultats du recrutement des volontaires, 53 (71,62%) machines utilisent Windows (à partir de la version 7) et 21 (28,38%) utilisent Linux, plus particulièrement Ubuntu. C'est pourquoi nous avons développé deux versions du logiciel pour nos expérimentations. Ce logiciel a été développé en utilisant le langage Python, plus particulièrement la bibliothèque IPython [1]. Plusieurs versions ont été développées en fonction du système d'exploitation (Linux et Windows) et la version de Python (Python2.x et Python3.x) de python qu'ils peuvent supporter.

Les code séquentiels (voir 7.1 et 7.3) et les code parallèles (7.2 et 7.4) de la multiplication des matrices et de la résonance stochastique ont été implémentés. Étant donné que nous sommes dans un système dans lequel les tâches ne communiquent pas entre elles, nous utilisons le modèle de programmation SPMD (*Simple Program Multiple Data*). En effet, le modèle SPMD implique que toutes les machines exécutent le même programme sur des paramètres différents. Les problèmes sont donc découpées en tâches et chaque tâche est gérée par une machine volontaire différente.

1. `https://ipython.org/`

Listing 7.1: *Algorithme séquentiel de la multiplication des matrices*

```
MatMult(A,B,n)
    Chargement des matrices A et B
    Pour i = 1 a n faire
        Pour j = 1 a n faire
            cij = 0
            pour k = 1 a n faire
                cij = cij + aik x bkj

MatMultBlocSeq(A,B,n)
    Decoupage des Matrices A et B en blocs de taille 2000x2000
    m = n / 2000
    Pour i = 1 a m faire
        Pour j = 1 a m faire
            Cij = 0
            Pour k = 1 a m faire
                Cij = Cij + MatMult(Aik,Bkj,2000)
```

Listing 7.2: *Algorithme parallèle de la multiplication des matrices*

```
MatMultBlocPar(A,B,n)
    machine client :
        Decoupage des Matrices A et B en blocs de taille 2000x2000
        m = n / 2000
        Envoie des blocs Aik et Bkj aux volontaires, 1 <= i,j,k <= m
    machine volontaire :
        Rijk = MatMult(Aik,Bkj,2000), 1 <= i,j,k <= m
        Envoie du resultat Rijk au client
    machine client :
        Pour i = 1 a m faire
            Pour j = 1 a m faire
                Cij = 0
                Pour k = 1 a m faire
                    Cij = Cij + Rijk
```

Listing 7.3: *Algorithme séquentiel de la résonance stochastique*

```
energieMoyT(n,T,i)
    kmax = 10000100
    N1 = 100000
    Pour j = 1 a n faire
        Initialiser x, la position de la particule j
    som = 0
    t = 0
    Pour k = 1 a kmax faire
        Pour j = 1 a n faire
            calcul de K1, coefficient de RK4
            calcul de K2, coefficient de RK4
            calcul de K3, coefficient de RK4
            calcul de K4, coefficient de RK4
            x = x + a1 * K1 + a2 * K2 + a3 * K3 + a4 * K4
        som = som + * x * w * A0 * sin( w * t)
        t = t + h
    som = (2 * som * h) / N1

energieMoySeq(nbParticule)
    Temperature = 0.001
    Tmax = 1.1
    dT = 0.005
    Pmax = 50
    Tant que Temperature <= Tmax faire
        somT = 0
        Pour positionInitiale = 1 a Pmax faire
            somT = somT + energieMoyT(nbParticule,Temperature,positionInitiale)
        somT = somT/pmax
        Temperature = Temperature + dT
```

Listing 7.4: *Algorithme parallèle de la résonance stochastique*

```
energieMoyPar(nbParticule)
    machine client :
        Temperature = 0.001
        Tmax = 1.1
        dT = 0.005
        Pmax = 50
        Envoie des calculs Enti aux volontaires
    machine volontaire :
        Enti = energieMoyT(n,T,i)
        n = nbParticule, Temperature <= T <= Tmax, 1 <= i <= Pmax
    machine client :
        Tant que Temperature <= Tmax faire
            somT = 0
            Pour positionInitiale = 1 a Pmax faire
                somT = somT + Enti
            somT = somT/pmax
            Temperature = Temperature + dT
```

En résumé, nous avons développé un outil composé des modules :

- *serverManager* : lors de l'expérimentation, l'application utilisée au niveau du serveur a été développée pour collecter les données sur les logs pendant l'exécution des tâches sur les machines volontaires ;
- *clientManager* : ce module est celui que nous avons utilisé pour diviser les calculs en tâches, distribuer les taches aux volontaires, récupérer les résultats et les fusionner. Pour faciliter la distribution du calcul aux volontaires, le code et les données sont compressés dans un fichier et distribués.
- *volunteerManager* : l'application au niveau du volontaire est responsable de l'exécution de la tâche soumise. Une fois le calcul copié sur la machine du volontaire, il l'exécute. Durant l'exécution, plusieurs points de sauvegarde ont été mis en place afin qu'en cas de coupure de courant, le calcul ne soit pas redémarré à partir de zéro. Une fois le calcul terminé, les résultats et les logs sur le déroulement du calcul sont envoyés au client.

7.2.2.3 La communication

Le principal problème que nous avons rencontré en utilisant les machines des volontaires à distances à travers le réseau Internet est la mise en disposition de ces volontaires d'une connexion Internet. En effet, la majorité des volontaires étaient des étudiants et l'une des conditions pour faire participer leurs machines devait être de leur rembourser le crédit Internet. Nous les avons donc invités lors de leur passage au campus à participer au calcul en connectant leur machine au réseau local mis en place à cet effet. Une fois leur machine connectée au réseau local (CUTI ou département) et le module *voluneerManager* installé, la collecte de données est lancée, le volontaire reçoit une tâche, l'exécute et renvoie les résultats et les logs.

7.2.3 Description de l'expérimentation

L'expérimentation s'est déroulée en trois étapes principales :

- Phase de test : pendant la phase de test, il s'agissait de vérifier la faisabilité d'un tel système dans notre environnement. Pour cela, nous avons invité les étudiants de niveau master à fournir du temps libre de leur machine pour du calcul. Une dizaine de machines d'étudiants avec accès Internet ont été sélectionnées pour les tests.
- Expérimentation sur les grosses matrices : la deuxième phase de l'expérimentation a consisté en une expérimentation d'envergure sur un plus grand nombre de machines et sur des tailles de matrices plus grandes. Nous avons contacté les étudiants et les enseignants enregistrés comme volontaires et 74 volontaires ont répondu à notre appel.
  Les matrices de nos expériences ont été construites à l'aide du générateur de nombres pseudo-aléatoires utilisant la fonction *random*() de la bibliothèque *random* du langage Python. Cette fonction génère un nombre à virgule flottante aléatoire de façon uniforme dans la plage semi-ouverte [0.0, 1.0). Au cours de l'expérience, différentes tailles de matrices ont été testées afin d'observer la variation des temps d'exécution et de la puissance de calcul.
- Expérimentation du problème de la résonance stochastique : la troisième phase de l'expérimentation a été consacré à la caractérisation de la résonance stochastique. Un sous-ensemble de 17 volontaires a été utilisé pour calculer l'énergie moyenne d'un ensemble de 4, 8, 12, 16 et 20 particules respectivement à partir de 50 énergies d'entrée par particule.

7.3 Expérimentations et résultats

Après la présentation du protocole expérimental, cette section est consacrée à l'expérimentation de notre système. Elle a commencée par une phase de test présentée dans la section 7.3.1, suivie

d'une phase d'expérimentation sur la multiplication de matrices de grande taille présentée dans la section 7.3.2 et s'est terminée par une phase d'expérimentation sur la caractérisation de la résonance stochastique dans la section 7.3.3.

7.3.1 Phase 1 : test du système

L'objectif de la phase de test était de bien comprendre la possibilité de faire du calcul haute performance et de déterminer les défis permettant de mettre en place des systèmes de calcul sur machines volontaires. La multiplication matricielle a été utilisée avec différentes tailles d'entrées sur un ordinateur (CPU Core i5 et 4 Go de mémoire RAM) et un ensemble de 10 machines décrit dans le tableau 7.2.

RAM	**CPU**	**Swap**	**système**	**Disque**
4Go	Core i3, 2,4GHz	1Go	Linux	50Go
4Go	Core i3, 2,4GHz	2Go	Windows	100Go
4Go	Core i4, 2,4GHz	2Go	Linux	100Go
4Go	Core i5, 2,7GHz	1,5Go	Linux	100Go
4Go	Core i5, 2,4GHz	1,5Go	Linux	50Go
4Go	Core i5, 2,4GHz	1Go	Linux	50Go
4Go	Core i3, 2,4GHz	2Go	Windows	50Go
4Go	Core i4, 2,4GHz	750Mo	Linux	100Go
4Go	Core i4, 2,6GHz	1Go	Linux	100Go
4Go	Core i4, 2,4GHz	2Go	Linux	100Go

TAB. 7.2 – *Caractéristiques des machines volontaires*

Si A et B sont deux matrices partitionnées en blocs, la taille de bloc de la matrice C résultante est déterminée en considérant le nombre de volontaires et la taille des blocs d'entrée. Un volontaire de chaque bloc résultant récupère tous les blocs nécessaires de A et B et exécute localement une opération de multiplication. L'algorithme de la multiplication des matrices Strassen a été implémenté en utilisant Python et de l'API IPython. La figure 7.1 montre les performances de l'exécution sur une machine. La figure 7.2 présente les performances de l'exécution sur un système composé de 10 machines et la figure 7.3 présente la contribution de chaque volontaire au calcul. La figure 7.3 montre que bien que les volontaires aient utilisés une puissance de calcul similaire, leur contribution au calcul varie en fonction de leur disponibilité.

7.3.2 Phase 2 : multiplication des matrices de grandes tailles

Après la phase de test, les expérimentations sur un grand nombre de volontaires sur les matrices plus larges ont été réalisées. Différentes configurations ont été étudiées afin d'avoir une vision relativement complète des éléments impactant la puissance de calcul de l'ensemble du système. Le but étant d'étudier le comportement de l'ensemble du système sur un grand panel de machines lorsqu'elles sont mises ensemble.

Nous avons repris les expérimentations en utilisant une seule machine afin d'effectuer la multiplication par blocs sur les matrices de plus grandes tailles. Une machine avec un processeur Intel Core i5, 8 Go de mémoire et le système Ubuntu a été utilisée. Les matrices A et B sont divisées en blocs de taille 2000×2000, l'algorithme de multiplication matricielle classique est ensuite utilisé sur ces blocs et les résultats sont à la fin agrégés pour obtenir la matrice résultat C. La figure 7.4 présente le résultat de l'expérimentation sur une seule machine. D'après cette figure, le temps d'exécution croit très rapidement avec la taille des matrices. Il passe d'environ 10 minutes à presque 100 minutes (1h40mn) lorsque la taille des matrices passe de 10000×10000 à 20000×20000, ce temps atteint 700 minutes (11h40mn) pour les matrices 40000×40000.

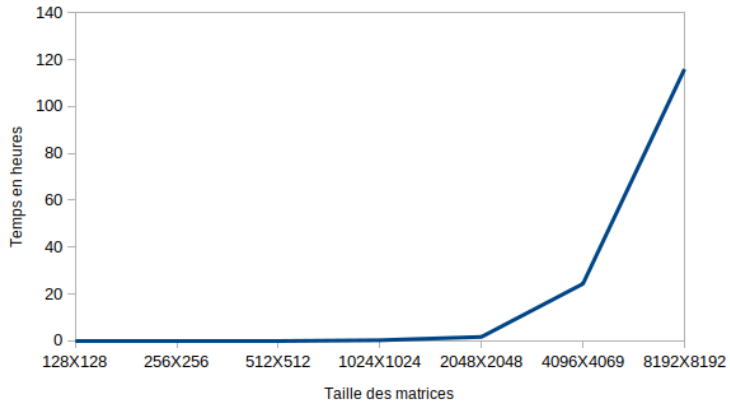

FIG. 7.1 – *Expérimentation de l'algorithme Strassen pour la multiplication matrice-matrice sur une machine Core i5 de CPU et 4 Go de RAM*

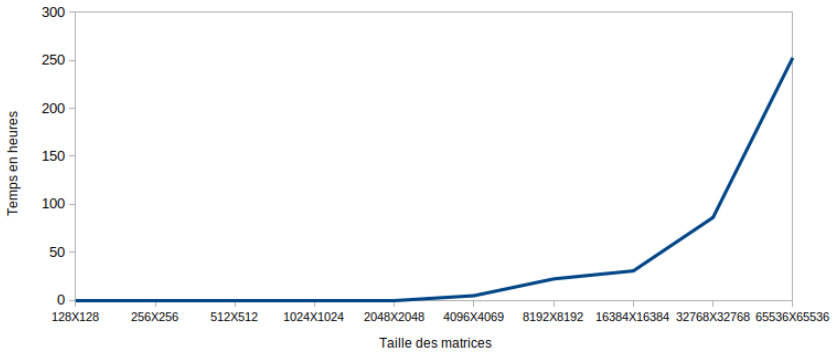

FIG. 7.2 – *Expérimentation de l'algorithme Strassen pour la multiplication matrice-matrice sur notre système informatique volontaire*

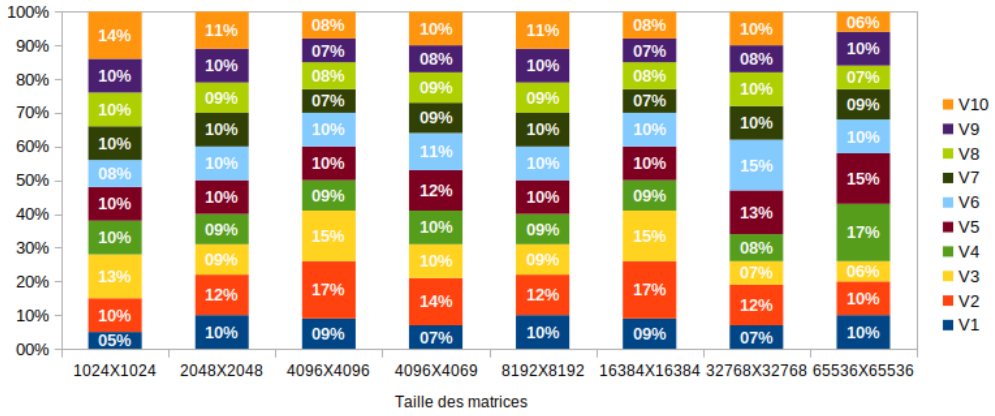

FIG. 7.3 – *Contribution de chaque machine. V_i représente la ième machine volontaire*

Nous avons continué les expérimentations en utilisant les machines volontaires sélectionnées durant la phase de la mise en place du protocole expérimental. Au cours de ces expérimentations qui ont duré une dizaine de jours (précisément 253 heures), des matrices jusqu'à 200000×200000 ont été multipliées. Pendant les 2 premiers jours de l'expérimentation, moins de 5 volontaires ont participé

activement aux calculs, la puissance de calcul était inférieure à 10 GFlops (voir figure 7.7). Environ 11 volontaires ont participé le 3e jour et la puissance de calcul a atteint 30 GFlops. Les 3 jours suivants (jours 4, 5 et 6), le nombre de volontaires a varié entre 20 et 25 et la puissance entre 60 et 80 GFlops. Entre les jours 7 et 9, le nombres de volontaires variait entre 25 et 40 et la puissance entre 80 et 110 GFlops. Le nombre de machines est tombé à 3 le dernier jour et la puissance à moins de 10 GFlops. La plus grande puissance obtenue est de 112 GFlops et le nombre maximum de volontaires participant au calcul à un moment donné de 42. La puissance cumulée obtenue est de 14,36 TFlops.

Le tableau 7.3 présente pour chaque taille de matrices expérimentées, le nombre de blocs de taille 2000×2000 obtenu après le découpage, le nombre de multiplications de blocs, le nombre de tâches générées et le nombre de volontaires ayant été utilisé. Chaque tâche contient 100 multiplications de blocs 2000×2000. Le résultat de l'expérimentation sur les machines volontaires est présenté par la figure 7.5.

Taille	2000	4000	6000	8000	10000	20000	40000	60000	80000	100000	200000
Nb_Bloc	1	4	9	16	25	100	400	900	1600	2500	10000
Nb_Mult_Blocs	1	8	27	64	125	1000	8000	27000	64000	125000	1000000
Nb_Tâches	1	1	1	1	2	10	80	270	640	1250	10000
Nb_Machine	1	1	1	1	1	3	18	23	25	25	68

TAB. 7.3 – *Caractéristiques des calculs exécutés sur les machines volontaires*

La multiplication des matrices 200000×200000 a généré une quantité de calcul autour de 16 PFlops sur une période d'environ 92h. La participation de 68 machines volontaires a été enregistrée et environ 1635 heures (plus de 68 jours) de calculs au total ont été effectuées. Chaque machine volontaire a contribué en fonction de sa puissance de calcul et de sa disponibilité. Sur les 10 000 tâches générées pour la multiplication des matrices 200000×200000, la contribution moyenne par volontaire a été de 147 tâches (1,47%), ce qui correspond à 88,54 heures (3jr16h30min) de calcul. La contribution minimale a été de 2,5 tâches (0,025%) équivalent à 59 minutes de calcul et la contribution maximale de 542 tâches (5,41%) soit environ 24 heures de calcul.

L'accélération est généralement définie comme le rapport du temps d'exécution d'un programme séquentiel et le temps d'exécution du programme parallèle équivalent. Durant notre expérimentation, nous avons définit un autre type d'accélération comme le temps d'exécution sur une machine et le temps d'exécution sur le système VC_UY1. La figure 7.6 présente l'accélération obtenue pour chaque taille de matrice exécutée. Pour les tailles de matrices au delà 40000×40000 sur une machine, les temps ont juste été estimée à partir des précédentes exécution sur la machine avec un processeur core

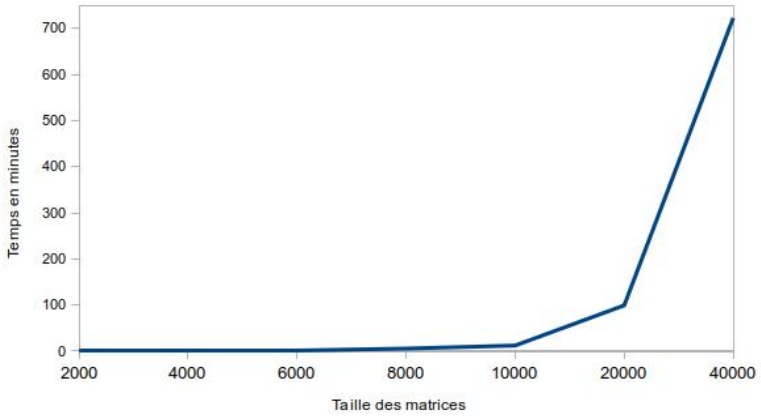

FIG. 7.4 – *Expérimentation de la multiplication des matrices sur une machine*

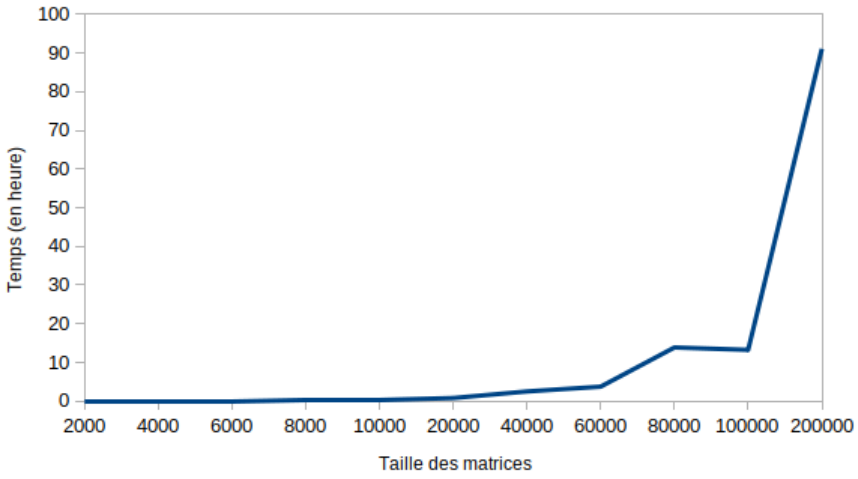

FIG. 7.5 – *Expérimentation de la multiplication des matrices sur les machines volontaires*

i5 avec 8 Go de mémoire. La figure 7.6 montre que notre système permet d'aller jusqu'à 23 fois plus vite que sur une machine. Ainsi plus nous avons de machines volontaires, plus les problèmes de plus grande taille pourront être résolus en temps raisonnable.

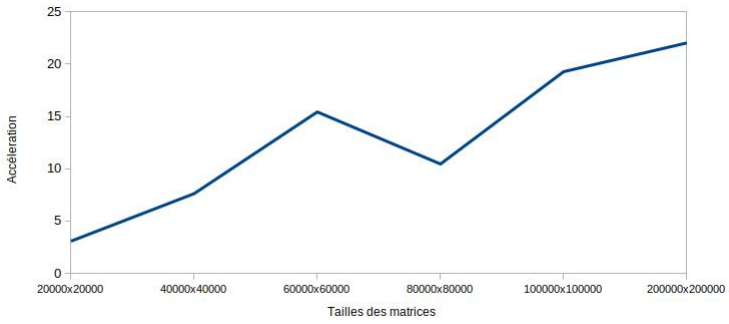

FIG. 7.6 – *Accélération pour chaque taille de matrice exécutée*

Le nombre de volontaires dans le système dans chaque intervalle horaire ainsi que la puissance de calcul obtenue sont présentés dans la figure 7.7

La figure 7.8 montre la puissance de calcul cumulée en fonction de l'heure de la journée. La puissance augmente entre 8h et 16h. Elle diminue de 16h à 20h et augmente à nouveau après 20h. Le pic se situe entre 14h et 16h.

7.3.3 Phase 3 : résonance stochastique

La troisième phase des expérimentations a été consacrée à la caractérisation de la résonance stochastique. La résonance stochastique a été expérimentée sur un nombre variable de particules n entre 4 et 20. La première partie des expériences a été réalisée sur une machine avec un processeur Intel Core i5, 8 Go de mémoire et le système Ubuntu. Le tableau 7.4 présente le résultat de l'expérimentation sur une seule machine.

D'après ce tableau, le temps d'exécution croit proportionnellement au nombre de particules. Le temps d'exécution totale pour la caractérisation de toutes les valeurs de n (4, 8, 12, 16 et 20) particules est de 11j07h21mn.

La suite des expérimentations s'est effectuée sur le système *VC_UY1*. Pour chaque valeur du

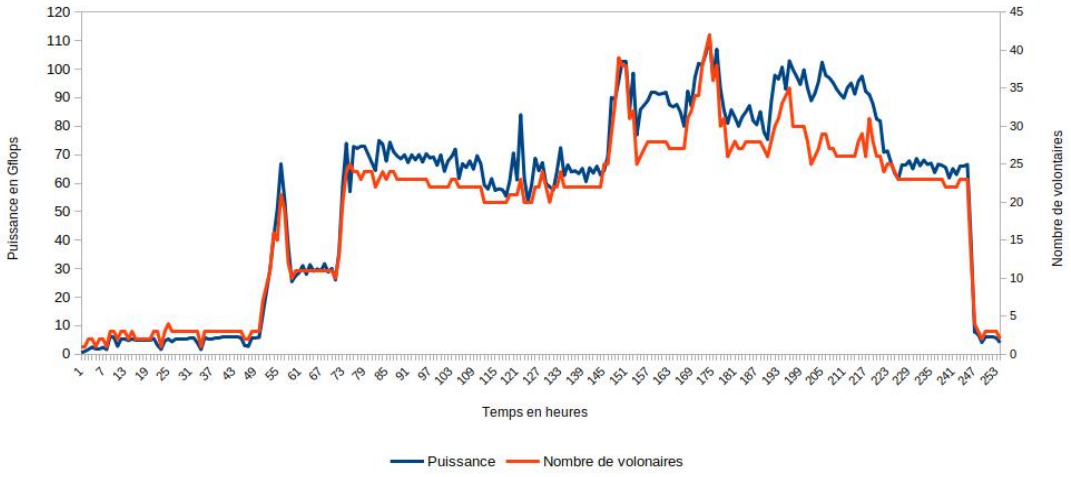

FIG. 7.7 – *Puissance de calcul en fonction du temps et du nombre de volontaires*

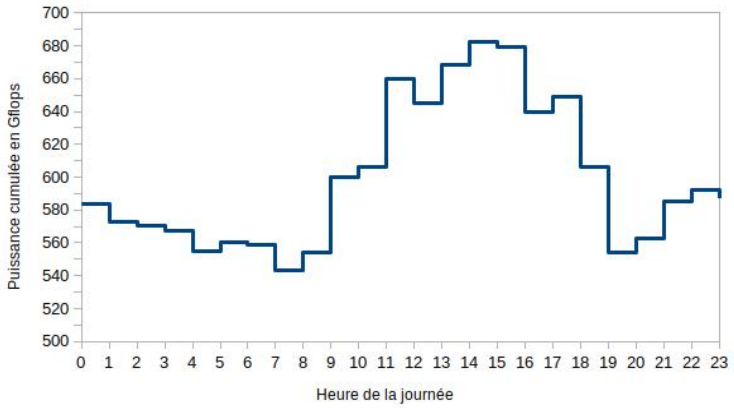

FIG. 7.8 – *Puissance cumulée de calcul en fonction de l'heure de la journée*

Nombre de particules	4	8	12	16	20	**Total**
Temps d'exécution	16,11h (00j16h06mn)	35,4h (01j11h24mn)	52,28h (02j04h16mn)	73,95h (03j01h57mn)	93,62h (03j21h37mn)	**271,35h (11j07h21mn)**

TAB. 7.4 – *Expérimentation de la caractérisation de la résonance stochastique sur une machine*

nombre de particules n, un ensemble de 1100 tâches est créé. Ces tâches sont affectées au fur et à mesure que les volontaires en demande. Un ensemble de 17 machines volontaires a participé à cette phase expérimentale. Le tableau 7.5 présente l'expérimentation sur les machines volontaires : Les résultats montrent que le temps d'exécution croit proportionnellement au nombre de particules

Nombre de particules	4	8	12	16	20	**Total**
Temps d'exécution	0h39mn	0h58mn	1h28mn	1h58mn	2h30mn	**7h35mn**
Nombre de volontaires	12	15	17	17	17	
Heure de calcul	6,35h 0j6h38mn	13,37h 0j13h22mn	21,76h 0j21h45mn	29,72h 1j5h43mn	37,5h 1j13h29mn	**108,69h 4j12h41mn**

TAB. 7.5 – *Expérimentation de la caractérisation de la résonance stochastique sur les machines volontaires*

comme sur une seule machine, mais en beaucoup plus réduit. Pour $n = 4$ particules par exemple, 12 volontaires ont participé au calcul et produit une quantité équivalente à 6h21mn de calcul en 39mn. 15 volontaires ont participé au calcul pour $n = 20$ particules et une qualité équivalente à 1j13h29mn de calcul a été produite en 2h30mn. Le temps d'exécution totale pour la caractérisation de toutes les valeurs de n (4, 8, 12, 16 et 20) particules est de 7h35mn sur notre système. Le tableau 7.6 présente

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}
3,93%	6,35%	6,20%	6,20%	6,35%	6,35%	6,20%	6,35%	4,08%	6,20%	6,35%	6,20%	6,35%	6,20%	4,54%	10,59%	01,51%

TAB. 7.6 – *Contribution de chaque volontaire dans le calcul pour $n = 20$*

le pourcentage de contribution de chaque volontaire ayant participé au calcul de la résonance pour $n = 20$.

7.4 Discussion et conclusion

Ce chapitre a porté sur l'expérimentation de la mise en place d'un système de calcul haute performance basé sur les machines des volontaires au sein de l'Université de Yaoundé I. La première phase de test s'est déroulée sur une dizaine de machines d'étudiants connectées à Internet et a montré qu'il est possible de mettre en place un système de calcul haute performance dans les environnements à ressources limitées. Face au succès de cette phase, nous avons lancé les expérimentations sur 74 machines. Nous avons également montré en faisant varier les tailles des matrices que le système passe facilement à l'échelle. Il suffit d'ajouter de nouvelles ressources de calcul pour avoir de meilleures performances.

En termes de coût, il convient de noter que nous n'avons dépensé ni en matériel ni en logiciels, puisque nous avons utilisé les logiciels libres. Cela montre et valide l'hypothèse selon laquelle il est possible d'agréger les ressources de calcul même dans le contexte des environnements pauvres en ressources pour former une plateforme de calcul haute performance.

Notons que les résultats positifs enregistrés lors des expérimentations ont été obtenus sous certaines conditions à prendre en considération :

- Stockage des données sur les machines des volontaires : pendant les expérimentations, plusieurs Mo de données ont été stockés sur les machines de volontaires. Les données stockées sur les machines des volontaires peuvent devenir volumineuses et les décourager de continuer à participer au système. Cependant, le fait que le volontaire puisse avoir besoin du système, stocker quelques données du système de calcul sur leur machine peut constituer pour eux un compromis.
- Prédiction de disponibilité : au cours de l'expérimentation sur les 74 machines, les machines ont été sélectionnées manuellement dans la liste des volontaires enregistrés. Cela peut devenir une tâche très difficile si l'on a des milliers de volontaires enregistrés dans le système. La solution à ce problème, même s'il n'est pas encore implémenté dans le système, est de prédire la disponibilité des ressources de calcul en fonction de l'historique de l'utilisation (voir chapitre 4 pour plus de détails).
- Durant l'expérimentation, nous avons considéré que les tâches sont indépendantes et qu'il n'y a pas d'échange de données entre elles. Ceci n'est pas vrai pour tous les calculs. D'autres types de calculs nécessitent une distribution des tâches sur les différentes machines, mais aussi une communication importante entre les machines. La synchronisation nécessaire entre les tâches s'exécutant sur les différentes machines provoque d'une part des temps d'inactivité pendant l'attente de messages, d'autre part des ralentissements dus aux temps de transferts des messages ainsi qu'aux latences du réseau, et enfin des risques de saturation du réseau qui provoqueraient un écroulement global du calcul. Il est donc important d'expérimenter le système avec ce type de calcul pour définir le spectre des applications qui peuvent y être exécutées.

- La transparence de l'utilisateur lors l'utilisation de sa machine n'a pas été évaluée puisque nous avons réquisitionné les machines pour une période de temps, pendant laquelle l'utilisateur a accepté de nous laisser toutes les ressources de sa machine. Dans un système réel, il faut pouvoir exécuter les tâches sur les machines des utilisateurs sans les perturber et sans qu'ils en soient conscients.
- La sécurité n'a pas été évaluée. Nous avons vu que la machine des utilisateurs doit être protégée contre d'éventuelles attaques par des volontaires malveillants et que les tâches doivent également être protégées. Durant l'expérimentation, nous n'avons pas considéré ces aspects.
- Prise en charge des coûts liés à l'Internet, en particulier pour les volontaires étudiants. Durant les expérimentations, les étudiants nous ont dit qu'ils pouvaient offrir leurs ressources de calcul. Mais, se posait le problème des coûts liés à la connexion Internet lors des transferts des calculs. Pour palier à ce problème, nous avons demandé aux étudiants qui pouvaient se rendre sur le campus de s'y rendre. Dans un véritable système impliquant des étudiants, la connexion Internet doit être prise en compte pour que le système ne leur coûte pas cher.

Comme le démontrent [28, 122, 123], nous pouvons aussi dire, malgré les limites des expérimentations, que les systèmes de calcul sur machines volontaires peuvent fournir une quantité importante de puissance de calcul. Plus particulièrement dans les environnements à ressources limitées, les machines volontaires peuvent sous certaines conditions être utilisées pour mettre en place une plateforme de calcul haute performance. Il s'agit donc une option de grande valeur pour les problèmes de calcul haute performance dans les environnements pauvres en ressources. Ce travail ouvre donc la porte à d'importantes possibilités dans les pays en voie de développement où les ressources de calcul sont généralement limitées aux machines des utilisateurs.

Chapitre Huit

Conclusion générale et perspectives

Les travaux menés dans le cadre de cette thèse visaient à étudier la possibilité de réaliser le calcul haute performance à faible coût dans les environnements pauvres en ressources tout en tenant compte des contraintes liées à ces environnements : manque de moyens financiers pour accéder aux plate-formes de calcul haute performance, instabilité de l'électricité et de l'Internet. À son terme, nous avons construit un système basé sur les machines des enseignants et des étudiants de l'université de Yaoundé I avec une capacité théorique de 12 PFlops. Le principal avantage de ce système est qu'il ne nécessite aucun investissement supplémentaire en matériel, mais repose sur les machines volontaires. Dans la suite, nous présentons dans un premier temps la démarche que nous avons utilisée (section 8.1) et les travaux que nous avons réalisés (section 8.2) pour construire ce système. Enfin, nous présentons à la section 8.3, les perspectives pour les travaux de recherche futurs.

8.1 Démarche utilisée

La démarche que nous avons utilisée pour atteindre nos objectifs s'articule autour des trois axes principaux :

- Dans un premier temps, une revue des systèmes de calcul haute performance a été faite, dans le but d'identifier ceux pouvant être accessibles dans les environnements à ressources limitées tant du point de vue des infrastructures que de l'Internet et de l'alimentation électrique. Cette revue nous a permis d'identifier les systèmes de calcul sur les machines dites volontaires comme une solution pertinente pour fournir du calcul parallèle dans les environnements présentant de telles contraintes et qui correspondent au contexte rencontré dans nos institutions universitaires et de recherche. Contrairement aux solutions basées sur l'accès distant aux plate-formes de calcul à l'étranger, cette approche a l'avantage de garantir une autonomie et la quasi-gratuité.
- Par la suite, nous avons proposé d'utiliser les méthodes basées sur l'apprentissage automatique pour la prédiction de la disponibilité des ressources de calcul. En effet, avec les systèmes de calcul sur machines volontaires, le fait que les utilisateurs du système partagent les nœuds de calcul avec leurs propriétaires et la nature instable de l'électricité et de l'Internet posent le problème de disponibilité des ressources de calcul. En effet, pour un calcul donné, trouver l'ensemble de machines pouvant l'exécuter dans un délai raisonnable est un défi. Il faut analyser les historiques de disponibilité des différentes machines pour trouver celles qui sont susceptibles d'être disponibles en continu pendant tout l'intervalle de temps estimé pour le calcul en question.
- Pour terminer, nous avons proposé une approche de mise en place des systèmes de calcul haute performance basée sur les machines volontaires prenant en compte les contraintes liées aux environnements à ressources limitées ; nous avons implémenté le système VC_UY1 au sein de l'Université de Yaoundé I et nous avons expérimenté le système sur le problème de la multiplication des matrices de grande taille et le problème de la caractérisation de la résonance stochastique.

8.2 Travaux réalisés

Dans cette thèse, nous avons étudié la possibilité d'exploiter les ressources de calcul provenant des particuliers et de coordonner ces ressources partagées afin d'utiliser leurs moments d'inactivité pour mettre en place un système de calcul haute performance. Les observations préliminaires (échanges avec des potentiels volontaires) nous ont permis de savoir qu'il était possible de mettre en place un tel système. Suite à ces constats, un ensemble de travaux a été effectué dans le but de mettre en place ce système.

- **Proposition d'une approche de prédiction de disponibilité des ressources basée sur le ML** : pour résoudre les problèmes de disponibilité des ressources dans un VCS, les méthodes de classification, plus spécifiquement la méthode naïve de Bayes et le perceptron multicouche et les méthodes de régression linéaires, comme la méthode LASSO sont généralement utilisé. ...Nous avons évalué ces prédicteurs et avons constaté qu'aucun d'eux n'est meilleur que les autres en termes de taux de prédiction. Pour tirer parti des avantages de chaque prédicteur, nous avons proposé un système de prédiction de disponibilité qui combine les prédicteurs : le perceptron multicouche et la régression de LASSO.
- **Expérimentation de la prédiction de disponibilité dans les systèmes de recommandation :** l'idée de prédiction de disponibilité des ressources a été étendue aux systèmes de recommandation pour déterminer quels produits peuvent être achetés dans une période de temps donnée. Cela a permis de construire un modèle de prédiction d'achat de produits à partir de l'historique des achats et de calculer un score représentant les chances que chaque produit soit acheté à un moment donné. Ce score a ensuite été utilisé pour pénaliser ou renforcer le résultat d'une recommandation. Les expérimentations menées sur trois jeux de données des systèmes de recommandation (MovieLens-2k, MovieLens-ls et Last.fm) ont montré que le système de prédiction d'achat de produits utilisé avec la métrique d'évaluation Hit-Ratio combiné à la recommandation Top-10 et au paramètre de calibrage de l'influence de la prévision de disponibilité beta $< 1/3$ améliore les résultats de la recommandation.
- **Proposition d'une approche de mise en place d'un VCS** : pour mettre en place les systèmes de calcul sur machines volontaires dans les environnements à ressources limitées, nous avons proposé une architecture hybride mettant en œuvre un serveur, un ensemble de volontaires et des clients. Par la suite, nous avons développé un logiciel nommé *vcSoftware* utilisé par le serveur, les volontaires et les clients.

  - **Le serveur** fait office de répertoire d'information. En effet, il permet de collecter et de stocker les données sur les volontaires (CPU, RAM, période de disponibilité, etc.), sur le système (puissance de chaque volontaire, puissance globale du système, etc.) afin de les restaurer au besoin. Ces informations permettent de connaître l'état du système et de l'ensemble des machines. Elles permettent également de trouver les machines ayant les caractéristiques les plus appropriées, en termes de disponibilité et de puissance, pour l'exécution d'un calcul donné. Le module *serverManager* du logiciel *vcSoftware* est ainsi utilisé à cet effet.
  - **Les machines volontaires** sont des nœuds de calcul qui récupèrent les tâches des clients, les exécutent et renvoient les résultats. Toute modification du profil de disponibilité ou de puissance de calcul au niveau des volontaires est signalée au serveur qui met à jour sa base de données. Lors d'un calcul, si l'utilisateur retourne à sa machine, l'état de la machine change, les calculs sont arrêtés et repris lorsque la machine revient dans l'état libre. Le module *volunteerManager* de *vcSoftware* est utilisé au niveau du volontaire.
  - **Les clients** sont des machines volontaires qui soumettent des calculs. Ils récupèrent la base de données du serveur (contenant le profil des volontaires) pour sélectionner pour un calcul donné, l'ensemble des machines pouvant l'exécuter jusqu'à la fin. Une fois que les

résultats des calculs intermédiaires sont récupérés, le client les combine pour obtenir la solution finale. Le module de *vcSoftware* qui est spécifique au client est *clientManager*.

- **Mise en place de VC_UY1** : après avoir défini notre approche, nous avons implémenté notre système. Ce système est composé d'un serveur, 164 volontaires, un logiciel permettant au serveur de collecter les informations sur les volontaires, aux volontaires d'exécuter les calculs et de rendre les résultats et aux clients d'envoyer les calculs, de collecter et fusionner les résultats. Il fournit une puissance de calcul théorique de 12 PFLOPS.
  - **Recrutement des volontaires** : les volontaires ont été recrutés parmi les enseignants et les étudiants de l'Université de Yaoundé I. A cet effet, un formulaire d'inscription des volontaires a été mis en place. Il y avait 164 volontaires dont 117 étudiants, 19 enseignants, 20 dans les salles de TP et 8 au CUTI enregistrés.
  - **Développement de vcSoftware** : le logiciel *vcSoftware* est la principale clé de notre système, il est utilisé par le serveur pour la collecte et le stockage de données, par les volontaires pour la collecte de données, l'exécution des calculs, le renvoie des résultats et les clients pour sélectionner l'ensemble des volontaires qui vont exécuter un calcul, distribuer un calcul, collecter et fusionner les résultats. Une première version de ce logiciel a été développée et utilisée pour la multiplication des matrices de grande taille et la caractérisation de la résonance stochastique.
- **Expérimentation de VC_UY1** :
  - **Phase de test** : pendant la phase de test, il s'agissait de vérifier la faisabilité d'un tel système. Pour cela, nous avons invité les étudiants de niveau master à fournir du temps libre de leur machine pour du calcul. Une dizaine de machines d'étudiants avec accès Internet ont été sélectionnées pour les tests.
  - **Expérimentation sur les grosses matrices** : la deuxième phase de l'expérimentation a consisté en une expérimentation d'envergure sur un plus grand nombre de machines et des matrices de tailles plus grandes. Au cours de ces expérimentations, 74 volontaires ont été sélectionnés et différentes tailles de matries ont été exécutées. L'exécution d'une matrice de taille 200000×200000 sur 68 machines a duré 91 heures et nous avons obtenu une puissance de calcul de 8 PFlops Cette expérimentation montre que les systèmes de calcul sur machines volontaires peuvent fournir une puissance de calcul importante dans les environnements à ressources limitées.
  - **Expérimentation du problème de la résonance stochastique** : la troisième phase de l'expérimentation a été consacré à la caractérisation de la résonance stochastique. Un sous-ensemble de 17 volontaires a été utilisé pour calculer l'énergie moyenne d'un ensemble de 4, 8, 12, 16 et 20 particules respectivement à partir de 50 énergies d'entrée par particule.

8.3 Perspectives

Les perspectives de cette thèse suivent cinq principales directions : finalisation du développement de **vcSoftware**, recrutement et motivation des volontaires, gestion des données stockées sur les machines des volontaires, prise en compte des calculs gourmands en données, pris en compte des calculs avec dépendances entre les tâches et la mise en place de la sécurité.

Développement logiciel : le logiciel *vcSoftware* que nous avons développé au cours cette thèse est un prototype qui dans sa version actuelle ne peut pas encore fonctionner en conditions réelles d'exploitation. En effet, ce prototype se limite à la collecte de données sur les machines volontaires, à la soumission des tâches de multiplication des matrices et de résonance stochastique aux volontaires

et la récupération des résultats. Son principal inconvénient est qu'il n'est pas facile d'installer pour un utilisateur non informaticien. Dans nos travaux futurs, *vcSoftware* devra :

- **Être facile à installer sur les machines volontaires** : un simple clic sur un lien doit permettre au volontaire de l'installer sur sa machine ;
- **Intégrer un système de prédiction** : la version actuelle de *vcSoftware* permet de soumettre les calculs et de récupérer les résultats. Dans les versions futures, nous intégrerons la fonctionnalité permettant d'utiliser le profil des machines (performance, disponibilité et utilisateur) et des tâches (besoin de puissance de calcul, de stockage et de communication) pour prédire pour un calcul donné, l'ensemble des machines pouvant l'exécuter jusqu'à la fin dans un temps raisonnable ;
- **Évaluation des volontaires** : un système d'évaluation et de classification des volontaires sera intégré dans le logiciel *vcSoftware*.

Recrutement et motivation des volontaires : la puissance d'un système de calcul sur machines volontaires réside sur le nombre de volontaires enregistrés et prêts à offrir leurs ressources de calcul pour le fonctionnement du système. Dans nos travaux futurs, une étude sur les mécanismes pour les encourager à s'inscrire et à contribuer davantage leurs ressources sera fait dans le but d'attirer même ceux au-delà de l'Université de Yaoundé I.

Gestion des données stockées sur les machines des volontaires : avec le système actuel, une copie de la base de données du serveur se trouve sur les machines des volontaires. Cependant, les données peuvent devenir volumineuses et décourager le volontaire de continuer à participer au système. Dans nos travaux futurs, un système permettant la sélection du minimum de données indispensable au volontaire sera mis en place.

Gestion des dépendances entre les tâches : dans cette thèse, nous n'avons pas considéré les calculs avec dépendances entre les tâches. Dans nos travaux futurs, nous considérerons la dépendance entre les tâches en tenant compte de la capacité de la machine, le coût de communication, la dépendance des données et des tâches, la synchronisation entre les tâches simultanément, afin de minimiser le temps d'exécution global.

Mise en place de la sécurité : le prototype présenté dans cette thèse ne prend pas compte la sécurité du volontaire, du serveur et des calculs. Dans nos travaux futurs, une étude sur les mécanismes de sécurité sera effectuée dans le but de définir un bon système de sécurité de notre système.

Bibliographie

[1] Mariza Ferro, Antonio R Mury, Laion F Manfroi, and Bruno Schlze. High performance computing evaluation a methodology based on scientific application requirements. *arXiv preprint arXiv:1412.1297*, 2014.

[2] High performance computing. In Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social Behavioral Sciences*, pages 6693 – 6697. Pergamon, Oxford, 2001.

[3] Matthieu Ospici. *Programming models and execution models for parallel and hybrid architectures. Application to physics simulations.* PhD thesis, Université de Grenoble, 2013.

[4] Brian TSAY. Tianhe-2 supercomputer: Less than meets the eye?, 2013.

[5] Jérôme Vienne. Prédiction de performances d'applications de calcul haute performance sur réseau infiniband. 07 2010.

[6] Ye Zhang. *Méthodes itératives hybrides asynchrones sur plateformes de calcul hétérogènes pour la résolution accélérée de grands systèmes linéaires.* PhD thesis, Université Lille1 - Sciences et Technologies, December 2009.

[7] Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Softw. Pract. Exper.*, 34(7):653–673, June 2004.

[8] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1989.

[9] Zhiwei Xu, Xuebin Chi, and Nong Xiao. High-performance computing environment: a review of twenty years of experiments in China. *National Science Review*, 3(1):36–48, 01 2016.

[10] Zahid Ansari, Asif Afzal, Moomin Muhiuddeen, and Sudarshan Nayak. Literature survey for the comparative study of various high performance computing techniques. *Int J Comput Trends Technol (IJCTT)*, 27(2):80–86, 2015.

[11] Ranjit Rajak. A comparative study: Taxonomy of high performance computing (hpc). *International Journal of Electrical and Computer Engineering (IJECE)*, 8(5):3388–3391, 2018.

[12] N. R. Adiga, G. Almasi, G. S. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, and M. Blumrich et al. An overview of the bluegene/l supercomputer. In *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60, Nov 2002.

[13] Erich Strohmaier, Hans Werner Meuer, Jack J. Dongarra, and Horst D. Simon. The TOP500 list and progress in high-performance computing. *IEEE Computer*, 48(11):42–49, 2015.

[14] Ezell Stephen and Atkinson Robert. *The Vital Importance of High-Performance Computing to U.S. Competitiveness.* INFORMATION TECHNOLOGY INNOVATION FOUNDATION, 2016.

[15] Chee Shin Yeo, Rajkumar Buyya, Hossein Pourreza, Rasit Eskicioglu, Peter Graham, and Frank Sommers. *Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers*, pages 521–551. Springer US, Boston, MA, 2006.

[16] Thomas Sterling. An introduction to pc clusters for high performance computing. *IJHPCA*, 15:92–101, 05 2001.

[17] I. Foster and C. Kesselman. *The Grid 2. Blueprint for a new computing infrastructure.* Morgan Kaufmann Publishers, San Francisco, USA, 2 edition, 2004.

[18] F. Magoules, T. M. H. Nguyen, and Lianbo Yu. Grid computing overview. In *Fundamentals of Grid Computing: Theory, Algorithms and Technologies*, Chapman & Hall/CRC Numerical Analysis & Scientific Computing, pages 1–28. CRC Press, USA, 2009.

[19] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[20] Abhishek Gupta, Laxmikant V Kale, Filippo Gioachin, Verdi March, Chun Hui Suen, Bu Sung Lee, Paolo Faraboschi, Richard Kaufmann, and Dejan Milojicic. The who, what, why and how of high performance computing applications in the cloud. *HP Laboratories Technical Report*, (49), 8 2013.

[21] Peter M. Kasson, Daniel L. Ensign, and Vijay S. Pande. Combining molecular dynamics with bayesian analysis to predict and evaluate ligand-binding mutations in influenza hemagglutinin. *Journal of the American Chemical Society*, 131(32):11338–11340, 2009. PMID: 19637916.

[22] Luis Francisco Gumaru Sarmenta. *Volunteer computing.* PhD thesis, Massachusetts Institute of Technology, 2001.

[23] Cristiano Tapparello, Colin Funai, Shurouq Hijazi, Abner Aquino, Bora Karaoglu, He Ba, Jiye Shi, and Wendi Heinzelman. Volunteer computing on mobile devices: State of the art and future research directions. In *Mobile Computing and Wireless Networks: Concepts, Methodologies, Tools, and Applications*, pages 2171–2198. IGI Global, 2016.

[24] Tessema M. Mengistu and Dunren Che. Survey and taxonomy of volunteer computing. *ACM Comput. Surv.*, 52(3), 2019.

[25] Muhammad Nouman Durrani and Jawwad A. Shamsi. Volunteer computing: requirements, challenges, and solutions. *J. Network and Computer Applications*, 39:369–380, 2014.

[26] W. Yu, Z. Wang, and X. Zhou. A volunteer computing in high performance environment based on availability model. In *2008 IFIP International Conference on Network and Parallel Computing*, pages 383–386, 2008.

[27] Sergio Ariel Salinas, Carlos García Garino, and Alejandro Zunino. PFS: A productivity forecasting system for desktop computers to improve grid applications performance in enterprise desktop grid. *Comput. Informatics*, 33(4):783–809, 2014.

[28] Erick Lavoie and Laurie Hendren. Personal volunteer computing. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, pages 240–246, New York, NY, USA, 2019. ACM.

[29] Oded Nov, David Anderson, and Ofer Arazy. Volunteer computing: A model of the factors determining contribution to community-based scientific research. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 741–750, New York, NY, USA, 2010. ACM.

[30] Bing Tang, Mingdong Tang, Gilles Fedak, and Haiwu He. Availability/network-aware mapreduce over the internet. *Inf. Sci.*, 379:94–111, 2017.

[31] Artur Andrzejak and Derrick Kondo. Modeling and optimizing availability of non-dedicated resources. *Desktop Grid Computing*, page 191, 2012.

[32] Jun Zhang and Chris Phillips. Job-scheduling via resource availability prediction for volunteer computational grids. *Int. J. Grid Util. Comput.*, 2(1):25–32, May 2011.

[33] Daniel Lázaro, Derrick Kondo, and Joan Manuel Marquès. Long-term availability prediction for groups of volunteer resources. *Journal of Parallel and Distributed Computing*, 72(2):281 – 296, 2012.

[34] Norm Matloff. *Programming on Parallel Machines: GPU, Multicore, Clusters and More*. 2012.

[35] National Research Council. *The Future of Computing Performance: Game Over or Next Level?* The National Academies Press, Washington, DC, 2011.

[36] Jack J. Dongarra. Performance of various computers using standard linear equations software in a fortran environment. 16(1):47–69, March 1988.

[37] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, and S.K. Weeratunga. The nas parallel benchmarks. 5(3):63–73, September 1991.

[38] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, 2016.

[39] Shurong Tian, Todd Takken, Vic Mahaney, Christopher Marroquin, Mark Schultz, Mark Hoffmeyer, Yuan Yao, Kevin O'Connell, Anil Yuksel, and Paul Coteus. Summit and sierra supercomputer cooling solutions. *IBM Journal of Research and Development*, 2019.

[40] Hai Jin, Rajkumar Buyya, and Mark Baker. Cluster computing tools, applications, and australian initiatives for low cost supercomputing. 11 2000.

[41] Rafiqul Zaman Khan and Md Firoj Ali. Current trends in parallel computing. *International Journal of Computer Applications*, 59(2):19–25, December 2012.

[42] Zvi Tannenbaum and Dean E Dauger. Cluster computing, June 25 2019. US Patent 10,333,768.

[43] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[44] Minakshi Tripathy and C.R. Tripathy. On a virtual shared memory cluster system with virtual machines. *International Journal of Computer and Electrical Engineering*, 3:754–761, 01 2011.

[45] Rajkumar Buyya. *High Performance Cluster Computing: Programming and Applications*. Prentice Hall PTR, USA, 1st edition, 1999.

[46] Sumit Srivastava, Pankaj Dadheech, and Mahender Kumar Beniwal. Load balancing using high performance computing cluster programming. *International Journal of Computer Science Issues (IJCSI)*, 8(1):62, 2011.

[47] Christian Bassek, Samuel Pierre, and Alejandro Quintero. Redundancy schemes for high availability computer clusters. *Journal of Computer Science*, 2, 01 2006.

[48] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[49] Leila Abidi, Christophe Cerin, and Mohamed Jemni. Desktop grid computing at the age of the web. In James J. (Jong Hyuk) Park, Hamid R. Arabnia, Cheonshik Kim, Weisong Shi, and Joon-Min Gil, editors, *Grid and Pervasive Computing*, pages 253–261, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[50] Harold Soh, Shazia Haque, Weili Liao, and Rajkumar Buyya. Grid programming models and environments. *Advanced Parallel and Distributed Computing*, page 141–173, 2006.

[51] Ian Foster. The grid: Computing without bounds. *Scientific American*, 288:78–85, 05 2003.

[52] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, 2009.

[53] Emanuel Coutinho, Gabriel Paillard, Ernesto Trajano de Lima, and Leonardo Moreira. An architecture proposal for high performance computing in cloud computing environments. 12 2015.

[54] Rocco Aversa, Beniamino Di Martino, Massimiliano Rak, Salvatore Venticinque, and Umberto Villano. Performance prediction for hpc on clouds. In *CloudCom 2011*, 2011.

[55] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[56] Aniruddha Marathe, Rachel Harris, David K. Lowenthal, Bronis R. de Supinski, Barry Rountree, Martin Schulz, and Xin Yuan. A comparative study of high-performance computing on the cloud. In *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '13, page 239–250, New York, NY, USA, 2013. Association for Computing Machinery.

[57] Top500. Zimbabwe to deploy 300-teraflop supercomputer from inspur, August 2018.

[58] Top500. Establishing the footprint of an african hpc centre creating and sustaining hpc in south africa, 2019.

[59] Bruno Richard, Nicolas Maillard, César A. F. De Rose, and Reynaldo Novaes. The i-cluster cloud: distributed management of idle resources for intense computing. *Parallel Comput.*, 31(8-9):813–838, 2005.

[60] Gilles Fedak. *Contributions to Desktop Grid Computing* . PhD thesis, Ecole Normale Supérieure de Lyon, May 2015.

[61] David P. Anderson. Volunteer computing: The ultimate cloud. *XRDS*, 16(3):7–10, March 2010.

[62] V. Posypkin M. Samtsevich A. AU Oganov A. R. Khrapov, N. Roizen. Volunteer computing for computational materials design. *Lobachevskii Journal of Mathematics*, 38:97–111, 2017.

[63] Sungjin Choi and Rajkumar Buyya. A taxonomy of desktop grids and its mapping to state-ofthe-art systems. 2007.

[64] Derrick Kondo, Andrew A. Chien, and Henri Casanova. Scheduling task parallel applications for rapid turnaround on enterprise desktop grids. *Journal of Grid Computing*, 5, 2007.

[65] Oleg Zaikin, Maxim Manzyuk, Stepan Kochemazov, Igor Bychkov, and Alexander Semenov. A volunteer-computing-based grid architecture incorporating idle resources of computational clusters. In Ivan Dimov, István Faragó, and Lubin Vulkov, editors, *Numerical Analysis and Its Applications*, pages 769–776, Cham, 2017. Springer International Publishing.

[66] Leila Abidi. *Révisiter les grilles de pcs avec des technologie du web et le cloud computing.* PhD thesis, Université Paris XIII, 2015.

[67] Bruno Richard and Philippe Augerat. I-cluster: Intense computing with untapped resources. 07 2002.

[68] Gordon Amoako, Nana Kwesi, and Peter Amoako-Yirenkyi. Volunteer computing: Application for african scientist. 01 2008.

[69] Michela Taufer, Chahm An, Andreas Kerstens, and Charles L. Brooks III. Predictor@home: A "protein structure prediction supercomputer' based on global computing. *IEEE Trans. Parallel Distributed Syst.*, 17(8):786–796, 2006.

[70] Barrera Harold, Rosero Edgar, and Cano Mario. *Desktop Grids and Volunteer Computing Systems*, chapter 7. IGI Global, 2012.

[71] C Adam-Bourdarios, R Bianchi, D Cameron, A Filipčič, G Isacchini, E Lançon, and W Wu and. Volunteer computing experience with ATLAS@home. *Journal of Physics: Conference Series*, 898:052009, oct 2017.

[72] Pawel Chorazyk, Mateusz Godzik, Kamil Pietak, Wojciech Turek, Marek Kisiel-Dorohinicki, and Aleksander Byrski. Lightweight volunteer computing platform using web workers. *Procedia Computer Science*, 108:948 – 957, 2017. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[73] David P. Anderson. Boinc: A platform for volunteer computing. *Journal of Grid Computing*, 18, 2020.

[74] Eric Korpela, David P. Anderson, Robert Bankay, Jeff Cobb, Andrew Howard, Matt Lebofsky, Andrew P. V. Siemion, Joshua Von Korff, and Dan Werthimer. Status of the uc-berkeley SETI efforts. *CoRR*, abs/1108.3134, 2011.

[75] F. Cappello, S. Djilali, Gilles Fedak, T. Herault, F. Magniette, V. Néri, and Oleg Lodygensky. Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21:417–437, 2005.

[76] S. Rubab, M. F. B. Hassan, A. K. B. Mahmood, and N. M. Shah. A review on resource availability prediction methods in volunteer grid computing. In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, pages 478–483, Nov 2014.

[77] Saddaf Rubab, Mohd Fadzil B Hassan, Ahmad Kamil B Mahmood, and Nasir Mehmood Shah. A review on resource availability prediction methods in volunteer grid computing. In *Control System, Computing and Engineering (ICCSCE), 2014 IEEE International Conference on*, pages 478–483. IEEE, 2014.

[78] James W. Mickens and Brian D. Noble. Exploiting availability prediction in distributed systems. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 6–6, Berkeley, CA, USA, 2006. USENIX Association.

[79] Farrukh Nadeem and Mahreen Nasir. Resource availability prediction in the grid: Taxonomy and review of state of the art. *International Journal of Engineering and Applied Computer Science (IJEACS)*, 01:46–53, 01 2017.

[80] Ranjita Bhagwan, Stefan Savage, and Geoffrey M. Voelker. *Understanding Availability*, pages 256–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[81] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, and Henri Casanova. Characterizing resource availability in enterprise desktop grids. *Future Generation Comp. Syst.*, 23(7):888–903, 2007.

[82] Brent Rood and Michael J Lewis. Resource availability prediction for improved grid scheduling. In *2008 IEEE Fourth International Conference on eScience*, pages 711–718. IEEE, 2008.

[83] Bing Tang, Mingdong Tang, Gilles Fedak, and Haiwu He. Availability/network-aware mapreduce over the internet. *Information Sciences*, 379:94 – 111, 2017.

[84] Soodeh Peyvandi, Rohiza Ahmad, and Mariam Zakaria. Scoring model for availability of volatile hosts in volunteer computing environment. 70, 01 2014.

[85] Lázaro Iglesias Daniel. *A Middleware for Service Deployment in Contributory Computing Systems*. PhD thesis, Universitat Oberta de Catalunya, 2011.

[86] Kondo Derrick. *Scheduling Task Parallel Applications For Rapid Turnaround on Desktop Grids*. PhD thesis, University of California, San Diego, 2005.

[87] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[88] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques, third edition, 2012.

[89] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2008.

[90] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[91] Mustafizur Rahman, Md. Rafiul Hassan, and Rajkumar Buyya. Jaccard index based availability prediction in enterprise grids. In Peter M. A. Sloot, G. Dick van Albada, and Jack Dongarra, editors, *ICCS*, volume 1 of *Procedia Computer Science*, pages 2707–2716. Elsevier, 2010.

[92] Bahman Javadi, Kenan Matawie, and David P Anderson. Modeling and analysis of resources availability in volunteer computing systems. In *2013 IEEE 32nd International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE, 2013.

[93] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.

[94] Marko Balabanovic and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–73, 1997.

[95] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. 2007.

[96] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[97] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 213–220. ACM, 2013.

[98] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 265–268. ACM, 2009.

[99] Linas Baltrunas and Xavier Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS'09)*, pages 25–30. Citeseer, 2009.

[100] Armel Jacques Nzekon Nzeko'o, Maurice Tchuente, and Matthieu Latapy. Time weight content-based extensions of temporal graphs for personalized recommendation. In *WEBIST 2017-13th International Conference on Web Information Systems and Technologies*, 2017.

[101] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.

[102] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(Dec):2935–2962, 2009.

[103] Arnaud Millien. *Access to electricity and economic development : determinants of favorable impacts for households*. Theses, Université Panthéon-Sorbonne - Paris I, September 2019.

[104] B.S. Diboma and T. Tamo Tatietse. Power interruption costs to industries in Cameroon. *Energy Policy*, 62(C):582–592, 2013.

[105] Giuseppe Aceto, Alessio Botta, Pietro Marchetta, Valerio Persico, and Antonio Pescapè. A comprehensive survey on internet outages. *Journal of Network and Computer Applications*, 113, 03 2018.

[106] Thomas Messi Nguélé, Maurice Tchuente, and Jean-François Méhaut. Using complex-network properties for efficient graph analysis. In *International Conference on Parallel Computing, ParCo 2017*, volume 32, pages 413–422. IOS Press Ebooks, 2017.

[107] Florentin Flambeau Jiechieu Kameni and Norbert Tsopzé. Simplifying the explanation of deep neural networks with sufficient and necessary feature-sets: case of text classification. *CoRR*, abs/2010.03724, 2020.

[108] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[109] Maarten Van Steen and Andrew S Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[110] Mike Litzkow, Miron Livny, and Matt Mutka. Condor - a hunter of idle workstations. volume 8, pages 104 – 111, 07 1988.

[111] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: An experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, November 2002.

[112] Chapter 13 - managing containerized applications. In Rick Sturm, Carol Pollard, and Julie Craig, editors, *Application Performance Management (APM) in the Digital Enterprise*, pages 177 – 185. Morgan Kaufmann, Boston, 2017.

[113] Qingshan Luo and John B. Drake. A scalable parallel strassen's matrix multiplication algorithm for distributed-memory computers. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 221–226, New York, NY, USA, 1995. ACM.

[114] Chandan Misra, Sourangshu Bhattacharya, and Soumya K. Ghosh. Stark: Fast and scalable strassen's matrix multiplication using apache spark, 2018.

[115] M. Son and K. Lee. Distributed matrix multiplication performance estimator for machine learning jobs in cloud computing. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 638–645, July 2018.

[116] Zhen Xie, Guangming Tan, Weifeng Liu, and Ninghui Sun. Ia-spgemm: An input-aware auto-tuning framework for parallel sparse matrix-matrix multiplication. In *Proceedings of the ACM International Conference on Supercomputing*, ICS '19, pages 94–105, New York, NY, USA, 2019. ACM.

[117] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[118] Wei Cheng, Xuemei Xu, Ding Yipeng, and Kehui Sun. Stochastic resonance in a single-well potential and its application in rolling bearing fault diagnosis. *Review of Scientific Instruments*, 91:064701, 06 2020.

[119] Jesús Casado-Pascual, José Gómez-Ordóñez, and Manuel Morillo. Stochastic resonance: Theory and numerics. *Chaos (Woodbury, N.Y.)*, 15:26115, 07 2005.

[120] Y. J. Wadop Ngouongo, G. Djuidjé Kenmoé, and T. C. Kofané. Effect of coupling on stochastic resonance and stochastic antiresonance processes in a unidirectionally N-coupled systems in periodic sinusoidal potential. *Physica A Statistical Mechanics and its Applications*, 472:25–31, April 2017.

[121] N. Jeremy Kasdin. Runge-kutta algorithm for the numerical integration of stochastic differential equations. *Journal of Guidance, Control, and Dynamics*, 18(1):114–120, 1995.

[122] Erick Lavoie, Laurie Hendren, Frederic Desprez, and Miguel Correia. Pando: Personal volunteer computing in browsers, 2018.

[123] Travis Desell. Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters. pages 19–28, 10 2017.

Table des figures

Liste des tableaux

Liste des Abréviations

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BOINC	Berkely Open Infrastructure for Network Computing
CCA	Common Component Architecture
CERN	Conseil européen pour la recherche nucléaire
CHPC	Center of High Performance Computing
CPU	Central Processing Unit
CUTI	Centre Universitaire des Technologies de l'Information
E/S	Entrée/Sortie
EC2	Elastic Compute Cloud
FAQ	Frequently Asked Questions
FCFS	First Come First Served
Flops	Floting Point Operation per Second
GFLOPS	gigaFLOPS
GIMPS	Great Internet Mersenne Prime Search
Go	Giga Octet
GPU	Graphics Processing Unit
HPC	High Performance Computing
HPCaaS	HPC-as-a-service
HPF	High-Performance Fortran
IaaS	Infrastructure-as-a-Service
IBM	International Business Machines Corporation
ICVolunteers	International Conference Volunteers
IPython	Interactive Python
Lasso	Least Absolute Shrinkage and Selection Operator
LHC	Large Hadron Collider
MFLOPS	megaFLOPS
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPI	Message Passing Interface
NEC	Nippon Electric Company
openMP	Open Multi-Processing
PaaS	Platform-as-a-Service
pil	prediction interval length

POD	Penguin On Demand
PVM	Parallel Virtual Machine
RAM	Random-Access Memory
SaaS	Software-as-a-Service
SETI	Search for Extra-Terrestrial Intelligence
TCP/IP	Transmission Control Protocol/Internet Protocol
TFLOPS	teraFLOPS
til	training interval length
VC_UY1	Volunteer Computing at the University of Yaounde I
VCS	Volunteer Computing Systems
WCG	World Community Grid
ZCHPC	Zimbabwe Center for High Performance

High Performance Computing in Resource Poor Settings: An Approach based on Volunteer Computing

Adamou Hamza[1], Azanzi Jiomekong[2]
University of Yaounde I, Faculty of Sciences, Yaounde, Cameroon;
IRD, Sorbonne Université, UMMISCO, F-93143, Bondy, France

***Abstract*—High Performance Computing (HPC) systems aim to solve complex computing problems (in a short amount of time) that are either too large for standard computers or would take too long. They are used to solve computational problems in many fields such as medical science (for drug discovery, breast cancer detection in images, etc.), climate science, physics, mathematical science, etc. Existing solutions such as HPC Supercomputer, HPC Cluster, HPC Cloud or HPC Grid are not adapted for resource poor settings (mainly for developing countries) because their fees are generally beyond the funding (particularly for academics) and the administrative complexity to access to HPC Grid creates a higher barrier. This paper presents an approach allowing to build a Volunteer Computing system for HPC in resource poor settings. This solution does not require any additional investment in hardware, but relies instead on voluntary machines already owned by the private users. The experiment has been made on the mathematical problem of solving the matrices multiplication using Volunteer Computing system. Given the success of this experiment, the enrollment of other volunteers has already started. The goal being to create a powerful Volunteer Computing system with the maximum number of computers.**

Keywords*—*Volunteer computing; resource poor settings; high performance computing; matrix multiplication

I. Introduction

High performance computing [1], [2], [3], [4], [5] is really important in most scientific and industrial sectors. It helps scientists gain valuable insights to boost innovation and discovery in almost all areas of science ranging from life sciences [6], [7], [8] to quantum mechanic [9] and large scale data mining [10], [11]. In the industrial sector, high performance computing gives companies a significant competitive advantage in reducing the costs of development cycles and in producing higher quality products and services [12]. Societal challenges, including preventing and managing natural disasters, early detection and treatment of disease, and forecasting climate evolution require very high computing power [13].

In the past, dedicated supercomputers [14], [15] powerful machines made up of a large number of processors, were only used to build a HPC system. Despite the processing capacity and the speed of calculation of these computers, they do not offer the commodity price advantage, especially for small businesses and academics [13], [16]. Then, the following solutions were proposed : HPC Clusters [17], [18], [19], HPC Cloud [20], [21], and HPC Grid [19], [22]. HPC Cluster is a system composed of computers connected to a local area network, while HPC Cloud is a system involving computers connected in a private or public network. Finally, HPC Grid is composed of computers connected in wide-area networks such as Internet.

The proposed solutions cannot be applied in low resource areas because of its costs. The HPC Cluster and the HPC Cloud require financial resources beyond the reach of most institutions in developing countries. For the HPC Grid, administrative complexity in obtaining the necessary permissions to use this system can be a higher barrier [21].

In developed countries, HPC have provided computing resources to projects at a huge scale [23], often resulting in major scientific discoveries and invention whether at the universities or businesses level. However, in resource poor or limited settings, many academics and small business do not always have enough resources to access to the HPC. Resource poor or constrained settings are defined as a local where the capability to provide HPC is limited to basic critical resources, including desktops and laptops. Resource poor settings can be stratified by no resources and limited resources. In resource limited settings, people use their personal computers for computation. This article focuses on the limited resources category. These include scientists in research teams with significant computational needs, individuals in developing countries with no access to other alternatives.

It should be noted that in many developing countries, the wide range of commercial centers of computers has made available many devices. For instance, in the Department of Computer Science at the University of Yaounde I in Cameroon, there are about 1,000 computers owned by lecturers and students; the Internet connection is widespread in Cameroon. These computers are idle most of the time (e.g., when students are in class or sleeping at night, etc.), this therefore constitutes a source of computing power available and largely reusable. If their owners are willing to actively lend CPU time and memory, these devices can be used as distributed computing infrastructure at no cost. This is called a Volunteer Computing (VC) system [24], [25], [26], [27]. A HPC Volunteer Computing system is obtained through community engagement by setting up a system of volunteer machines. Their goal is similar to HPC Grid, which is to gather distributed computing resources and federate them to solve large computational problems. The difference between these two systems is that the resources come from non-dedicated computers, underutilized and controlled by their owners (volunteers). This approach requires no additional hardware investment, but relies on

devices already owned by users and their communities.

This article presents an approach to build a Volunteer Computing system for HPC in resource poor settings. Initially, the Section II describes the main solutions to HPC problems. Then, the Sections III and IV follow with our solution and the experience of this solution respectively. Finally, the Section V concludes and opens future directions.

II. High Performance Computing (HPC)

High Performance Computing uses the principle of parallel computing to address the high computing requirements of applications by dividing them into smaller ones that can be processed simultaneously on different computing units. These computing units can reside on the same computer (Supercomputer) or on multiple computers connected by a network forming HPC Cluster, HPC Cloud, HPC Grid and HPC Volunteer Computing. These solutions are presented in this order in the rest of this section.

A. HPC Supercomputers

One of the best known type of HPC solution is the Supercomputer [14], [15], [28]. A Supercomputer contains hundreds, thousands or even millions of computing units forming a massively paralleled processor organized in a network of processors [29], [30]. For instance, the *Summit - IBM Power System AC922*[1] contains $2,414,592$ cores. These processors work together to solve large computational problems as efficiently and quickly as possible.

Supercomputers allow us to obtain a great computing power. However, with the entry fees [13], [15], acquiring a Supercomputer is almost impossible for scientists and engineers working in resource poor settings. Because of this limitation, other HPC solutions that do not require a fully dedicated computer have been developed. These solutions are HPC Cluster, Cloud, Grid and Volunteer Computing.

B. HPC Cluster

A HPC cluster is a computing system in which independent computers are connected by a high performance local area network in order to solve complex problems [18], [19], [31], [32]. Each machine in the HPC cluster is a complete computer consisting of one or more CPUs or cores, memory, disk drives and network interfaces. HPC cluster is generally owned by a single administrative entity. The software used to manage clusters give users the illusion that they are with a single large computer when in reality the cluster may consists of hundreds or thousands of individual machines [33]. A cluster is much more cost-effective than a single supercomputer of comparable speed [13], [16].

An example of a computer cluster is the dedicated physical cluster at HP Labs Singapore (HPLS)[2]. This cluster is connected to a Gigabit Ethernet network on a single switch. Each server, with 2 CPU sockets (populated with a 6 core CPU), results in twelve physical cores per machine.

[1]https://www.top500.org/system/179397

[2]https://xrds.acm.org/article.cfm?aid=2000789

C. HPC Cloud

A High Performance Computing Cloud [20], [21], [34], [35] is an on-demand availability of computing power, without direct active management by the user. It provides dynamic and scalable computing power through resources organized in data centers distributed around the world. By purchasing on-the-go and not as an asset, HPC Cloud delivers consistent, scalable results, minimizing the initial costs of computing infrastructure. A cloud can be private (operating for a single organization) or public (open for public use). Since 2016, many IT companies such as Amazon Web Services (AWS)[3], Microsoft Azure cloud[4], Google Cloud Platform[5] have offered HPC Cloud.

The advantage of the HPC cloud over other types of HPC is that it allows the user to immediately access computing resources without the approval of an allocation committee and the service can be provided without human interaction with the service provider. Software can be used without the need to purchase a license or install it, and users do not need to have strong software/infrastructure management skills [21], [36]. However, in the context of resource poor settings, the main drawback of HPC Cloud is that it relies on dedicated hardware managed centrally, which implies a minimum cost which can be high for academics and small businesses [21].

D. HPC Grid

A Grid consists of many computing resources (from multiple administrative domains) connected to a network (e.g., The Internet) working together to solve large problems requiring HPC. The whole system is called a HPC Grid [19], [22], [37]. The HPC Grid differs from the HPC Cluster or Cloud in that it uses many computers, but with a much more distributed nature. Some HPC Grids span the world while others are located within a single organization.

The HPC Grid capabilities are generally managed by a precise organization and computational resources are provided by various supporting institutions, such as companies, research groups, laboratories, and universities. Large Grid computer facilities are often used by a large number of users to solve intensive scientific, mathematical, and academic problems. However, the administrative complexity (obtaining necessary authorizations) allowing the public to access a HPC Grid is a real barrier for academics and small businesses. [25].

An example of Grid is the French Grid'5000[6]. This Grid is distributed over 8 sites, 31 clusters and 12,328 cores. Each site hosts a cluster and all sites are connected by high speed network. It aims to provide a highly reconfigurable, controllable and monitorable experimental platform for research in large-scale parallel and distributed systems.

E. HPC Volunteer Computing

In HPC Volunteer Computing (VC) [24], [25], [27], [38], [39], computer users/owners, who are members of the general public contribute their computing resources to solve HPC

[3]https://aws.amazon.com

[4]https://azure.microsoft.com

[5]https://cloud.google.com

[6]https://www.grid5000.fr/w/Grid5000:Home

problems. It is based on two pillars: the first is the allocation and management of large computing tasks; the second is the participation of a large number of individuals volunteer who offer their computing resources to a project. Compared to HPC Cluster, HPC Cloud and HPC Grid, HPC VC removes financial and administrative barriers. The costs are supported by volunteers, which cover the acquisition, operation, and maintenance of the computing devices.

HPC Volunteer Computing has produced many remarkable scientific results over the last decade. The most popular Volunteer Computing systems are: SETI@home[7][8] [40] for searching the extraterrestrial intelligent life and Folding@home[9] [41] for statistical calculations of molecular dynamics trajectories for models of biological systems. The Folding@home project is a good example of how important scientific results can be produced with VC for affordable problems for other HPC schemes. For instance, in 2008, it was used to study mutations of influenza hemagglutinin [23].

The main challenges of the Volunteer Computing approach is the capabilities of personal devices, the need to encourage and maintain volunteer engagement, and the automatic management of volunteer unreliability [42], [43]. Despite the previous challenges, Volunteer Computing is the solution for HPC in resource poor settings. In fact, in many developing countries, the wide range of commercial centers of computers has made available a lot of devices. These devices spend a lot of time without being used. They can then be used free of charge as a distributed computing infrastructure. It requires no investment in additional hardware, rather relies on devices that generally belong to users and their community, and favours simple tools that can be implemented part-time by a single developer. Section III, will be concerned with a methodology for deploying and using HPC to improve computing in resource poor settings while in Section IV, it will be shown in experiments that this solution can solve the problem of HPC in resource poor settings.

III. An approach based on Volunteer Computing for HPC in resource poor settings

The lack of HPC resources is a challenge for scientists, engineers and businesses in resource poor settings. However many countries have an Internet or local network and many computers are owned by individuals. For example, in the computer science department of the University of Yaounde I in Cameroon, there are around 1,000 computers belonging to students and lecturers. Generally, these computers are not used full time. For example, students spend a lot of time a week attending classes, sleeping, or taking breaks. These computers can be used during their idle time to build a platform for Volunteer Computing. Considering the resource poor settings constraints, this section presents an approach (summarized in Fig. 1) for HPC in these environments. This approach is divided into three main activities: management activity (Section III-A), processing activity (Section III-B) and support activity (Section III-C).

[7]https://setiathome.berkeley.edu/
[8]https://www.seti.org/
[9]https://foldingathome.org/

A. The Management Activity

The management activity uses information about the daily work of volunteers (e.g., the performance of the system and of each volunteer) to supervise the system and increase efficiency. This is the essential key to the success of the Volunteer Computing system. For instance, information about the volunteers, the types of tasks performed (e.g. matrix multiplication, polynomial multiplication), the name of each task, the duration of execution of the tasks, the success or failure of the execution of the tasks and the dates of their execution, will allow us to consider the resource poor settings context. The management activity involves the following activities: design of the Volunteer Computing system (Section III-A1), planning (Section III-A2), coordination (Section III-A3), staffing (Section III-A4), motivation (Section III-A5), control (Section III-A6) and quality assurance (Section III-A7).

1) Design of the Volunteer Computing System: The first and main activity of this approach is to set up the Volunteer Computing system. Resource poor settings generally suffer from power outages and malfunctions in Internet connectivity. In our approach, potential volunteers are firstly users who are often connected to the Internet, secondly those who need the system later and finally friends, family members or communities. Each volunteer has information about the others. A server is designed and contains the most up-to-date information about the volunteers, sent by the latter. All other local information for volunteers is updated by the server. If the server crashes (due to a power outage or a failure), the election algorithm [44], [45] is used to choose the volunteer that will replace the server by waiting that the problem is solved. In our case, it will consist of choosing the volunteer who connects the most to the Internet. Overall, the proposed VC is defined by the equation 1, where:

- S is the server. In our approach, the server is used to centralize all the volunteer information in the system in order to restore it whenever a volunteer needs it. Thus, the server will be used to store all the information on the system, the performance of the system, and of each volunteer in the system, the logs on the computation performed, the type of tasks already completed and its performances, tasks in progress or waiting. These information are essential for efficient management of the system. The server will use the feedback to help volunteers estimate the execution time each time a volunteer wants to perform a task by the system.
- V_i a volunteer computer. It receives tasks, performs them and returns the result. It can submit tasks to other volunteers, retrieve the results and merge to obtain the final result. It informs the server of all the computation activities carried out, in progress or pending. It receives updated information from other volunteers from the server.
- P_i the profile of the volunteer V_i. The profile P_i contains information on the elements that can be used to determine its computing capacity. These are: CPU speed (cpu_speed), number of cores (num_core), memory size (mem_size), disk size (disk_size), operating system name (os_name), its location and its

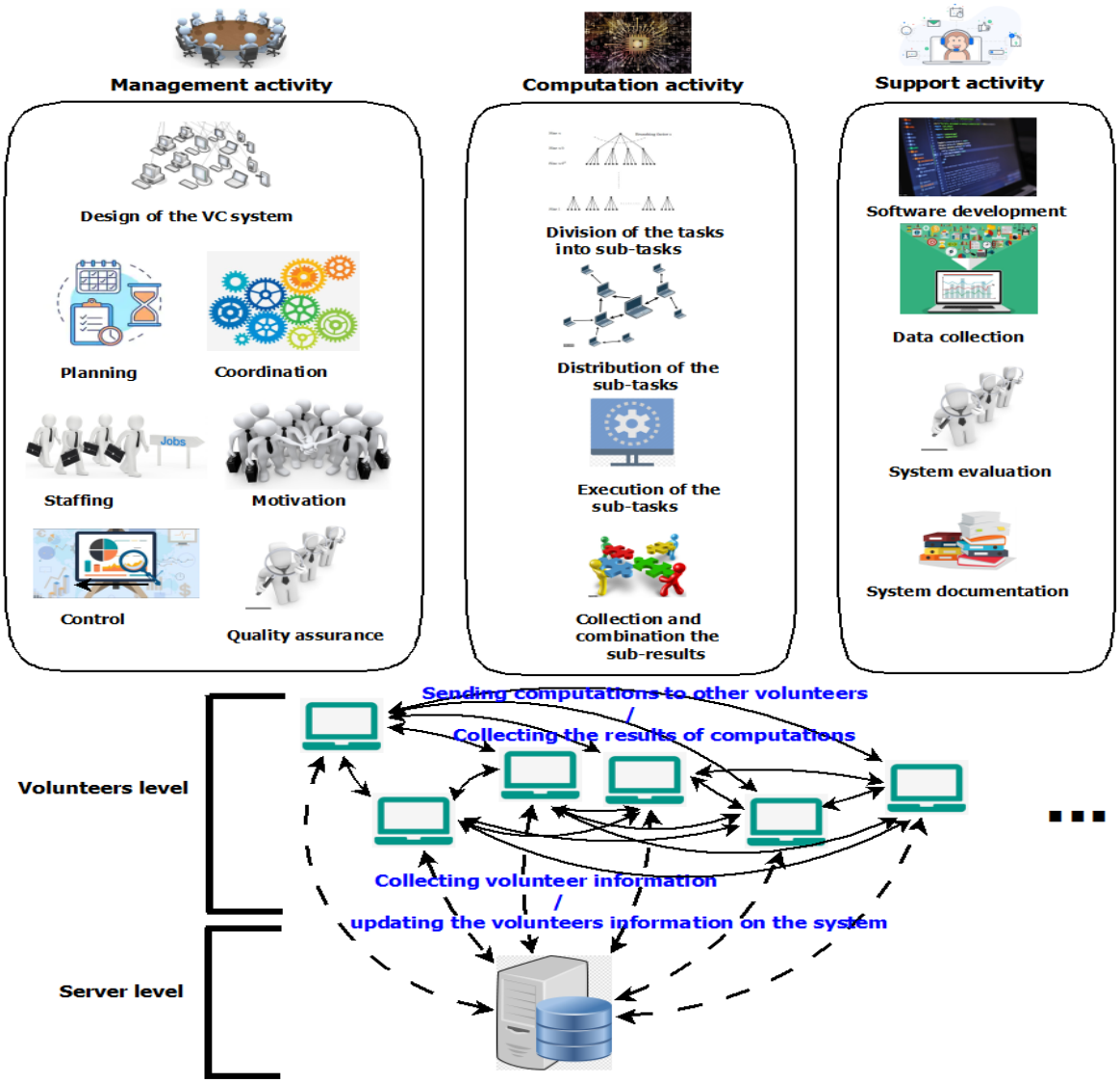

Fig. 1. The Volunteer Computing approach

availability (avail). These information will be used by each volunteer to identify to which volunteers they can send their computations to, what types of computations to send and when.

$$VC = [S, (V_1, P_1), ..., (V_i, P_i)],$$
$$P_i = [cpu\_speed, num\_core, mem\_size, \quad (1)$$
$$disk\_size, os\_name, location, avail].$$

A software named VCSoftware is used in the system. It is composed of the volunteersManager module and the serverManager module:

- volunteersManager module: The volunteerManager module is used at the volunteer level. A new volunteer signs up to the system by filing out their profile in a form, installing the VCSoftware and activating the volunteerManager on their machine. The volunteersManager is used to collect information about each volunteer and send it to the server, receives tasks from other volunteers, performs and returns the result, allocate tasks to other volunteers, collect and merge the results and send feedback of computation to the server. The volunteer contribution level is determined by the setting of their profile. The participant can choose to contribute permanently or only when the computer is idle, or can decide whether to contribute or not when the computer is running on battery/inverter. If the server is down, this software informs the computer owner and all other volunteers, the goal being to design the new server.
- serverManager module: The serverManager module is used at the server level. It allows to periodically collect information on the profile of each volunteer and to record a log of all the tasks performed by the system.

This information will allow to recommend a profile to a volunteer (e.g. computation time taking into account the idle period of the computer), to know the performance of each volunteer and the performance of the whole system. When the server is down, the volunteer elected as server while waiting for the problem to be solved activates this module.

2) Planning: Planning activity is the foundation of management. It refers of determining the future course of action towards the desired objective of the system. The planning activity must anticipate and precede all the other functions of the management activity and permit to meet the challenges of environmental changes (e.g., arrival and departure of volunteers, server failure, power/Internet outage etc.). In our VC approach the planning activity is at two levels:

- At the server level, planning involves continuous assessment of the strengths and weaknesses of the VC system using information about volunteers and the results of computation; identification of the actions assigned to a task already performed by the system. It requires all the information collected on the volunteers during the support activity (see Section III-C). This information will allow the server to know the idle time of each volunteer computer, when each one connects to the Internet, to evaluate the performance of each volunteer and the performance of the whole system. For example, information about a task already executed by the system can be used to efficiently execute it the next time the same type of task is submitted.
- At the volunteer level, planning requires the information from other volunteers. Stored on the server, this information is sent to each volunteers when they connect to the system. This information will be used to decide which volunteer to send a task taking into account the computing power offered by each volunteer and the period of connection to the system: which task to send, when to send the task, to whom the task can be sent, the duration of each task sent to a volunteer and the duration of the whole task. At the volunteer level, good planning will help to effectively address the challenges of environmental change (e.g., when a volunteer's computer cannot complete a sub-task).

3) Coordination: Like the planning activity, the coordination activity is done at two levels:

- At the server level, the server identifies the volunteers available for the computation, the computation power they offer and the duration for which they remain connected to the Internet. It also stores information about the different types of tasks already performed. This information will be used to predict the time and resources required to complete a task. Any change in volunteer information (computing power) should result in organizational changes. Information about leaving/joining the system by a volunteer must be considered.
- At the volunteer level, the coordination activity involves organizing the sub-tasks to be performed, as well as the time and resources necessary to carry them out. It will use the information obtained from the other volunteers to identify the resources needed to achieve a given task.

4) Staffing: The staffing activity includes recruiting good volunteers, selecting a group of volunteers to perform a task, and evaluating the volunteers registered in the system. Since volunteers registered in the system are not paid, the manager must be careful during their recruitment. In a resource poor settings, we recommend starting with people who need High Performance Computing and who do not have enough financial resources; and encourage them to invite members of their community to register. Information on volunteer profiles will be used to identify the good volunteers profiles.

5) Motivation: The motivation activity consists of attracting volunteers to contribute to the system and those in the system to increase their contributions. Since volunteers are not paid, a motivational environment must be created. In our case, the possibility of having access to a High Performance Computing is a great motivation for students and researchers. Data collected on the use of the system by other volunteers must be made available to the public, mainly students, engineers and researcher in order to encourage them to join the system. Performance data desired by the system will be made available to volunteers enrolled in the system to encourage them to participate and to encourage members of their communities to participate.

6) Control: The control activity aims to guarantee that scheduled tasks are completed as planned. During the control activity, the performance evaluation of each volunteer and the whole system is made in order to identify weaknesses and strengths. Planning is the basis of control. It focuses on the tasks that are performed and the results of those tasks. It plays an important role in ensuring the efficiency and effectiveness of the VC system. The information collected during the use of the VC system will help to measure the performance of each volunteer but also to identify gaps. At the server level, the comparison between planned performance and actual performance is analyzed to know if there are deviations, and the reasons of the deviations are analyzed. At the volunteer level, each result of a sub-task collected from volunteers is used to evaluate it. Real time information will allow quick control, which will reduce the costs of planning errors.

7) Quality assurance: The quality assurance activity guarantees that the quality of each computation is satisfactory (took the expected time, returned the expected results).

B. Computation Activity

To perform a task with the VC, that task must be divided into sub-tasks and each sub-task sent to volunteers. After the execution by the volunteers, the results and the results logs are collected by the volunteer who initiated the computation. Globally, computation activity involves: dividing the task into sub-tasks, distributing of sub-tasks to volunteers, performing of sub-tasks by volunteers, collecting and combining the results.

1) Division of the task into sub-tasks: This step consists of dividing the task into smaller and independent sub-tasks (preferably atomic sub-tasks). Consider T as a given task to

be performed by a volunteer. Then, $T = t_1, t_2, ..., t_n$ where n is the number of independent tasks that compose the task T. Since t_i are independent, they can be executed in parallel.

2) Distribution of the sub-tasks amongst volunteers: Given the data obtained from the server on the system (computing power offered by each volunteer, time to connect to the Internet of each volunteer, etc.), the volunteersManager will identify the volunteers that can participate in the computation. These are the computers most likely to be available until the end of a given sub-task. Subsequently, the computation will be sent to these volunteers when they connect to the system. Depending on the number of volunteers available and the computation time they provide, many sub-tasks can be sent to a volunteer.

The volunteer computer that sent the tasks to others will send a log file to the server to inform it that a job has started. The server will broadcast this information to all other volunteers. If a machine starts a new task, it must consider the existing tasks running in the system.

3) Execution of the sub-tasks: The volunteersManager on the volunteer machine that receives a task will start within the period specified in the volunteer profile. If many tasks have been assigned to a volunteer, the volunteersManager will perform the oldest first. At the end, a log records information on the execution time and the execution status (finished or failed).

4) Collection of the sub-results: After completion, the volunteersManager collects the results and the result logs. If some computations failed, more efficient volunteers are identified and these computations are sent to them. At the end of the computation, the results logs are sent to the server.

5) Combination of the sub-results: The last activity of the computation activity is the combination of the sub-results obtained from the volunteers.

C. Support Activity

The support activity involves the series of activities performed at the same time as the management and the computation activities. It aims to facilitate management and computation by providing all the needed tools and information. The VCSoftware software is developed during this activity. Overall, this activity includes the software development (section III-C1), data collection on the system and each volunteer computer (section III-C2), system evaluation (section III-C3) and system documentation (section III-C4).

1) Software development: The software development activity involves the development/updating of the VCSoftware that will be used by the server and the volunteers. This tool is composed of two main modules: the serverManager module and the volunteersManager module.

The serverManager will help acquire data sent by volunteers to the server. These data are those provided by the volunteer during registration, but also other information on the idle time of the volunteer computer and the time of connection to the Internet. The Information collected from the volunteers will allow the server to: make suggestions of profiles to the volunteers. The volunteer can use the suggestion or not. For example, during the holidays, the idle period is not necessarily the same as during the working period. Then, during the holidays, the server can suggest to volunteers to update their profile.

The volunteersManager is the module which, on the volunteer side, will: periodically collect information on the volunteer and send it to the server; allow identification of volunteer computers that can perform a given task, send tasks to volunteers and collect the results; reception of tasks and their scheduling according to other tasks already received; and sending logs on the execution of the tasks and the performance of each volunteers.

2) Data collection: The data collection activity is done both at the server and at the volunteer level. At the server level, the data collected is used to evaluate the performance of the system and each volunteer. The performance of the whole system is calculated based on the number of volunteers registered in the system, the tasks performed by the volunteers, and when the volunteers will connect to the Internet.

At the volunteer computer level, data is collected in two cases: firstly, the volunteer fills out a registration form in which information about the device and its availability for computation is provided. Then, an automatic data collection takes place (using volunteersManager module). This can be done periodically (hourly or daily) and will concern the idle time of the computer and the time when the computer is connected to the Internet. This data is used to: suggest profile updates to the volunteer and evaluate each volunteer. Each time the server receives updated information from a volunteer, this information is aggregated with existing information and forwarded to other volunteers.

3) System evaluation: The system evaluation involves the evaluation of the server and each volunteers. Indeed data collected during the computations will allow the server to know if the system is efficient and if it provides relevant computations. For this purpose, during the execution of each task, the name, the type, the execution time of the task, the date of execution and the status (execution failed or not) are recorded in a log file and sent to the server.

4) System documentation: The system documentation activity is the vital and continuous activity of our approach. In fact, during this activity, the documents are produced to help the manager and the volunteers to use the system. It provides all information on the capabilities and characteristics of the system. It helps users understand the system and its essential reference materials. VC documentation is updated every time there are changes in the system. The documentation activity is at two levels: the manager level and the volunteer level.

At the manager level, the documents will allow the manager to install and configure the server. It will also give tips to the manager on how to motivate the volunteers to register, contribute and maximize system performance as well as how to customize the software so that it works best for each volunteers (e.g., profile suggestion given data collected from volunteers). The manager can also write Frequently Asked Questions (FAQ) to help volunteers.

At the volunteer level, documents are used to inform the volunteer about the system, and describe what it is intended to do and how it works. Overall, it explains to volunteers how to

install the VCSoftware in order to contribute/use the system and how to submit computations and collect the results. To facilitate access to non IT volunteers, a short video is a good way to show how to install the VCSoftware and another on how to submit tasks and collect results. The documentation about the contribution of other users is provided to newcomers in order to motivate the latter.

IV. Experimentation

For Volunteer Computing to be adopted, volunteer devices must provide enough computing power to solve associated computing problems. The first step in the development of our VC system consists of a pilot phase, which is the development and experimentation of a VC system. This section shows how this system has been built and used to solve the problem of matrix multiplication. In the following, the problem of matrix multiplication is first described in section IV-A. Section IV-B follows with the VC_UY Volunteer Computing system built at the University of Yaounde I in Cameroon. Finally, Section IV-C presents the experimentation of VC_UY on the matrix-matrix multiplication.

A. Matrix Multiplication

Matrix multiplication [46], [47], [48], [49] is an operation that produces a matrix product from two input matrices. Each matrix product entry is the dot product of a row in the first matrix and a column in the second (see equation 2). Matrix multiplication has many applications. It is the basic computational kernel for many algorithms of machine learning systems and recommendation systems. Matrix-vector multiplication is the core kernel of the PageRank algorithm [21], [48]. Matrix multiplication is often a computational bottleneck because generally, matrices are very large with dimensions which can easily reach hundreds, thousands and even millions [46], [47], [48], [49].

Three popular algorithms for matrix multiplication have been proposed in the literature: the iterative algorithm, the recursive algorithm and the Strassen algorithm. Let us consider two square matrices A and B given below.

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix},$$

$$\mathrm{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{bmatrix}$$

The matrix multiplication of A and B gives the matrix C obtained by using the equation 2.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{2}$$

From the previous equation, a simple iterative algorithm can be constructed using loops on the indices i, j from 1 to n. This algorithm takes time on the order of n^3. Using the Divide and Conquer approach, the matrices A, B are partitioned into blocks. Then, the multiplication gives:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

The Divide and Conquer approach works for all square matrices whose dimensions are powers of two. For matrices which do not respect this condition (e.g., matrix-vector multiplication), fill the missing rows and columns with zeros. The complexity of the Divide and Conquer approach is the same as the iterative algorithm, i.e. n^3. In fact, this approach requires 8 blocks multiplications to calculate the product matrix which still requires n^3 running time [46], [47], [49].

Another matrix multiplication approach is called Strassen algorithm. The Strassen algorithm [46], [50] is a recursive Divide and Conquer approach. For each recursive call, the input matrices are divided into 4 blocks but only 7 blocks multiplications are needed. Compared to the iterative and the Divide and Conquer approach, the Strassen approach needs only 7 block matrix multiplications, which involves a time complexity of $n^{2.807}$ [46], [47], [49]. When the matrices are large, the execution time of the matrix multiplication can be very long. Since matrix multiplication can be divided into sub-matrix multiplication, this task can be parallelized. The next sections will present how our Volunteer Computing System was built and used for matrix multiplication. All experiments will be performed using the Strassen algorithm.

B. VC_UY: The Volunteer Computing System of the University of Yaounde I

In order to overcome the HPC problems faced by researchers and students from the university of Yaounde I in Cameroon, it was decided to build a VC system named VC_UY. This section presents the summary of the pilot phase in two main points: the recruitment of volunteers and the designing of the VC system.

1) Recruitment of Volunteers: During the volunteer recruitment phase, the master students at the Department of Computer Science of the University of Yaounde I were met. Then, the HPC concepts and the problems encountered in the building of HPC platforms in resource poor settings were explained. Our approach based on volunteering was presented and their participation as volunteers was asked. Ten students were selected from those who generally connect to the Internet at least twice a day and whose computers can be available for computations for at least one hour per day. Table I presents the profile of each volunteer.

TABLE I. Characteristics of volunteer computers

RAM	CPU	SWAP	Operating system	Disk space
4GB	Core i3, 2.4GH	1GB	LINUX	50GB
4GB	Core i3, 2.4GH	2GB	WINDOWS	100GB
4GB	Core i4, 2.4GH	2GB	LINUX	100GB
4GB	Core i5, 2.7GH	1.5GB	LINUX	100GB
4GB	Core i5, 2.4GH	1.5GB	LINUX	50GB
4GB	Core i5, 2.4GH	1GB	LINUX	50GB
4GB	Core i3, 2.4GH	2GB	WINDOWS	50GB
4GB	Core i4, 2.4GH	750MB	LINUX	100GB
4GB	Core i4, 2.6GH	1GB	LINUX	100GB
4GB	Core i4, 2.4GH	2GB	LINUX	100GB

2) VC_UY system design: Once the volunteers were recruited, our system was designed as follows: the machine (CPU core i5 and 4 GB of RAM) was used as a server. The VCSoftware[10] was deployed on the server and on each volunteers.

As presented in Section III, the server and volunteers run different modules to exchange information and perform tasks. For the purpose of experimentation, all our source code was written using the Python programming language[11], IPython Application Programming Interface[12] and the Django framework[13]. IPython is an API for parallel and distributed computing. It enables to develop, execute, debug and monitor interactively all types of parallel applications. The architecture of serverManager and volunteersManager modules is completely based on the architecture of IPython API. The Django framework has enabled the implementation of a user-friendly web interface for volunteers (registration, consultation of information about other volunteers, etc.) and the manager (for monitoring using a dashboard).

The serverManager module consists of the front-end developed using the Django framework and the back-end developed using the python programming language and the IPython API. The main feature of the front-end is to support all managements activities (Section III-A) by presenting relevant information on a dashboard. On the back-end side, the IPython API allows the server to listen to the network and collect information on volunteers. For the purpose of experimentation, a script has been written[14] allowing to collect information on volunteers; and a software developed for the visualization of the contribution of each volunteers and the performance of the whole system.

The volunteersManager module consists of the front-end and the back-end. At the front-end, the Django framework presents a dashboard describing all the other volunteers to the volunteer. At the back-end side, the volunteer machine performs the following operations: sends computation to other volunteers; listens to the requests on the network, executes the code, and returns results back to related volunteers; accepts tasks, performs them; collects the results and sends them back to the volunteers; sends profile information to the server. For the purpose of experimentation, scripts have been written (available on github[15]) and permitting to send matrices to volunteers, perform the Strassen matrix multiplication algorithm and collect results.

C. Experimentation of Matrix-Matrix Multiplication on VC_UY

To test the VC_UY Volunteer Computing system, the Strassen matrix-matrix multiplication algorithm was implemented using Python and IPython API. Matrix multiplication was used with different input sizes on one computer (CPU Core i5 and 4GB of RAM) and the whole VC system. If A and B are block-partitioned matrices, the block dimension of the resulting matrix C is determined by considering the number of volunteers and the input block dimensions. A volunteer responsible for each resulting block retrieves all the necessary blocks from A and B to execute a multiplication operation locally. Fig. 2 shows the performance of the execution on one machine. Fig. 3 presents the matrix multiplication on VC_UY system, and Fig. 4 presents the contribution of each volunteers in the computing. Fig. 4 shows that although the volunteers used have close computing power, their contribution to the calculation varies according to their availability.

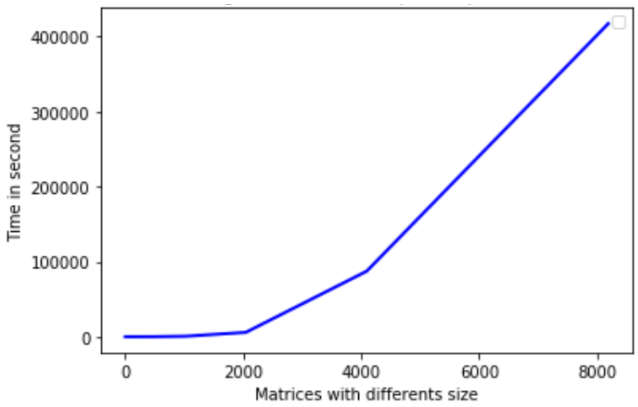

Fig. 2. Experimenting the Strassen algorithm for matrix-matrix multiplication on one machine Core i5 of CPU and 4GB of RAM

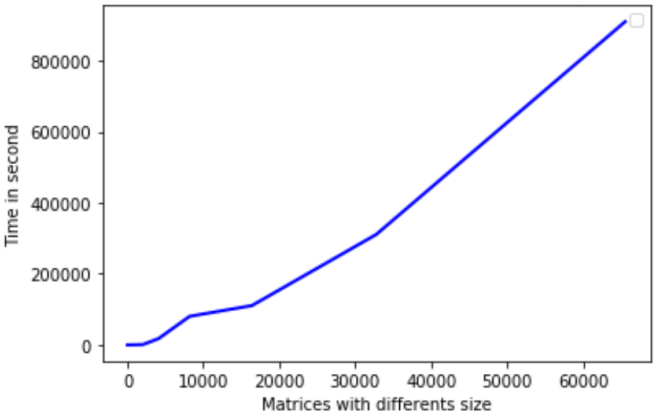

Fig. 3. Experimenting the Strassen algorithm for matrix-matrix multiplication on our Volunteer Computing system

As demonstrated by [25], [26], [51], the experiments conducted in this section shows that Volunteer Computing systems can provide a significant amount of computing power. Then, it is an important option for HPC problems in resource poor settings.

V. Conclusion

This article presented an approach allowing to build a Volunteer Computing system for HPC in resource poor settings. The experiments were made on the mathematical problem of solving matrix multiplication. Volunteers were recruited amongst students at the University of Yaounde I in Cameroon.

[10]https://github.com/admhamza/VC_UY
[11]https://www.python.org/
[12]https://ipyparallel.readthedocs.io/
[13]https://www.djangoproject.com/
[14]https://github.com/admhamza/VC_UY/blob/master/automatisation_collecte_informations.py
[15]https://github.com/admhamza/VC_UY/blob/master/calcul_distribue_dans_un_reseaux.py

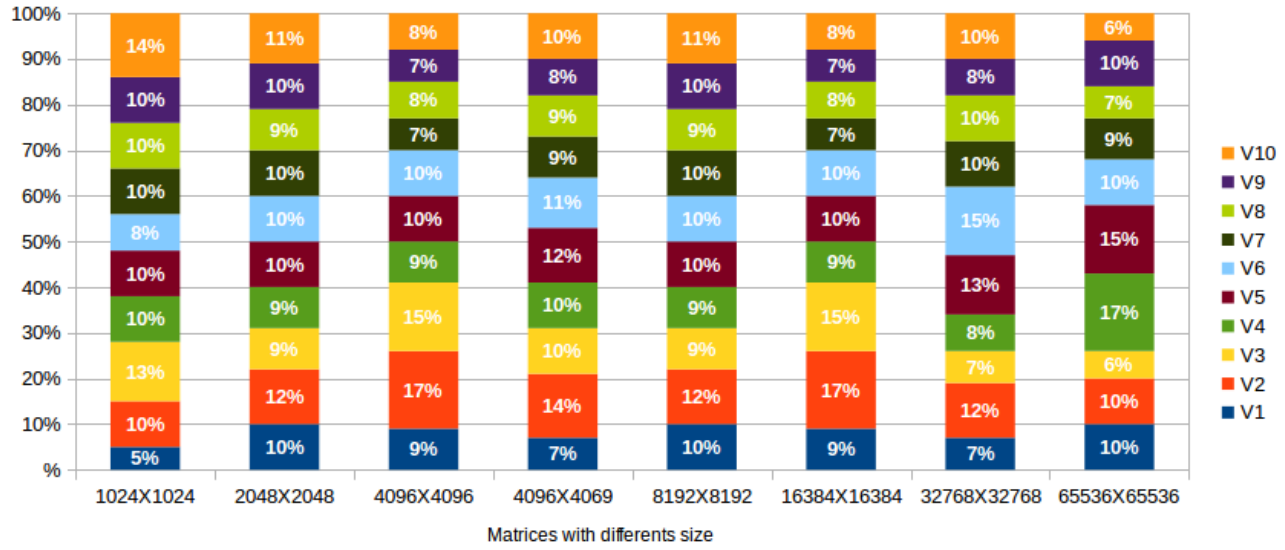

Fig. 4. Contribution of each machine. V_i represents the i^{th} volunteer

Experiments showed that a VC system can be used to enhance HPC in resource poor settings.

This work opens doors for significant possibilities in developing countries where the computing resources are generally limited. Given the success of the experiments, the recruitment of other volunteers from the University of Yaounde I is in progress. The goal being to create a powerful Volunteer Computing system with a maximum number of computers. During the experiment, potential volunteer machines were selected manually according to the data collected. This can be a difficult task if there are hundreds of volunteers registered in the system. Thus, also planned is the exploration and implementation of automatic methods for predicting volunteer machines for a given task.

ACKNOWLEDGMENTS

Our gratitude goes to all students who accepted to participate in this project, in particular Mr. Romeo Koati who recruited volunteers and participated to the development of the software.

REFERENCES

[1] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1989.

[2] Zahid Ansari, Asif Afzal, Moomin Muhiuddeen, and Sudarshan Nayak. Literature survey for the comparative study of various high performance computing techniques. *Int J Comput Trends Technol (IJCTT)*, 27(2):80–86, 2015.

[3] Fatima El Jamiy, Abderrahmane Daif, Mohamed Azouazi, and Abdelaziz Marzak. An effective storage mechanism for high performance computing (hpc). *International Journal of Advanced Computer Science and Applications*, 6(10), 2015.

[4] Zhiwei Xu, Xuebin Chi, and Nong Xiao. High-performance computing environment: a review of twenty years of experiments in China. *National Science Review*, 3(1):36–48, 01 2016.

[5] Ranjit Rajak. A comparative study: Taxonomy of high performance computing (hpc). *International Journal of Electrical & Computer Engineering (2088-8708)*, 8, 2018.

[6] Olaf Schenk, Helmar Burkhart, and Hema Reddy. Towards personalized medicine: High-performance computing in the life sciences. *ERCIM News*, 2008(74), 2008.

[7] Shaoliang Peng. High performance computational biology and drug design on tianhe supercomputers. In *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2016, Shenzhen, China, December 15-18, 2016*, page 7, 2016.

[8] Bertil Schmidt and Andreas Hildebrandt. Next-generation sequencing: big data meets high performance computing. *Drug discovery today*, 22(4):712–717, 2017.

[9] Georg Hager and Gerhard Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. Chapman and Hall / CRC computational science series. CRC Press, 2011.

[10] Daniel A Reed and Jack Dongarra. Exascale computing and big data. *Communications of the ACM*, 58(7):56–68, 2015.

[11] Bo Tang, Zhen Chen, Gerald Hefferman, Shuyi Pei, Tao Wei, Haibo He, and Qing Yang. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial informatics*, 13(5):2140–2150, 2017.

[12] Anwar Osseyran and Merle Giles. *Industrial applications of high-performance computing: best global practices*, volume 25. CRC Press, 2015.

[13] Ezell Stephen and Atkinson Robert. *The Vital Importance of High-Performance Computing to U.S. Competitiveness*. INFORMATION TECHNOLOGY INNOVATION FOUNDATION, 2016.

[14] N. R. Adiga, G. Almasi, G. S. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, and M. Blumrich et al. An overview of the bluegene/l supercomputer. In *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60, Nov 2002.

[15] Erich Strohmaier, Hans Werner Meuer, Jack J. Dongarra, and Horst D. Simon. The TOP500 list and progress in high-performance computing. *IEEE Computer*, 48(11):42–49, 2015.

[16] Douglas Eadline. *High Performance Computing For Dummies*. Wiley Publishing, Inc., USA, 5th edition, 2009.

[17] Chee Shin Yeo, Rajkumar Buyya, Hossein Pourreza, Rasit Eskicioglu, Peter Graham, and Frank Sommers. *Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers*, pages 521–551. Springer US, Boston, MA, 2006.

[18] Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. Gpu cluster for high performance computing. In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, SC '04, pages 47–, Washington, DC, USA, 2004. IEEE Computer Society.

[19] Violeta Holmes and Ibad Kureshi. Developing high performance computing resources for teaching cluster and grid computing courses. *Procedia Computer Science*, 51:1714 – 1723, 2015. International Conference On Computational Science, ICCS 2015.

[20] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[21] Abhishek Gupta, Laxmikant V Kale, Filippo Gioachin, Verdi March, Chun Hui Suen, Bu Sung Lee, Paolo Faraboschi, Richard Kaufmann, and Dejan Milojicic. The who, what, why and how of high performance computing applications in the cloud. *HP Laboratories Technical Report*, (49), 8 2013.

[22] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[23] Peter M. Kasson, Daniel L. Ensign, and Vijay S. Pande. Combining molecular dynamics with bayesian analysis to predict and evaluate ligand-binding mutations in influenza hemagglutinin. *Journal of the American Chemical Society*, 131(32):11338–11340, 2009. PMID: 19637916.

[24] Zied TRIFA, Mohamed LABIDI, and Maher KHEMAKHEM. Arabic cursive characters distributed recognition using the dtw algorithm on boinc: Performance analysis. *International Journal of Advanced Computer Science and Applications*, 2(3), 2011.

[25] Erick Lavoie and Laurie Hendren. Personal volunteer computing. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, pages 240–246, New York, NY, USA, 2019. ACM.

[26] Erick Lavoie, Laurie Hendren, Frederic Desprez, and Miguel Correia. Pando: Personal volunteer computing in browsers, 2018.

[27] Oded Nov, David Anderson, and Ofer Arazy. Volunteer computing: A model of the factors determining contribution to community-based scientific research. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 741–750, New York, NY, USA, 2010. ACM.

[28] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, 2016.

[29] Shurong Tian, Todd Takken, Vic Mahaney, Christopher Marroquin, Mark Schultz, Mark Hoffmeyer, Yuan Yao, Kevin O'Connell, Anil Yuksel, and Paul Coteus. Summit and sierra supercomputer cooling solutions. *IBM Journal of Research and Development*, 2019.

[30] James R Kozloski, Timothy M Lynar, Mark D Nelson, and John M Wagner. Energy efficient supercomputer job allocation, 2018. US Patent 10,025,639.

[31] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[32] Zvi Tannenbaum and Dean E Dauger. Cluster computing, 2019. US Patent 10,333,768.

[33] Minakshi Tripathy and C.R. Tripathy. On a virtual shared memory cluster system with virtual machines. *International Journal of Computer and Electrical Engineering*, 3:754–761, 01 2011.

[34] Mohammad Moghadasi, Seyed Majid Mousavi, and Gábor Fazekas. Cloud computing auditing. *International Journal of Advanced Computer Science and Applications*, 9(12), 2018.

[35] Aferdita Ibrahimi. Cloud computing: Pricing model. *International Journal of Advanced Computer Science and Applications*, 8(6), 2017.

[36] Babur Hayat Malik, Jazba Asad, Sabila Kousar, Faiza Nawaz, Zainab, Farania Hayder, Sehresh Bibi, Amina Yousaf, and Ali Raza. Cloud computing adoption in small and medium- sized enterprises (smes) of asia and africa. *International Journal of Advanced Computer Science and Applications*, 10(5), 2019.

[37] Hans-Joachim Bungartz. Sparse grids and their impact on hpc and big data. In *Kolloquiumsvortrag*, 2019.

[38] Tessema M Mengistu and Dunren Che. Survey and taxonomy of volunteer computing. *ACM Computing Surveys (CSUR)*, 52(3):1–35, 2019.

[39] Bruno Donassolo, Henri Casanova, Arnaud Legrand, and Pedro Velho. Fast and scalable simulation of volunteer computing systems using simgrid. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 605–612, New York, NY, USA, 2010. ACM.

[40] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: An experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[41] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–8, May 2009.

[42] Wei Li and William W Guo. The optimization potential of volunteer computing for compute or data intensive applications. *Journal of Communications*, 14(10), 2019.

[43] Harold E Castro. Capacity of desktop clouds for running hpc applications: A revisited analysis. In *Applied Informatics: Second International Conference, ICAI 2019, Madrid, Spain, November 7–9, 2019, Proceedings*, volume 1051, page 257. Springer Nature, 2019.

[44] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[45] Maarten Van Steen and Andrew S Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[46] Qingshan Luo and John B. Drake. A scalable parallel strassen's matrix multiplication algorithm for distributed-memory computers. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 221–226, New York, NY, USA, 1995. ACM.

[47] Chandan Misra, Sourangshu Bhattacharya, and Soumya K. Ghosh. Stark: Fast and scalable strassen's matrix multiplication using apache spark, 2018.

[48] M. Son and K. Lee. Distributed matrix multiplication performance estimator for machine learning jobs in cloud computing. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 638–645, July 2018.

[49] Zhen Xie, Guangming Tan, Weifeng Liu, and Ninghui Sun. Ia-spgemm: An input-aware auto-tuning framework for parallel sparse matrix-matrix multiplication. In *Proceedings of the ACM International Conference on Supercomputing*, ICS '19, pages 94–105, New York, NY, USA, 2019. ACM.

[50] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[51] Travis Desell. Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters. pages 19–28, 10 2017.

UNIVERSITÉ DE YAOUNDÉ I **Faculté des Sciences** Division de la Programmation et du Suivi des Activités Académiques		THE UNIVERSITY OF YAOUNDE I **Faculty of Science** Division of Programming and Follow-up of Academic Affaires
LISTE DES ENSEIGNANTS PERMANENTS		**LIST OF PERMANENT TEACHING STAFF**

ANNÉE ACADÉMIQUE 2019/2020
(Par Département et par Grade)
DATE D'ACTUALISATION 12 Juin 2020

ADMINISTRATION

DOYEN : TCHOUANKEU Jean- Claude, *Maître de Conférences*
VICE-DOYEN / DPSAA : ATCHADE Alex de Théodore, *Maître de Conférences*
VICE-DOYEN / DSSE : AJEAGAH Gideon AGHAINDUM, *Professeur*
VICE-DOYEN / DRC : ABOSSOLO Monique, *Maître de Conférences*
Chef Division Administrative et Financière : NDOYE FOE Marie C. F., *Maître de Conférences*
Chef Division des Affaires Académiques, de la Scolarité et de la Recherche DAASR : MBAZE MEVA'A Luc Léonard, *Professeur*

1- DÉPARTEMENT DE BIOCHIMIE (BC) (38)

N°	NOMS ET PRÉNOMS	GRADE	OBSERVATIONS
1	BIGOGA DIAGA Jude	Professeur	En poste
2	FEKAM BOYOM Fabrice	Professeur	En poste
3	FOKOU Elie	Professeur	En poste
4	KANSCI Germain	Professeur	En poste
5	MBACHAM FON Wilfried	Professeur	En poste
6	MOUNDIPA FEWOU Paul	Professeur	Chef de Département
7	NINTCHOM PENLAP V. épse BENG	Professeur	En poste
8	OBEN Julius ENYONG	Professeur	En poste
9	ACHU Merci BIH	Maître de Conférences	En poste
10	ATOGHO Barbara Mma	Maître de Conférences	En poste
11	AZANTSA KINGUE GABIN BORIS	Maître de Conférences	En poste
12	BELINGA née NDOYE FOE M. C. F.	Maître de Conférences	Chef DAF / FS
13	BOUDJEKO Thaddée	Maître de Conférences	En poste
14	DJUIDJE NGOUNOUE Marcelline	Maître de Conférences	En poste

15	EFFA NNOMO Pierre	Maître de Conférences	En poste
16	NANA Louise épouse WAKAM	Maître de Conférences	En poste
17	NGONDI Judith Laure	Maître de Conférences	En poste
18	NGUEFACK Julienne	Maître de Conférences	En poste
19	NJAYOU Frédéric Nico	Maître de Conférences	En poste
20	MOFOR née TEUGWA Clotilde	Maître de Conférences	Inspecteur de Service MINESUP
21	TCHANA KOUATCHOUA Angèle	Maître de Conférences	En poste

22	AKINDEH MBUH NJI	Chargé de Cours	En poste
23	BEBOY EDZENGUELE Sara Nathalie	Chargée de Cours	En poste
24	DAKOLE DABOY Charles	Chargé de Cours	En poste
25	DJUIKWO NKONGA Ruth Viviane	Chargée de Cours	En poste
26	DONGMO LEKAGNE Joseph Blaise	Chargé de Cours	En poste
27	EWANE Cécile Anne	Chargée de Cours	En poste
28	FONKOUA Martin	Chargé de Cours	En poste
29	BEBEE Fadimatou	Chargée de Cours	En poste
30	KOTUE KAPTUE Charles	Chargé de Cours	En poste
31	LUNGA Paul KEILAH	Chargé de Cours	En poste
32	MANANGA Marlyse Joséphine	Chargée de Cours	En poste
33	MBONG ANGIE M. Mary Anne	Chargée de Cours	En poste
34	PECHANGOU NSANGOU Sylvain	Chargé de Cours	En poste
35	Palmer MASUMBE NETONGO	Chargé de Cours	En poste

36	MBOUCHE FANMOE Marceline Joëlle	Assistante	En poste
37	OWONA AYISSI Vincent Brice	Assistant	En poste
38	WILFRIED ANGIE Abia	Assistante	En poste

2- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE ANIMALES (BPA) (48)

1	AJEAGAH Gideon AGHAINDUM	Professeur	*VICE-DOYEN / DSSE*
2	BILONG BILONG Charles-Félix	Professeur	Chef de Département
3	DIMO Théophile	Professeur	En Poste
4	DJIETO LORDON Champlain	Professeur	En Poste

5	ESSOMBA née NTSAMA MBALA	Professeur	*Vice Doyen/FMSB/UYI*
6	FOMENA Abraham	Professeur	En Poste
7	KAMTCHOUING Pierre	Professeur	En poste
8	NJAMEN Dieudonné	Professeur	En poste
9	NJIOKOU Flobert	Professeur	En Poste
10	NOLA Moïse	Professeur	En poste
11	TAN Paul VERNYUY	Professeur	En poste
12	TCHUEM TCHUENTE Louis Albert	Professeur	*Inspecteur de service Coord.Progr./MINSANTE*
13	ZEBAZE TOGOUET Serge Hubert	Professeur	En poste

14	BILANDA Danielle Claude	Maître de Conférences	En poste
15	DJIOGUE Séfirin	Maître de Conférences	En poste
16	DZEUFIET DJOMENI Paul Désiré	Maître de Conférences	En poste
17	JATSA BOUKENG Hermine épse MEGAPTCHE	Maître de Conférences	En Poste
18	KEKEUNOU Sévilor	Maître de Conférences	En poste
19	MEGNEKOU Rosette	Maître de Conférences	En poste
20	MONY Ruth épse NTONE	Maître de Conférences	En Poste
21	NGUEGUIM TSOFACK Florence	Maître de Conférences	En poste
22	TOMBI Jeannette	Maître de Conférences	En poste

23	ALENE Désirée Chantal	Chargée de Cours	En poste
26	ATSAMO Albert Donatien	Chargé de Cours	En poste
27	BELLET EDIMO Oscar Roger	Chargé de Cours	En poste
28	DONFACK Mireille	Chargée de Cours	En poste
29	ETEME ENAMA Serge	Chargé de Cours	En poste
30	GOUNOUE KAMKUMO Raceline	Chargée de Cours	En poste
31	KANDEDA KAVAYE Antoine	Chargé de Cours	En poste
32	LEKEUFACK FOLEFACK Guy B.	Chargé de Cours	En poste
33	MAHOB Raymond Joseph	Chargé de Cours	En poste
34	MBENOUN MASSE Paul Serge	Chargé de Cours	En poste
35	MOUNGANG Luciane Marlyse	Chargée de Cours	En poste
36	MVEYO NDANKEU Yves Patrick	Chargé de Cours	En poste
37	NGOUATEU KENFACK Omer Bébé	Chargé de Cours	En poste
38	NGUEMBOK	Chargé de Cours	En poste
39	NJUA Clarisse Yafi	Chargée de Cours	Chef Div. UBA

40	NOAH EWOTI Olive Vivien	Chargée de Cours	En poste
41	TADU Zephyrin	Chargé de Cours	En poste
42	TAMSA ARFAO Antoine	Chargé de Cours	En poste
43	YEDE	Chargé de Cours	En poste

44	BASSOCK BAYIHA Etienne Didier	Assistant	En poste
45	ESSAMA MBIDA Désirée Sandrine	Assistante	En poste
46	KOGA MANG DOBARA	Assistant	En poste
47	LEME BANOCK Lucie	Assistante	En poste
48	YOUNOUSSA LAME	Assistant	En poste

3- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE VÉGÉTALES (BPV) (33)

1	AMBANG Zachée	Professeur	Chef Division/UYII
2	BELL Joseph Martin	Professeur	En poste
3	DJOCGOUE Pierre François	Professeur	En poste
4	MOSSEBO Dominique Claude	Professeur	En poste
5	YOUMBI Emmanuel	Professeur	Chef de Département
6	ZAPFACK Louis	Professeur	En poste

7	ANGONI Hyacinthe	Maître de Conférences	En poste
8	BIYE Elvire Hortense	Maître de Conférences	En poste
9	KENGNE NOUMSI Ives Magloire	Maître de Conférences	En poste
10	MALA Armand William	Maître de Conférences	En poste
11	MBARGA BINDZI Marie Alain	Maître de Conférences	CT/ MINESUP
12	MBOLO Marie	Maître de Conférences	En poste
13	NDONGO BEKOLO	Maître de Conférences	*CE / MINRESI*
14	NGODO MELINGUI Jean Baptiste	Maître de Conférences	En poste
15	NGONKEU MAGAPTCHE Eddy L.	Maître de Conférences	En poste
16	TSOATA Esaïe	Maître de Conférences	En poste
17	TONFACK Libert Brice	Maître de Conférences	En poste

18	DJEUANI Astride Carole	Chargé de Cours	En poste
19	GOMANDJE Christelle	Chargée de Cours	En poste

20	MAFFO MAFFO Nicole Liliane	Chargé de Cours	En poste
21	MAHBOU SOMO TOUKAM. Gabriel	Chargé de Cours	En poste
22	NGALLE Hermine BILLE	Chargée de Cours	En poste
23	NGOUO Lucas Vincent	Chargé de Cours	En poste
24	NNANGA MEBENGA Ruth Laure	Chargé de Cours	En poste
25	NOUKEU KOUAKAM Armelle	Chargé de Cours	En poste
26	ONANA JEAN MICHEL	Chargé de Cours	En poste
27	GODSWILL NTSOMBAH NTSEFONG	Assistant	En poste
28	KABELONG BANAHO Louis-Paul-Roger	Assistant	En poste
29	KONO Léon Dieudonné	Assistant	En poste
30	LIBALAH Moses BAKONCK	Assistant	En poste
31	LIKENG-LI-NGUE Benoit C	Assistant	En poste
32	TAEDOUNG Evariste Hermann	Assistant	En poste
33	TEMEGNE NONO Carine	Assistant	En poste

4- DÉPARTEMENT DE CHIMIE INORGANIQUE (CI) (34)

1	AGWARA ONDOH Moïse	Professeur	*Chef de Département*
2	ELIMBI Antoine	Professeur	En poste
3	Florence UFI CHINJE épouse MELO	Professeur	*Recteur Univ.Ngaoundere*
4	GHOGOMU Paul MINGO	Professeur	*Ministre Chargé deMiss.PR*
5	NANSEU Njiki Charles Péguy	Professeur	En poste
6	NDIFON Peter TEKE	Professeur	*CT MINRESI*
7	NGOMO Horace MANGA	Professeur	*Vice Chancelor/UB*
8	NDIKONTAR Maurice KOR	Professeur	*Vice-Doyen Univ. Bamenda*
9	NENWA Justin	Professeur	En poste
10	NGAMENI Emmanuel	Professeur	*DOYEN FS UDs*

11	BABALE née DJAM DOUDOU	Maître de Conférences	*Chargée Mission P.R.*
12	DJOUFAC WOUMFO Emmanuel	Maître de Conférences	En poste
13	EMADACK Alphonse	Maître de Conférences	En poste
14	KAMGANG YOUBI Georges	Maître de Conférences	En poste
15	KEMMEGNE MBOUGUEM Jean C.	Maître de Conférences	En poste

16	KONG SAKEO	Maître de Conférences	En poste
17	NDI NSAMI Julius	Maître de Conférences	En poste
18	NJIOMOU C. épse DJANGANG	Maître de Conférences	En poste
19	NJOYA Dayirou	Maître de Conférences	En poste

20	ACAYANKA Elie	Chargé de Cours	En poste
21	BELIBI BELIBI Placide Désiré	Chargé de Cours	CS/ ENS Bertoua
22	CHEUMANI YONA Arnaud M.	Chargé de Cours	En poste
23	KENNE DEDZO GUSTAVE	Chargé de Cours	En poste
24	KOUOTOU DAOUDA	Chargé de Cours	En poste
25	MAKON Thomas Beauregard	Chargé de Cours	En poste
26	MBEY Jean Aime	Chargé de Cours	En poste
27	NCHIMI NONO KATIA	Chargé de Cours	En poste
28	NEBA nee NDOSIRI Bridget NDOYE	Chargée de Cours	CT/ MINFEM
29	NYAMEN Linda Dyorisse	Chargée de Cours	En poste
30	PABOUDAM GBAMBIE A.	Chargée de Cours	En poste
31	TCHAKOUTE KOUAMO Hervé	Chargé de Cours	En poste
32	NJANKWA NJABONG N. Eric	Assistant	En poste
33	PATOUOSSA ISSOFA	Assistant	En poste
34	SIEWE Jean Mermoz	Assistant	En Poste

5- DÉPARTEMENT DE CHIMIE ORGANIQUE (CO) (35)			
1	DONGO Etienne	Professeur	Vice-Doyen
2	GHOGOMU TIH Robert Ralph	Professeur	Dir. IBAF/UDA
3	NGOUELA Silvère Augustin	Professeur	Chef de Departement UDS
4	NKENGFACK Augustin Ephrem	Professeur	Chef de Département
5	NYASSE Barthélemy	Professeur	En poste
6	PEGNYEMB Dieudonné Emmanuel	Professeur	*Directeur/ MINESUP*
7	WANDJI Jean	Professeur	En poste

8	Alex de Théodore ATCHADE	Maître de Conférences	Vice-Doyen / DPSAA
9	EYONG Kenneth OBEN	Maître de Conférences	En poste
10	FOLEFOC Gabriel NGOSONG	Maître de Conférences	En poste
11	FOTSO WABO Ghislain	Maître de Conférences	En poste

12	KEUMEDJIO Félix	Maître de Conférences	En poste
13	KEUMOGNE Marguerite	Maître de Conférences	En poste
14	KOUAM Jacques	Maître de Conférences	En poste
15	MBAZOA née DJAMA Céline	Maître de Conférences	En poste
16	MKOUNGA Pierre	Maître de Conférences	En poste
17	NOTE LOUGBOT Olivier Placide	Maître de Conférences	Chef Service/MINESUP
18	NGO MBING Joséphine	Maître de Conférences	Sous/Direct. MINERESI
19	NGONO BIKOBO Dominique Serge	Maître de Conférences	En poste
20	NOUNGOUE TCHAMO Diderot	Maître de Conférences	En poste
21	TABOPDA KUATE Turibio	Maître de Conférences	En poste
22	TCHOUANKEU Jean-Claude	Maître de Conférences	*Doyen /FS/ UYI*
23	TIH née NGO BILONG E. Anastasie	Maître de Conférences	En poste
24	YANKEP Emmanuel	Maître de Conférences	En poste

25	AMBASSA Pantaléon	Chargé de Cours	En poste
26	KAMTO Eutrophe Le Doux	Chargé de Cours	En poste
27	MVOT AKAK CARINE	Chargé de Cours	En poste
28	NGNINTEDO Dominique	Chargé de Cours	En poste
29	NGOMO Orléans	Chargée de Cours	En poste
30	OUAHOUO WACHE Blandine M.	Chargée de Cours	En poste
31	SIELINOU TEDJON Valérie	Chargé de Cours	En poste
32	TAGATSING FOTSING Maurice	Chargé de Cours	En poste
33	ZONDENDEGOUMBA Ernestine	Chargée de Cours	En poste

34	MESSI Angélique Nicolas	Assistant	En poste
35	TSEMEUGNE Joseph	Assistant	En poste

6- DÉPARTEMENT D'INFORMATIQUE (IN) (27)

1	ATSA ETOUNDI Roger	Professeur	*Chef Div.MINESUP*
2	FOUDA NDJODO Marcel Laurent	Professeur	*Chef Dpt ENS/Chef IGA.MINESUP*

3	NDOUNDAM Réné	Maître de Conférences	En poste

4	AMINOU Halidou	Chargé de Cours	*Chef de Département*
5	DJAM Xaviera YOUH - KIMBI	Chargé de Cours	En Poste
6	EBELE Serge Alain	Chargé de Cours	En poste
7	KOUOKAM KOUOKAM E. A.	Chargé de Cours	En poste
8	MELATAGIA YONTA Paulin	Chargé de Cours	En poste
9	MOTO MPONG Serge Alain	Chargé de Cours	En poste
10	TAPAMO Hyppolite	Chargé de Cours	En poste
11	ABESSOLO ALO'O Gislain	Chargé de Cours	En poste
12	MONTHE DJIADEU Valery M.	Chargé de Cours	En poste
13	OLLE OLLE Daniel Claude Delort	Chargé de Cours	C/D Enset. Ebolowa
14	TINDO Gilbert	Chargé de Cours	En poste
15	TSOPZE Norbert	Chargé de Cours	En poste
16	WAKU KOUAMOU Jules	Chargé de Cours	En poste

17	BAYEM Jacques Narcisse	Assistant	En poste
18	DOMGA KOMGUEM Rodrigue	Assistant	En poste
19	EKODECK Stéphane Gaël Raymond	Assistant	En poste
20	HAMZA Adamou	Assistant	En poste
21	JIOMEKONG AZANZI Fidel	Assistant	En poste
22	MAKEMBE. S . Oswald	Assistant	En poste
23	MESSI NGUELE Thomas	Assistant	En poste
24	MEYEMDOU Nadège Sylvianne	Assistante	En poste
25	NKONDOCK. MI. BAHANACK.N.	Assistant	En poste

7- DÉPARTEMENT DE MATHÉMATIQUES (MA) (30)

1	EMVUDU WONO Yves S.	Professeur	*Inspecteur MINESUP*

2	AYISSI Raoult Domingo	Maître de Conférences	Chef de Département
3	NKUIMI JUGNIA Célestin	Maître de Conférences	En poste
4	NOUNDJEU Pierre	Maître de Conférences	*Chef service des programmes & Diplômes*
5	MBEHOU Mohamed	Maître de Conférences	En poste
6	TCHAPNDA NJABO Sophonie B.	Maître de Conférences	Directeur/AIMS Rwanda

7	AGHOUKENG JIOFACK Jean Gérard	Chargé de Cours	Chef Cellule MINPLAMAT
8	CHENDJOU Gilbert	Chargé de Cours	En poste
9	DJIADEU NGAHA Michel	Chargé de Cours	En poste
10	DOUANLA YONTA Herman	Chargé de Cours	En poste
11	FOMEKONG Christophe	Chargé de Cours	En poste
12	KIANPI Maurice	Chargé de Cours	En poste
13	KIKI Maxime Armand	Chargé de Cours	En poste
14	MBAKOP Guy Merlin	Chargé de Cours	En poste
15	MBANG Joseph	Chargé de Cours	En poste
16	MBELE BIDIMA Martin Ledoux	Chargé de Cours	En poste
17	MENGUE MENGUE David Joe	Chargé de Cours	En poste
18	NGUEFACK Bernard	Chargé de Cours	En poste
19	NIMPA PEFOUKEU Romain	Chargée de Cours	En poste
20	POLA DOUNDOU Emmanuel	Chargé de Cours	En poste
21	TAKAM SOH Patrice	Chargé de Cours	En poste
22	TCHANGANG Roger Duclos	Chargé de Cours	En poste
23	TCHOUNDJA Edgar Landry	Chargé de Cours	En poste
24	TETSADJIO TCHILEPECK M. E.	Chargée de Cours	En poste
25	TIAYA TSAGUE N. Anne-Marie	Chargée de Cours	En poste
26	MBIAKOP Hilaire George	Assistant	En poste
27	BITYE MVONDO Esther Claudine	Assistante	En poste
28	MBATAKOU Salomon Joseph	Assistant	En poste
29	MEFENZA NOUNTU Thiery	Assistant	En poste
30	TCHEUTIA Daniel Duviol	Assistant	En poste

8- DÉPARTEMENT DE MICROBIOLOGIE (MIB) (18)			
1	ESSIA NGANG Jean Justin	Professeur	*Chef de Département*

2	BOYOMO ONANA	Maître de Conférences	En poste
3	NWAGA Dieudonné M.	Maître de Conférences	En poste
4	NYEGUE Maximilienne Ascension	Maître de Conférences	En poste
5	RIWOM Sara Honorine	Maître de Conférences	En poste
6	SADO KAMDEM Sylvain Leroy	Maître de Conférences	En poste

7	ASSAM ASSAM Jean Paul	Chargé de Cours	En poste
8	BODA Maurice	Chargé de Cours	En poste
9	BOUGNOM Blaise Pascal	Chargé de Cours	En poste
10	ESSONO OBOUGOU Germain G.	Chargé de Cours	En poste
11	NJIKI BIKOÏ Jacky	Chargée de Cours	En poste
12	TCHIKOUA Roger	Chargé de Cours	En poste
13	ESSONO Damien Marie	Assistant	En poste
14	LAMYE Glory MOH	Assistant	En poste
15	MEYIN A EBONG Solange	Assistante	En poste
16	NKOUDOU ZE Nardis	Assistant	En poste
17	SAKE NGANE Carole Stéphanie	Assistante	En poste
18	TOBOLBAÏ Richard	Assistant	En poste

9. DEPARTEMENT DE PYSIQUE(PHY) (40)

1	BEN- BOLIE Germain Hubert	Professeur	En poste
2	EKOBENA FOUDA Henri Paul	Professeur	*Chef Division. UN*
3	ESSIMBI ZOBO Bernard	Professeur	En poste
4	KOFANE Timoléon Crépin	Professeur	En poste
5	NANA ENGO Serge Guy	Professeur	En poste
6	NDJAKA Jean Marie Bienvenu	Professeur	Chef de Département
7	NOUAYOU Robert	Professeur	En poste
8	NJANDJOCK NOUCK Philippe	Professeur	*Sous Directeur/ MINRESI*
9	PEMHA Elkana	Professeur	En poste
10	TABOD Charles TABOD	Professeur	Doyen Univ/Bda
11	TCHAWOUA Clément	Professeur	En poste
12	WOAFO Paul	Professeur	En poste

13	BIYA MOTTO Frédéric	Maître de Conférences	DG/HYDRO Mekin
14	BODO Bertrand	Maître de Conférences	En poste
15	DJUIDJE KENMOE épouse ALOYEM	Maître de Conférences	En poste
16	EYEBE FOUDA Jean sire	Maître de Conférences	En poste
17	FEWO Serge Ibraïd	Maître de Conférences	En poste
18	HONA Jacques	Maître de Conférences	En poste
19	MBANE BIOUELE César	Maître de Conférences	En poste
20	NANA NBENDJO Blaise	Maître de Conférences	En poste
21	NDOP Joseph	Maître de Conférences	En poste
22	SAIDOU	Maître de Conférences	MINERESI
23	SIEWE SIEWE Martin	Maître de Conférences	En poste
24	SIMO Elie	Maître de Conférences	En poste
25	VONDOU Derbetini Appolinaire	Maître de Conférences	En poste
26	WAKATA née BEYA Annie	Maître de Conférences	*Sous Directeur/ MINESUP*
27	ZEKENG Serge Sylvain	Maître de Conférences	En poste

28	ABDOURAHIMI	Chargé de Cours	En poste
29	EDONGUE HERVAIS	Chargé de Cours	En poste
30	ENYEGUE A NYAM épse BELINGA	Chargée de Cours	En poste
31	FOUEDJIO David	Chargé de Cours	Chef Cell. MINADER
32	MBINACK Clément	Chargé de Cours	En poste
33	MBONO SAMBA Yves Christian U.	Chargé de Cours	En poste
34	MELI'I Joelle Larissa	Chargée de Cours	En poste
35	MVOGO ALAIN	Chargé de Cours	En poste
36	OBOUNOU Marcel	Chargé de Cours	DA/Univ Inter Etat/Sangmalima
37	WOULACHE Rosalie Laure	Chargée de Cours	En poste

38	AYISSI EYEBE Guy François Valérie	Assistant	En poste
39	CHAMANI Roméo	Assistant	En poste
40	TEYOU NGOUPOU Ariel	Assistant	En poste

10- DÉPARTEMENT DE SCIENCES DE LA TERRE (ST) (43)

1	BITOM Dieudonné	Professeur	*Doyen / FASA / UDs*
2	FOUATEU Rose épse YONGUE	Professeur	En poste
3	KAMGANG Pierre	Professeur	En poste
4	NDJIGUI Paul Désiré	Professeur	Chef de Département
5	NDAM NGOUPAYOU Jules-Remy	Professeur	En poste
6	NGOS III Simon	Professeur	DAAC/Uma
7	NKOUMBOU Charles	Professeur	En poste
8	NZENTI Jean-Paul	Professeur	En poste

9	ABOSSOLO née ANGUE Monique	Maître de Conférences	*Vice-Doyen / DRC*
10	GHOGOMU Richard TANWI	Maître de Conférences	CD/Uma
11	MOUNDI Amidou	Maître de Conférences	*CT/ MINIMDT*
12	NGUEUTCHOUA Gabriel	Maître de Conférences	CEA/MINRESI
13	NJILAH Isaac KONFOR	Maître de Conférences	En poste
14	ONANA Vincent Laurent	Maître de Conférences	*Chef service Maintenance & du Matériel*
15	BISSO Dieudonné	Maître de Conférences	*Directeur/Projet Barrage Memve'ele*
16	EKOMANE Emile	Maître de Conférences	En poste
17	GANNO Sylvestre	Maître de Conférences	En poste
18	NYECK Bruno	Maître de Conférences	En poste
19	TCHOUANKOUE Jean-Pierre	Maître de Conférences	En poste
20	TEMDJIM Robert	Maître de Conférences	En poste
21	YENE ATANGANA Joseph Q.	Maître de Conférences	*Chef Div. /MINTP*
22	ZO'O ZAME Philémon	Maître de Conférences	*DG/ART*

23	ANABA ONANA Achille Basile	Chargé de Cours	En poste
24	BEKOA Etienne	Chargé de Cours	En poste
25	ELISE SABABA	Chargé de Cours	En poste
26	ESSONO Jean	Chargé de Cours	En poste
27	EYONG JOHN TAKEM	Chargé de Cours	En poste
28	FUH Calistus Gentry	Chargé de Cours	*Sec. D'Etat/MINMIDT*
29	LAMILEN BILLA Daniel	Chargé de Cours	En poste

30	MBESSE CECILE OLIVE	Chargée de Cours	En poste
31	MBIDA YEM	Chargé de Cours	En poste
32	METANG Victor	Chargé de Cours	En poste
33	MINYEM Dieudonné-Lucien	Chargé de Cours	*CD/Uma*
34	NGO BELNOUN Rose Noël	Chargée de Cours	En poste
35	NGO BIDJECK Louise Marie	Chargée de Cours	En poste
36	NOMO NEGUE Emmanuel	Chargé de Cours	En poste
37	NTSAMA ATANGANA Jacqueline	Chargé de Cours	En poste
38	TCHAKOUNTE J. épse NOUMBEM	Chargée de Cours	*Chef.cell / MINRESI*
39	TCHAPTCHET TCHATO De P.	Chargé de Cours	En poste
40	TEHNA Nathanaël	Chargé de Cours	En poste
41	TEMGA Jean Pierre	Chargé de Cours	En poste
42	FEUMBA Roger	Assistant	En poste
43	MBANGA NYOBE Jules	Assistant	En poste

Répartition chiffrée des Enseignants de la Faculté des Sciences de l'Université de Yaoundé I

	NOMBRE D'ENSEIGNANTS				
DÉPARTEMENT	**Professeurs**	**Maîtres de Conférences**	**Chargés de Cours**	**Assistants**	**Total**
BCH	9 (1)	13 (09)	14 (06)	3 (2)	**39 (18)**
BPA	13 (1)	09 (06)	19 (05)	05 (2)	**46 (14)**
BPV	06 (0)	11 (02)	9 (06)	07 (01)	**33 (9)**
CI	10 (1)	9 (02)	12 (02)	03 (0)	**34 (5)**
CO	7 (0)	17 (04)	09 (03)	02 (0)	**35(7)**
IN	2 (0)	1 (0)	13 (01)	09 (01)	**25 (2)**
MAT	1 (0)	5 (0)	19 (01)	05 (02)	**30 (3)**
MIB	1 (0)	5 (02)	06 (01)	06 (02)	**18 (5)**
PHY	12 (0)	15 (02)	10 (03)	03 (0)	**40 (5)**
ST	8 (1)	14 (01)	19 (05)	02 (0)	**43(7)**
Total	**69 (4)**	**99 (28)**	**130 (33)**	**45 (10)**	**343 (75)**

Soit un total de **344 (**75**)** dont :

- Professeurs **68 (4)**
- Maîtres de Conférences **99 (**28**)**
- Chargés de Cours **130 (33)**
- Assistants **46** (10**)**

( ) = Nombre de Femmes **75**