RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DE BATNA 2

FACULTÉ DES MATHÉMATIQUES ET D'INFORMATIQUE

THÈSE

pour obtenir le diplôme de

Docteur EN SCIENCES

Spécialité : **Informatique**

présentée et soutenue publiquement par

KADACHE NABIL

le 08/12/2020

Titre:

Un environnement d'exécution de simulation basé sur le calcul volontaire

Jury

M. GUEZOULI Larbi,	MCA,	Université de Batna 2,	Président
M. SEGHIR Rachid,	Prof.,	Université de Batna 2,	Rapporteur
M. KAHLOUL Laid,	Prof.,	Université de Biskra,	Examinateur
M. REZEG Khaled,	MCA,	Université de Biskra ,	Examinateur
M. LABOUDI Zakaria,	MCA,	Université de Oum El Bouaghi,	Examinateur
M. AOUAG Sofiane,	MCA,	Université de Batna 2,	Examinateur

Remerciements

Dans un premier lieu, je tiens à exprimer toute ma reconnaissance à mon directeur de thèse, Le professeur Seghir Rachid. Je le remercie de m'avoir encadré, orienté, beaucoup aidé et conseillé.

Sans oublier de rendre hommage à mon ex-directeur de thèse, le Professeur Belattar Brahim, qui nous a quitté prématurément dans un accident tragique. Que notre dieu, le miséricordieux, l'accueille dans son vaste paradis.

Je tiens à remercier tous les membres de jury, d'avoir accepter l'évaluation de ce travail, en sacrifiant leur précieux temps surtout en cette période de pandémie du COVID19.

J'adresse mes sincères remerciements à tous les intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches, et particulièrement, mes collègues enseignants du département informatique de Batna.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

Le calcul volontaire (CV) est devenu une technique relativement mature de calcul distribué. Son principe consiste à exploiter le temps de repos des machines ordinaires connectées à internet et avec le consentement de leurs propriétaires. Les applications cibles sont généralement des projets scientifiques nécessitant un temps et des ressources énormes de calcul. Les plateformes de calcul volontaire existantes soulèvent plusieurs défis concernant les différentes fonctions qui doivent être assurées. Dans une première partie de notre travail, nous essayons d'apporter des solutions à deux défis inhérents au CV ; Le premier concerne l'implication de volontaires, qui constitue le maillon faible de ce type de systèmes. Nous proposons un nouvel environnement de calcul volontaire social (SVCE) intégrant les fonctionnalités récentes de Facebook afin d'impliquer davantage de volontaires. Le deuxième problème, que nous avons abordé, est celui de l'ordonnancement des tâches dans les systèmes de CV. Pour cela, nous proposons un algorithme qui consiste à générer, pour chaque volontaire, un nombre de tâches élémentaires dont le coût d'exécution reflète la capacité de calcul momentanée des ressources disponibles des volontaires, la validité de notre algorithme est illustrée expérimentalement. Dans la deuxième partie de notre thèse, nous nous penchons sur la simulation distribuée, nous proposons à cet effet un environnement de simulation (VolSIM) qui combine les techniques liées au calcul volontaire afin d'exécuter des simulations qui requièrent des ressources conséquentes.

Mots Clefs : Calcul Volontaire, Réseau Social, Politique d'ordonnancement, Simulation distribuée.

Abstract

Volunteer Computing (VC) has become a relatively mature technique of distributed computing. It is based on exploiting the idle time of ordinary machines connected to the internet with the consent of their owners. The target applications are generally scientific projects requiring a huge amount of time and computational resources. Existing volunteer computing platforms raise several challenges concerning the different functions that a VC system must ensure. In the first part of our work, we attempt to bring solutions for two defeats inherent to VC. The first one is the involvement of volunteers, which is the weak link in this type of systems. Indeed, less volunteers means a decrease in computational resources, which has a detrimental effect on the global performances. To cope with this, we propose a new social volunteer computing environment (SVCE) which integrates recent Facebook functionalities in order to involve more volunteers. The second problem we have addressed is the scheduling of tasks in VC systems, which aims to optimize the use of the resources of the available volunteers. For this purpose, we propose an algorithm which consists in generating, for each volunteer, a number of tasks whose cost of execution reflects the momentary computing capacity of the volunteer's available resources. The effectiveness of our algorithm is experimentally validated. In the second part of this thesis, we study the distributed simulation, we propose for this purpose a simulation environment (VolSIM) which combines techniques related to volunteer computing in order to execute simulations which require substantial computing resources.

Keywords : Volunteer computing, Social Network, Scheduling policy, Distributed simulation.

الملخص.

أصبحت الحوسبة التطوعية (VC) تقنية ناضجة نسبيًا للحوسبة الموزعة. مبدأها يرتكز على استغلال الوقت غير المستغل للآلات العادية المتصلة بالإنترنت وذلك بموافقة أصحابها. التطبيقات المستهدفة بشكل عام هي مشاريع علمية تتطلب وقتًا وموارداً حوسبة هائلة. تثير منصات الحوسبة التطوعية الحالية العديد من التحديات فيما يتعلق بالوظائف المختلفة التي يجب أن يوفرها هذا النوع من المنصات. في الجزء الأول من عملنا ، نحاول تقديم حلول لتحديين مهمين في الحوسبة التطوعية؛ الأول يتعلق بدرجة مشاركة المتطوعين التي تعتبر الحلقة الأضعف في هذا النوع من الأنظمة. نقترح بيئة حوسبة اجتماعية تطوعية جديدة (SVCE) تدمج وظائف Facebook الحديثة من أجل إشراك المزيد من المتطوعين. التحدي الثاني الذي تناولناه هو جدولة المهام. لهذا الغرض، نقترح خوارزمية تتمثل في إنشاء عدد من المهام الأولية لكل متطوع تعكس تكلفة تنفيذها القدرة اللحظية للحساب المتعلقة بموارد المتطوعين، ويتم توضيح صحة الخوارزمية تجريبيا. في الجزء الثاني من أطروحتنا، نلقي نظرة على المحاكاة الموزعة، نقترح لهذا الغرض بيئة محاكاة (VolSIM) تجمع بين التقنيات المتعلقة بالحساب التطوعي من أجل تنفيذ المحاكاة التي تتطلب موارد حساب كبيرة.

الكلمات الرئيسية: الحساب التطوعي ، الشبكة الاجتماعية ، سياسة الجدولة ، المحاكاة الموزعة.

Table des matières

Table des figures

Liste des tableaux

Abréviations

AC	Automate Cellulaire 84
BLAS	Basic Linear Algebra Subprograms 7
BOINC	Berkeley Open Infrastructure for Network Computing . . 1, 11, 12, 14, 15, 18
BOT	Bag Of Task 18
BSP	Bulk Synchronous Parallel 9
CUDA	Compute Unified Device Architecture 7
CV	calcul volontaire . iii, 1, 7, 10–12, 15, 18, 35–39, 41, 42, 46–48, 92
DARPA	Defense Advanced Research Project Agency 63
DHT	Distributed Hash Table 18
DLT	Divisible Load Theory 40
DMSO	Defense Modeling and Simulation Office 63
FBML	FaceBook Modeling Language 30
FCFS	First Comming First Served 12
FOM	Federation Object Model 65–67, 75
GIMPS	Great Internet Mersenne Prime 7
GVT	Global Virtual Time 63
HADOOP	High Availability Distributed Object Oriented Platform . . . 10
HLA	High Level Architecture 1, 53, 63–67, 72, 74, 75, 77
HORB	Hirano Object Request Broker 12
KB	Knowledge Based 47
M-W	Master-Worker 76
MIM	Management Initialization Module 66
MOM	Management Object Model 66
MPI	Message Passing Interface 9
NOW	Network Of Workstations 7, 10
NUMA	Non Uniform Memory Access 7
OMT	Object Model Template 63, 65–67, 75
OpenMP	Open Multi-Processing 7, 9
OSN	Online Social Network 25
P2P	Peer To Peer 18
PDES	Parallel Discret Event Simulation 76
PL	Processus Logique 57, 60, 62, 63
POSIX	Portable Operating System Interface 9
PVM	Parallel Virtual Machine 9
RTI	RunTime Infrastructure 63–67, 69, 74, 75
SD	Simulation Distribuée 1, 53, 55–57, 62, 63, 75

SIMNET Simulation Networking Programm 63
SNA Social Network Analysis 27
SOM Simulation Object Model 65–67, 75
SVCE Social Volunteer Computing Environment 1
TASP Task-Adapted Scheduling Policy 35, 40, 47, 48
TSO Time Stamp Order 67
UMA Uniform Memory Access 7
VolSIM Volunteer computing based SIMulation 1, 91
VPS Virtual Private Server 37

Introduction

Le calcul volontaire CV est une forme de calcul distribué [Nouman Durrani et Shamsi, 2014], qui permet à des participants (appelés également volontaires, bénévoles ou encore utilisateurs publiques) d'offrir bénévolement une partie de leurs capacités de calcul dans le but d'aider à l'exécution de projets. Ces projets, généralement à caractère scientifique, nécessitent des ressources de calcul conséquentes dont l'exécution sur des ordinateurs ou même des clusters d'ordinateurs peut s'étaler sur un temps exorbitant. Les projets de calcul volontaire, stockés et gérés par des serveurs de CV, sont divisés en plusieurs unités appelées tâches qui seront distribuées, selon une politique d'ordonnancement donnée, aux bénévoles qui les exécutent et renvoient le résultat vers les serveurs. Ces derniers, et après les vérifications qui s'imposent valident, stockent et éventuellement consolident lesdits résultats et réitèrent le processus pour les tâches suivantes des projets de calcul. Les environnements de calcul volontaire diffèrent des autres types d'environnements tels ceux destinés au calcul sur les grilles et/ou les clusters par le caractère dynamique de la disponibilité des ressources de calcul. En effet, la fréquence élevée des adhésions et d'annulations de participation (inopinés suite à des erreurs ou voulus) de la part des bénévoles conduit à mettre en place des fonctionnalités spécifiques en terme de ré-attribution des tâches défaillantes à d'autres volontaires, de vérification des résultats et de sécurité.
L'utilisation des systèmes de calcul volontaire est relativement ancienne et date depuis les années 90. La plateforme Berkeley Open Infrastructure for Network Computing (BOINC) [Anderson, 2004], qui est un développement du fameux projet seti@home, a permet la vulgarisation du concept de CV en faisant fonctionner plusieurs projets scientifiques. Depuis, d'autres améliorations et idées on été proposées pour booster l'utilisation du CV comme technique de calcul écologique et cost-free et qui peut présenter potentiellement une alternative aux énergivores grilles et clusters de calcul traditionnels.
Nous nous intéressons, dans une première partie de cette thèse, à deux défis inhérents aux systèmes de CV, le premier est l'accroit du nombre de volontaires qui constitue le talent d'Achille de ce type de systèmes ; en effet, la baisse du nombre de participants ralentis d'une manière considérable l'exécution des projets de CV nécessitant parfois des mois de calcul. Pour ce problème, nous proposons un nouvel environnement tirant profit de la densité des liens du réseau social Facebook afin d'impliquer d'avantages de volontaires potentiels via les liens d'amitié dans ce réseau qui peuvent être vus aussi comme des liens

de confiance plus persuasifs pour la participation dans les projets de CV.
Dans la deuxième partie de notre travail, nous nous sommes penchés sur la simulation et particulièrement la simulation distribuée. La simulation est une vieille technique de modélisation du comportement des systèmes existants (où l'expérimentation est impossible ou bien couteuse) ou à concevoir de nouveaux systèmes. Son utilisation dans divers domaines particulièrement les modèles climatiques et aéronautique a fait ressentir un besoin en ressources de calcul colossales. Le standard High Level Architecture (HLA) [IEEE, 2010b] fixe les modalités et les règles de distribution de simulations. Dans ce standard, plusieurs simulations, appelées les fédérés, peuvent inter-opérer au sein d'une fédération HLA qui peut être vu comme un bus logiciel offrant les fonctionnalités de publication/abonnement aux fédérés afin de communiquer leurs états. Cette architecture sera étalée dans le quatrième chapitre. La SD est adaptée, dans un deuxième travail, afin d'inclure l'abondance théorique des ressources qu'un système de CV peut mettre à disposition.
Cette thèse est organisée en deux grandes parties ; la première, portant sur le calcul volontaire, englobe les trois premiers chapitres. Quant à la deuxième partie, traitant la Simulation Distribuée (SD) et l'utilisation du CV dans l'exécution des SD, elle est répartie sur le quatrième et le cinquième chapitre.
Dans le premier chapitre, le calcul volontaire est positionné au sein de la discipline du calcul parallèle et distribué. Une clarification nécessaire sur la différence entre le CV et les grilles de calcul et les clusters est relatée à cause de la confusion apparente entre ces deux types de calcul. En deuxième lieu, Nous présentons quelques travaux afférant au calcul volontaire et en relation avec le thème traité.
Le concept de réseau social est présenté dans le deuxième chapitre. Après une brève présentation littéraire incluant la modélisation de ces réseaux en termes de nœuds et de lien, nous mettons l'accent, dans une section à part, sur le célèbre réseau social sur internet Facebook et plus précisément l'aspect technique de son intégration dans les applications informatiques suivant les dernières spécifications de ses concepteurs.
Dans le troisième chapitre, après la formalisation du modèle de calcul volontaire communément utilisé, nous présentons dans un premier volet notre environnement Social Volunteer Computing Environment (SVCE) (Social Volunteer Computing Environment) comprenant son architecture, ses composants, son fonctionnement coté serveur et la partie client s'exécutant sur les machines des bénévoles. Dans le deuxième volet, nous commençons par un petit état de l'art sur les politiques d'ordonnancement utilisées dans les systèmes de CV, après, nous présentons notre propre algorithme d'ordonnancement baptisé TASP (Task-Adapted Scheduling Policy). Une expérimentation de notre environnement est décrite dans le quatrième chapitre. Un projet de calcul volontaire pilote consistant au calcul de la suite de Collatz, formalisée au début du chapitre, est mis en

œuvre dans notre environnement. Les résultats de comparaison entre notre algorithme TASP et deux algorithmes d'ordonnancement largement utilisés à savoir FCFS (politique aveugle) et "World Community Grid" (politique informée) illustrent l'optimalité de l'algorithme proposé.
Nous introduisons, dans le quatrième chapitre, une grande discipline de l'informatique : la simulation distribuée. Après les généralités sur la simulation à évènements discrets et ses problématiques classiques telles que la causalité et la cohérence, nous passons à la simulation distribuée. Nous mettons l'accent sur le standard HLA dans un but de vulgarisation pédagogique et d'explication des principaux concepts de la HLA puisés directement de la norme officielle. Nous terminons le chapitre par quelques travaux récents sur la simulation distribuée et le standard HLA.
Dans le cinquième chapitre, nous proposons une conception d'un environnement de simulation à base du calcul volontaire Volunteer computing based SIMulation (VolSIM) où nous tentons de mixer les concepts de la première partie de notre travail (le CV) avec la simulation distribuée. Nous étudions en particulier la possibilité d'exécuter des unités de simulation issues de la décomposition d'une simulation complexe sur des machines de volontaires avec discussion des problématiques liées à ce mixage.
Nous terminons la présente thèse par une conclusion en présentant les futurs axes de recherche s'inscrivant dans la continuité du présent travail.

Première partie

Le Calcul Volontaire

Chapitre 1

Calcul Volontaire : Positionnement et état de l'art

1.1 Introduction

Dans ce chapitre, le concept de calcul volontaire CV est introduit, partant de l'origine de la terminologie, passant par le positionnement du CV par rapport aux autres formes de calcul parallèle et distribué en le comparant avec les types de calcul les plus proches en l'occurrence les clusters et les grilles (réseau de stations). Enfin, nous présentons l'essentiel des travaux de recherche portants sur le calcul volontaire durant ces dernières années.

1.2 Origines du calcul volontaire

Great Internet Mersenne Prime (GIMPS) [Mersenne Research, 1996] est le premier projet où un calcul distribué sur des participants bénévoles est utilisé. Une version optimisée du test de primalité de Lucas-Lehmer (Le nombre de Mersenne $M_p = 2^p - 1$ est premier s'il divise le terme Sp-2 de la série de Lucas-Luhmer définie par $(S_p = S_{p-1}^2 - 2; S_0 = 4)$ écrite en assembleur est envoyée par email à des participants qui l'exécutent et retournent les résultats toujours par email.
Le terme de « calcul volontaire » appelé aussi « calcul bénévole » est inventé par Luis F.G Sarmenta 1996 au MIT. L'auteur défini le concept par « permettre une formation d'un réseau de calcul parallèle haute performance facilement, rapidement et à moindre coût » [Sarmenta, 2001]. Dans une autre définition relatée par le même auteur : « une nouvelle forme de calcul qui permet aux utilisateurs connectés à Internet de mettre en commun leurs ressources informatiques et travailler ensemble pour résoudre de gros problèmes de calcul que personne ne peut les résoudre tout seul ».
Le calcul volontaire fonctionne en permettant aux personnes de partager le temps de repos de leurs ressources de calcul (durant les mises en veille par exemple) grâce à un logiciel facile à déployer et qui ne nécessite pas d'expertise ou connaissances techniques préalables. De ce point de vue, le CV peut être appréhendé comme une généralisation de la technique du vol de cycle, utilisée dans les processeurs [Goossens, 2002], au niveau des machines des bénévoles en exploitant le résidu de temps d'inactivité de ces derniers.
Le concept du CV est mis en valeur par son inventaire dans sa plateforme "Bayanihan" [Sarmenta *et al.*, 1998] qui consiste à un environnement de calcul volontaire développé en java avec un client Web qui se résume à des applets rapatriées d'un serveur et exécutant des tâches d'un projet.

1.3 Revue des modèles d'architecture parallèles

Dans la discipline du calcul parallèle et distribué, plusieurs taxonomies ont été proposées selon des critères très variés. Une taxinomie donnée implique une conception architecturale plus des modèles et des outils de programmation. Une des premières classifications fut celle de M.Flynn [Flynn, 1972] qui décrit quatre catégories de machines selon les combinaisons possible de l'unicité ou la multiplicité des flots d'instructions et de données : SISD, MISD, SIMD et MIMD [1]. SISD décrit l'architecture classique de Von Neumann quant à MISD elle correspond à des architectures très rares tels que les processeurs pipelines ou une donnée subit plusieurs traitements en cascade donnant l'impression de l'exécution de plusieurs instructions sur la même donnée. Dans l'architecture SIMD, la même instruction est appliquée sur plusieurs données dans des unités de traitement séparées. Son implémentation matérielle est réalisée sous diverses formes, la forme la plus répandue est celle des instructions SIMD intégrées dans la plupart des processeurs actuels qui orchestrent nos ordinateurs. Intel®, à titre d'exemple, propose depuis l'année 2000, dans sa gamme de processeurs x86/64, un ensemble d'instructions dénotées SSE/AVX [2] permettant d'exécuter des opérations sur des registres spécifiques allant jusqu'à 256 bits pour AVX2 et pouvant stocker des vecteurs de valeurs. Les machines vectorielles comme les fameux Cray 1&2 sont une deuxième implémentation du paradigme matériel SIMD. Au niveau logiciel, plusieurs Framework adoptent la philosophie SIMD tels que Open Multi-Processing (OpenMP), Compute Unified Device Architecture (CUDA) et Basic Linear Algebra Subprograms (BLAS).

MIMD est le modèle le plus générique d'architectures parallèles, il couvre pratiquement toutes les architectures actuelles. On distingue, dans cette catégorie, deux architectures : les multiprocesseurs et les multi-ordinateurs.

Les multiprocesseurs sont des machines avec plusieurs processeurs partageant une mémoire qui se présente logiquement comme un espace d'adressage unique pour tous les processeurs. Ils se distinguent par leurs types de mémoire, on parle de Uniform Memory Access (UMA) si l'accès est uniforme pour tous les processeurs et de Non Uniform Memory Access (NUMA) si la mémoire est physiquement distribuée. Les multi-ordinateurs sont des unités de calculs indépendantes avec des mémoires individuelles, ces entités sont reliées par un réseau d'interconnexion qui peut être propriétaire, un réseau local ou un réseau large (Internet). Le seul moyen d'échange de données dans ce type d'architectures est le passage de message. Les unités ou nœuds dans les multi-ordinateurs peuvent être homogènes avec la même image système et complètement dédiées au calcul avec un nœud maitre qui supervise le calcul, on parle de multi-ordinateur asymétrique ou Cluster. Comme elles peuvent être hétérogènes, indépendantes et dont le calcul est une activité secondaire qui peut être

1. (Single,Multiple) X (Instruction,Data).
2. Streaming SIMD Extensions qui a évolué ensuite à AVX ((Advanced Vector eXtensions)

initié par n'importe quelle unité. Dans ce cas, on parle de multi-ordinateurs symétrique ou réseaux de stations Network Of Workstations (NOW). La figure 1.1 illustre les modèles d'architectures parallèles décrites précédemment.

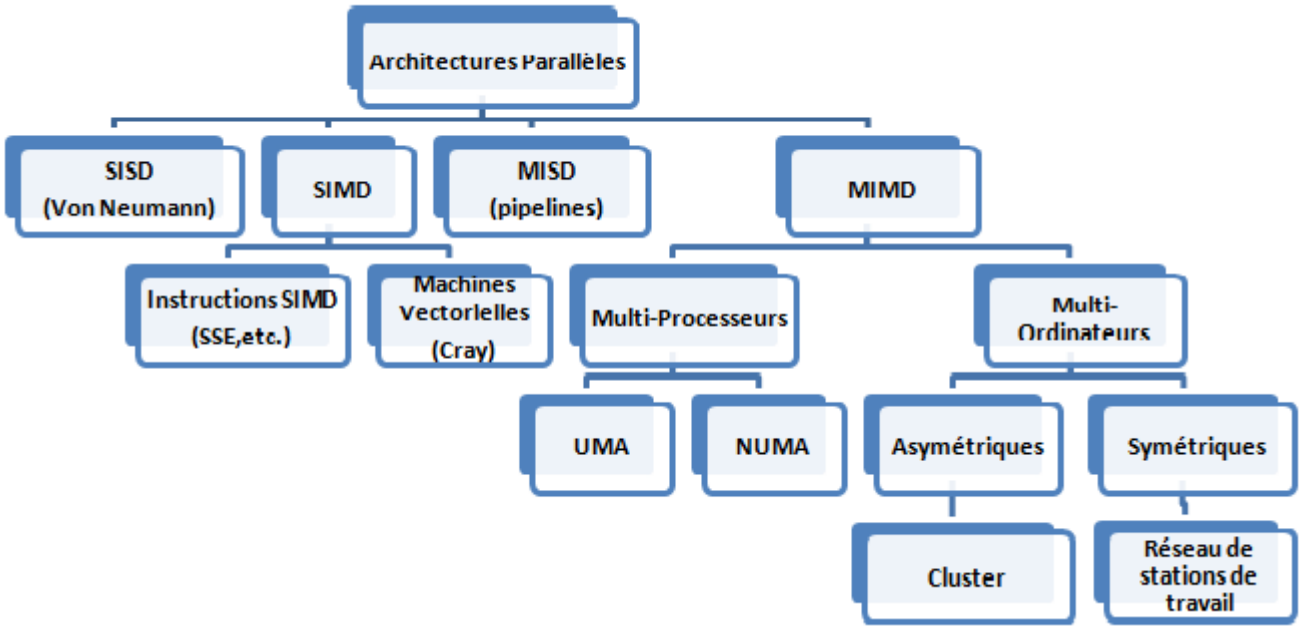

FIGURE 1.1 – Modèles d'architectures parallèles.

1.3.1 Modèles abstraits de calcul parallèle

Un modèle abstrait de calcul parallèle décrit une organisation virtuelle des unités d'exécution (processeurs logiques) ainsi que le type d'échange de données entre elles sans se préoccuper de la configuration matérielle de l'implémentation. Les modèles abstraits sont répartis en deux grandes classes :

— **Modèles à mémoire partagée :** Les unités de calcul (processeurs logiques) partage un espace d'adressage unique, l'échange se fait via des variables en mémoire avec les mécanismes de synchronisation issus de la programmation concurrente (sémaphores, moniteurs etc.). Dans cette classe on peut citer les frameworks OPENMP, PThreads (standard Portable Operating System Interface (POSIX) des threads Unix), Thread Java, etc.

— **Modèles à passage de messages :** Dans ce modèle, les unités de calcul disposent d'espaces d'adressages privés et disjoints. Elles sont reliées par un réseau d'interconnexion selon une topologie logique (Bus, Hypercube, Tore,etc.). L'échange de données se fait exclusivement via des routines de communication synchrones et

asynchrones et qui peuvent porter sur deux unités (envoi et réception simple) ou plusieurs unités (communication collective telle que la diffusion). D'autres mécanismes de synchronisation peuvent être ajoutés tels que les Rendez-vous multiples appelées barrières de synchronisation. Message Passing Interface (MPI),Parallel Virtual Machine (PVM) et Bulk Synchronous Parallel (BSP) sont des bibliothèques représentatives de cette classe de modèles abstraits.

D'autres modèles se focalisent sur le partitionnement des traitements et des données sans faire référence aux unités d'exécution logiques. Dans MapReduce par exemple, les données sont partitionnées par une première fonction map en associant à chaque donnée une clé. Après une série de traitements map, les résultats sont agrégés par une deuxième fonction reduce, l'agrégation se fait par rapport aux clés associées aux données. Hadoop [3] utilise le paradigme MapReduce combiné avec un système distribué de gestion de fichiers High Availability Distributed Object Oriented Platform (HADOOP) afin de pouvoir gérer des quantités colossales de données réparties sur les clusters.

1.4 Positionnement du calcul volontaire

Le calcul Volontaire peut être vu comme un modèle abstrait de calcul avec une architecture hybride entre le clustering et les réseaux de stations NOW ; il est asymétrique du fait que le calcul volontaire se déroule selon le modèle maître esclave, où le serveur et le client de CV supervisent l'exécution des tâches sur les machines des bénévoles (attribution, suspension, reprise, annulation, etc.), en contre partie, les unités de calcul sont hétérogènes, autonomes et ne sont pas dédiées exclusivement aux calcul. la table 1.1 illustre une comparaison entre ces trois types de modèles de calcul.

TABLE 1.1 – Comparaison entre le CV,le Clustering et les NOW.

	Clustering	**Calcul volontaire**	**Réseaux de stations**
Distribution géographique	Regroupé	Fortement dispersé	Dispersé
Connectivité	LAN, fortement couplé	Internet, faiblement couplé	Internet, faiblement couplé
Architecture	Asymétrique	Asymétrique	Symétrique
Type d'unités de calcul	Homogènes	Hétérogènes	Hétérogènes
Modèle commercial	Payant / propriétaire	Libre	Libre, Interorganisation

3. Apache™ Hadoop® : Framework utilisant le paradigme MapReduce dans les clusters.

1.5 Exigences et composants des systèmes de CV

Tout système de calcul volontaire de qualité doit satisfaire un certain nombre d'exigences [Nouman Durrani et Shamsi, 2014], entre autres, on peut citer :

1. Répartition efficace des tâches sur les volontaires,
2. Robustesse, i.e. détection des volontaires défaillants et reprise des tâches non exécutées,
3. Confiance au middleware,
4. Sécurité du fonctionnement des machines des volontaires.

Un système de CV est composé d'un ensemble de composants s'exécutant au niveau d'un ou plusieurs serveurs, et un logiciel client qui est fourni à la demande aux participants. Les fonctions essentielles que le système doit assurer au niveau du serveur sont :

— **L'ordonnancement :** consiste à l'assignation des tâches aux volontaires ; l'ordonnanceur prends en entrée un ensemble de tâches, éventuellement pondérées par leurs coûts d'exécution, et un ensemble de ressources correspondant aux machines des volontaires connectées à un instant donné. selon une politique d'ordonnancement donnée, les tâches sont assignées aux différentes ressources de CV.

— **Validation et assimilation :** Une tâche exécutée convenablement (retour de résultats corrects) par un volontaire est une tâche validée. Afin de confirmer l'exactitude des résultats, des systèmes, tel BOINC, procèdent à son réexécution par un ou plusieurs autres volontaires. Les différents résultats sont comparés entre eux et l'un est pris comme résultat canonique (de référence). Un résultat d'une tâche n'est validé que s'il est comparable au résultat canonique. Une tâche avec un nombre donné de résultats valides est dite assimilée.

— **Gestion d'état des tâches :** Les tâches possèdent un cycle de vie dans les systèmes de CV, elles passent de l'état « crées » à l'état « assignées » lorsqu'elles sont envoyées aux volontaires, comme précisé dans la fonction de validation et assimilation, elles passent à l'état « validées » puis « assimilées ». Dans certains cas, elles sont « supprimées » à fin de purger la base de données du serveur. Tout système de CV doit assurer cette fonction de gestion du cycle de vie des tâches. Cette dernière doit garder, à tout moment, la trace de l'état de toutes les tâches dans le système.

— **Gestion de stockage :** En général, les systèmes de CV stockent et génèrent des volumes très importants de données, ce qui alourdi d'une manière considérable le fonctionnement des serveurs de données sous jacents. Une opération d'archivage et de purge de ces dernières s'avère nécessaire.

— **Code client :** Le code client doit assurer d'une part la supervision de l'exécution des tâches au niveau des ressources des bénévoles ; par exemple il doit lancer

l'exécution durant les temps de repos de la machine et la suspendre dans le cas échéant. D'autre part, il assure la communication avec le serveur CV qui comprend entre autre la réception des tâches à exécuter, l'envoi des résultats et la signalisation de l'état d'avancement des travaux en cours.

1.6 Travaux y afférents

Depuis l'introduction du CV, ses différents aspects ont fait l'objet d'une multitude de travaux de recherche vu son attraction comme outil de calcul à faible coût.
Une revue des différents concepts, défis et solutions liés au calcul volontaire sont détaillés dans [Nouman Durrani et Shamsi, 2014]. Passant par une intéressante analyse statistique sur la nature des volontaires à savoir volontaire à domicile, au travail ou aux établissements académiques [Toth et Finkel, 2009] ainsi que la disponibilité des ressources réparties par type de volontaire. Cette analyse permet éventuellement d'adapter certaines fonctionnalités du CV tel que l'ordonnancement. Les auteurs ont répertorié également un ensemble de caractéristiques des ressources des volontaires tel que le type de processeur, la mémoire, la capacité de stockage, le système d'exploitation, la bande passante, la disponibilité, la robustesse, etc. Ces caractéristiques peuvent être combinées avec une pondération adéquate afin d'évaluer quantitativement les capacités de calcul d'une ressource. Les politiques d'ordonnancement communément utilisées dans le CV ont été catégorisées en deux grandes classes : les politiques aveugles et les politiques informées. Dans les politiques aveugles(First Comming First Served (FCFS), aléatoire, locale, etc.), les informations sur les capacités des volontaires ne sont pas prises en considération tandis que les politiques informées utilisent ces informations pour attribuer les tâches aux volontaires adéquats. Les auteurs présentent aussi l'aspect de sécurité dans les CV et terminent par quelques problèmes ouverts comme l'hétérogénéité des ressources, la distribution de la charge de calcule, l'indisponibilité des résultats intermédiaires.
Le calcul volontaire est introduit dans le système Bayanihan [Sarmenta *et al.*, 1998], [Sarmenta, 2001] ; un système développé en Java offrant un client Web utilisable par les navigateurs au niveau des machines des participants et qui communiquent via le protocole Hirano Object Request Broker (HORB). De plus, l'auteur a ajouté un autre protocole basé sur le modèle BSP qui permet la communication entre les clients s'exécutant au niveau des bénévoles ce qui enrichi le modèle maitre-esclave prévu initialement dans le CV.
BOINC [Anderson, 2004], [Anderson et Reed, 2009] fut surement la plateforme phare du calcul volontaire. Elle est constituée d'une infrastructure (serveurs) hébergeant plusieurs

projets. Seti@home [4], Folder@home [5] , PrimeGrid [Bethune, 2015], etc. tournent sous la plateforme BOINC. Les projets sont divisés en plusieurs petites tâches appelées WorkUnit écrites et destinées pour divers systèmes (Windows/Linux/Macosx/Android/GPU). La plateforme est constituée d'un ordonnanceur, d'un validateur, d'un purgeur, d'un transitionneur et d'autres modules utilitaires de statistiques. Les clients doivent télécharger et exécuter sur leur machine un client BOINC qui représente l'environnement d'exécution coté client responsable de l'interaction avec le serveur BOINC (téléchargement des WorkUnits, Communication de l'avancement et le renvoi des résultats). Le Client BOINC récupère le temps de repos de la machine hôte (temps de veille / arrière plan de faible priorité) pour l'exécution des tâches selon une politique d'ordonnancement locale. La figure 1.2 montre les différents composants de BOINC.

Le client BOINC n'a besoin de communiquer avec le serveur que durant les phases de

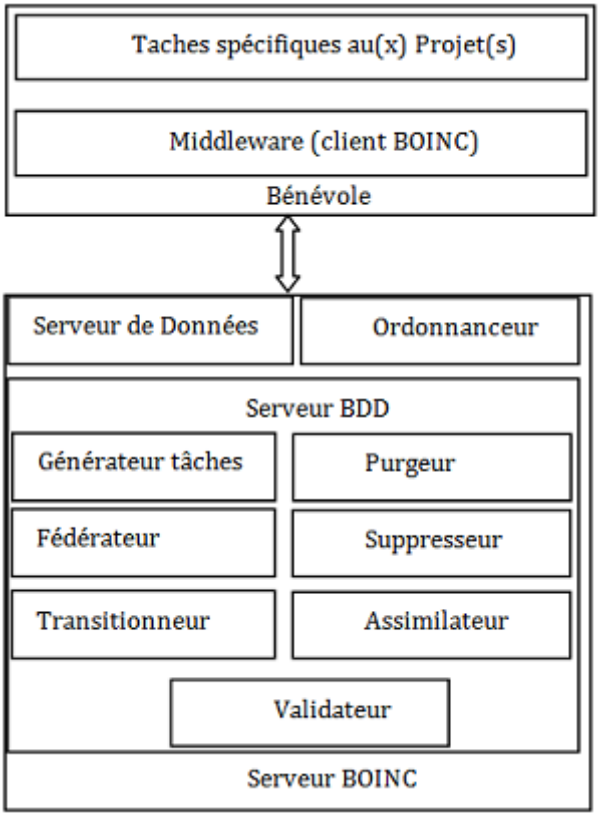

FIGURE 1.2 – Architecture de BOINC [Anderson, 2004].

récupération de tâches et de renvoi des résultats, l'exécution des tâches se fait indépendamment au niveau de la machine du volontaire. BOINC associe à chaque tâche un délai maximum d'exécution, au-delà de ce dernier et en l'absence de confirmation de résultat par le client, la tâche est déclarée interrompue et doit être relancée.

4. Seti acronyme de Search for ExtraTerrestrial Intelligence. Initié en 1999, en 2005 il adopte BOINC comme plateforme de calcul.

5. Folder@home : Projet fonctionnant sous BOINC et porte sur l'analyse de la structure des molécules.

De plus, les résultats d'exécution d'une tâche doivent être validés par au moins deux résultats identiques. Le concepteur du projet doit définir le nombre de résultats valides nécessaire pour l'assimilation de la tâche. BOINC comporte un validateur par défaut qui compare les résultats bit-à-bit, le créateur de projet peut définir son propre validateur de résultats afin d'introduire une validation spécifique à son projet. La figure 1.3 montre un schéma de validation/Assimilation ; la tâche est validée et assimilée après deux résultats identiques, d'autres seuils peuvent être utilisés.

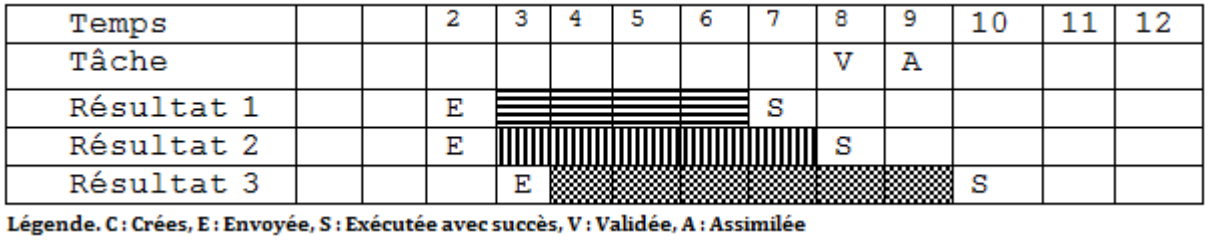

FIGURE 1.3 – Validation et assimilation d'une tâche dans BOINC.

Par défaut, BOINC supprime les données en entrée de toute tâche assimilée. La plateforme archive les tâches et leurs résultats selon des critères fixés par le concepteur du projet (les plus anciennes, un nombre donné de tâches, etc.) dans des fichiers XML et les supprime de la base de donnée.
Sur le site des statistiques de la plateforme BOINC [6] , les statistiques des projets concernant plusieurs mesures sont maintenues à jour et alimentées continuellement. A titre d'exemple, La figure 1.4 montre la répartition des ordinateurs des volontaires par disponibilité, type de processeur et type de systèmes d'exploitation respectivement.

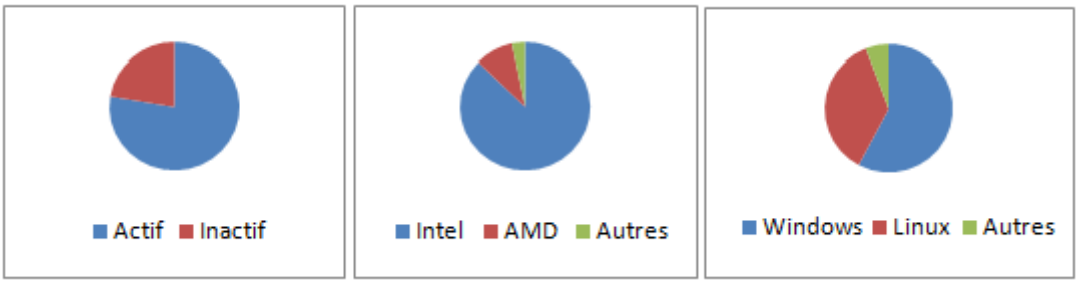

FIGURE 1.4 – Répartition des ordinateurs dans BOINC par activité/type processeur et type d'OS (au 25/01/2019).

La table 1.2 illustre quelques chiffres signalétiques relevés en date du 25/09/2018 quant à la table 1.3, elle montre la répartition du nombre d'utilisateurs, d'ordinateurs, d'équipes

6. https ://www.boincstats.com/

et de crédits sur les principaux projets tournant sous BOINC.
Nous avons également, relevé des mesures s'étalant sur une période de 60 jours (allant du

TABLE 1.2 – Ressources et puissance de calcul dans BOINC.

Total Crédits	36,086,944,658,612
Nombre d'opérations/sec	21,513 TeraFLOPS
Utilisateurs	4,610,047
Utilisateurs actifs	156,329 (3.39%)
Ordinateurs	957,161
Ordinateurs actifs	646,455 (67.54%)
Équipes	107,622
Équipes actives	15,322 (14.24%)

TABLE 1.3 – Répartition des bénévoles sur les projets BOINC.

Projet BOINC	**Utilisateurs**	**Ordinateurs**	**Équipes**	**Crédit total**
Global	4,610,047	957,161	107,622	36,086,944,658,612
Collatz Conjecture	62,756	5,715	1,715	1,745,879,133,770
Einstein@Home	466,233	57,133	11,615	644,522,486,702
PrimeGrid	346,077	87,095	3,095	780,649,736,646
GPUGRID	36,979	5,74	1,64	714,130,002,546
AmicableNumbers	3,758	13,175	238	82,353,386,750
SETI@Home	1,743,314	171,229	64,433	496,021,882,283
MilkyWay@home	221,56	24,58	4,402	374,097,597,062
World CommunityGrid	542,646	197,649	26,346	309,003,364,962
Enigma@Home	68,149	8,493	1,645	47,051,031,026
Asteroids@home	119,974	33,449	1,726	80,106,734,613
Citizen Science Grid	13,659	8,99	578	31,452,403,800
Moo! Wrapper	55,873	18,328	760	154,364,261,527
Rosetta@Home	1,286,300	69,624	11,207	71,887,202,838
ClimatePrediction	300,092	13,105	8,021	36,267,646,488
Universe@Home	24,924	12,954	647	15,726,803,619

27/07/2018 au 23/09/2018) concernant deux projets mathématiques « Collatz Conjecture » et « PrimeGrid ». Le premier projet est utilisé comme application pilote de notre environnement que nous allons décrire dans le chapitre 3.
Dans la Figure 1.5 on a en moyenne 3.6 nouvel utilisateur par jour pour le projet « Collatz Conjecture » contre 7.4 pour le projet « PrimeGrid », on remarque une petite variation sur un petit nombre de nouveaux participants.

La figure 1.6 montre une certaine stabilité pour le nombre d'utilisateurs actifs pour les deux projets (entre 1200 et 1400 pour la « Collatz Conjecture » et entre 5200 et 5400 pour le « PrimeGrid ».

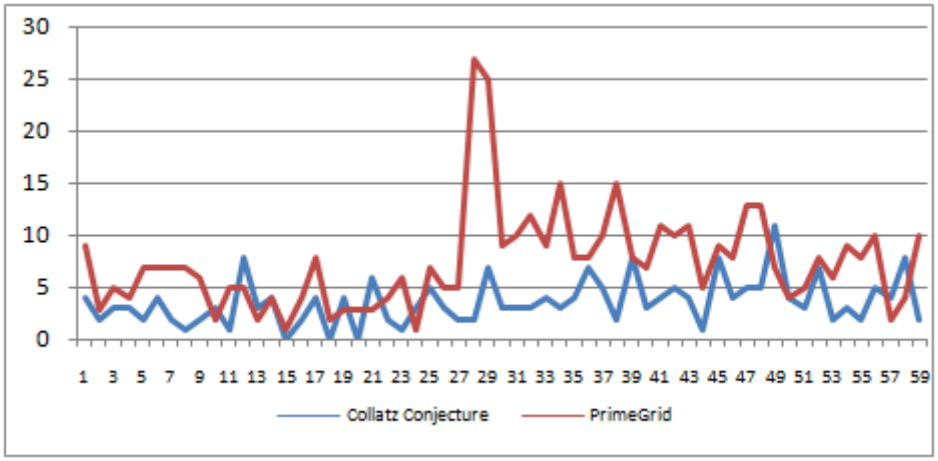

FIGURE 1.5 – Nouveaux Utilisateurs/ jour pour deux projets BOINC.

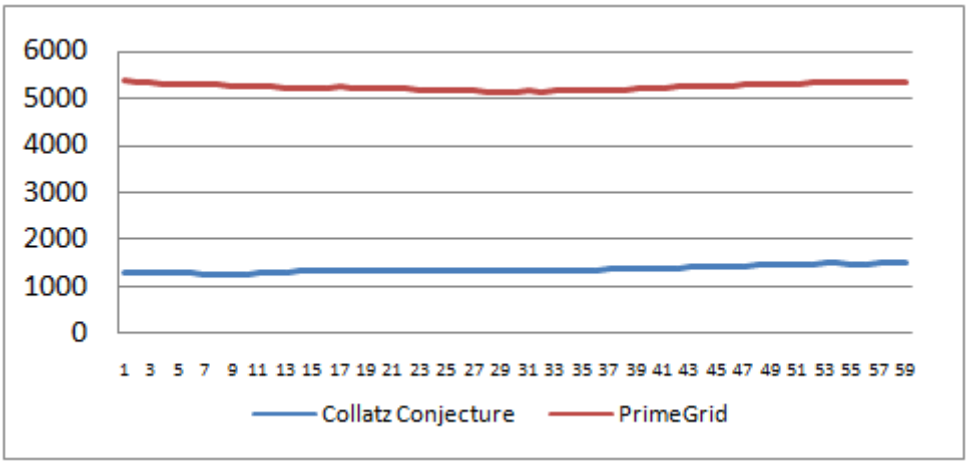

FIGURE 1.6 – Utilisateurs actifs/jour pour deux projets BOINC.

Plusieurs travaux portant sur la plateforme BOINC visant l'amélioration de cette dernière. BOINC étant un Framework libre, toute organisation peut l'utiliser pour mettre en place ses propres projets de calcul. La conception et le codage des projets sous cette plateforme peuvent s'avérer complexes et nécessitant une certaine maitrise technique d'un bon nombre d'outils (Mysql, Apache, PHP, C++, XML, etc.). Dans [Ries, 2013] l'auteur propose une extension d'UML 2.4 pour modéliser les projets de calcul haute performance basée sur une abstraction de BOINC. L'extension proposée « UML for BOINC » comprend des contraintes semi formelles, des notations graphiques et approches de génération de code de projets.

Plusieurs études statistiques sur des projets tournant sous BOINC ont fait l'objet d'une multitude de travaux ces dernières années. Dans [Bethune, 2015], le projet PrimeGrid, visant des challenges dans le domaine de la théorie des nombres tel que la recherche des grands nombres premiers, est détaillé. PrimeGrid comporte plusieurs sous projets les uns

terminés et les autres sont toujours en cours d'exécution tels que énumérés dans la table 1.4.

Les auteurs dans [Estrada *et al.*, 2009] se sont penchés sur l'impact d'utilisation de

TABLE 1.4 – Anciens et actuels Sous-projets PrimeGrid extraits de [Bethune, 2015].

Sous-projet PrimeGrid	**Début**	**Forme des nombres premiers cibles**
Twin Prime Search	11/2006	p et p+2
Cullen Prime Search	08/2007	$n * 2^n + 1$
Woodall Prime Search	08/2007	$n * 2^n - 1$
Proth Prime Search	02/2008	$k * 2^n + 1$
321 Prime Search	06/2008	$k * 2^n + 1$ (proth prime avec k=3)
Prime Sierpinski-Problem	07/2008	Prouver que 271129 est le plus petit nombre premier de Sierpinski
AP26 Search	12/2008	Recherche d'une séquence de 26 nombres premiers selon une progression arthmétique (terminé en 2010)
Sophie Germain Search	08/2009	paire de nombres premiers de la forme p et 2p+1 (Fusionné avec Twin Prime)
Seventeen or Bust	01/2010	Prouver que 78557 est le plus petit nombre de Sierpinski
The RieselProblem (TRP)	03/2010	Prouver que 509203 est le plus petit nombre de Riesel
Generalized Fermat Number (GFN)	01/2012	$b^{2^n} + 1$
SierpinskiRiesel base 5 problem	06/2013	Prouver que 159986 et 346802 sont les plus petits nombres de Sierpinski et riesel à base 5 respectivement
Extended SierpinskiProblem	06/2014	Prouver que 271129 est le deuxième plus petit nombre de Sierpinski

différentes politiques d'ordonnancement et d'autres paramètres sur la performance des calculs dans les projets BOINC. Partant de la difficulté de réaliser les tests sur les projets en production (en exécution) ou bien la difficulté de simuler de tels systèmes, les auteurs ont proposé un émulateur de BOINC qui l'ont nommé EmBOINC pour étudier à travers trois cas d'étude à savoir une comparaison des politiques d'ordonnancement, la variation de la redondance homogène (envoi de la tâche à plusieurs volontaires avec des ressources hardware/software homogènes) et la relation entre les niveaux de réplication et le taux d'erreurs. Les auteurs ont utilisé un simulateur à événements discrets avec des utilisateurs et des travaux simulés comme entités qui génèrent des évènements relatifs au fonctionnement du CV (tels que start, terminate, enqueue, callValidator, callAssimiltor, etc.). La figure 1.7 montre le paramétrage nécessaire au fonctionnement de EmBOINC.

Dans [Chorazyk *et al.*, 2017] les auteurs ont présenté un simple système de CV à base

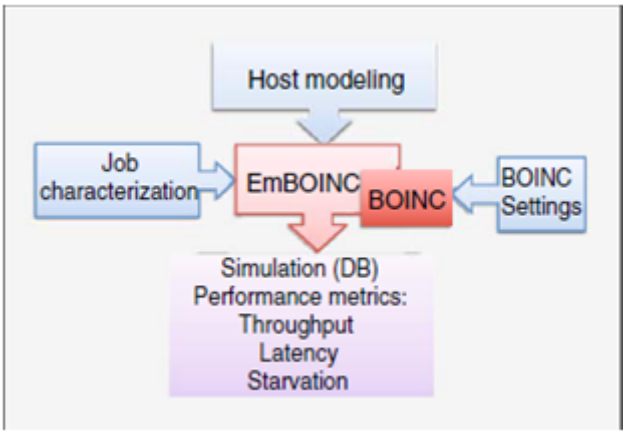

FIGURE 1.7 – EmBOINC :schéma de fonctionnement [Estrada *et al.*, 2009].

du Web s'appuyant sur les nouvelles fonctionnalités des browsers, notamment les Web-Workers introduits dans HTML5. L'environnement proposé se compose d'un :

- Noyau dont dépendent tous les autres modules. Il contient le modèle de données et gère la configuration de la plateforme,
- Module de persistance : responsable de la communication avec la base de données,
- Gestionnaire de tâches qui gère l'ordonnancement et le suivi des ressources des bénévoles.

L'environnement présenté est testé par une application qui analyse 2665 articles WikiPédia par des volontaires en subdivisant cette analyse respectivement en 64, 128, 256 et 512 tâches distribuées à des volontaires. Les résultats reportés par les auteurs illustrent bien l'intérêt du calcul volontaire via des accélérations intéressantes (ex : 27 pour 32 volontaires). L'avantage du système réside dans le fait qu'il se contente d'un browser sur la machine du volontaire (aucun middleware n'est requis au bénévole, ce qui représente un facteur d'attraction en particulier de point de vue sécurité). L'inconvénient de tels environnements est proche du celui des environnements classiques (BOINC) dans le sens où les bénévoles potentiels doivent être mis au courant de l'existence de ces applications d'une part, et d'autre part, le problème de sécurité induit par l'absence d'une relation de confiance incitant les volontaires à accéder au site de l'application et à exécuter les tâches dans leurs browsers.

Un environnement proche du précédent et baptisé par leurs auteurs WebCom est présenté dans [Morrison *et al.*, 2001]. WebCom est basé sur l'utilisation du modèle de graphe condensé (CG) qui est composé de nœuds correspondants soit à une opération atomique ou à un sous graphe d'opérations, aboutissant à une organisation hiérarchique où les nœuds terminaux correspondent toujours à des opérations atomiques. Cette dernière (l'organisation) est mappée sur les clients qui peuvent être des esclaves exécutant des

opérations atomiques ou des maitres contenant des sous graphes condensés. Les clients esclaves téléchargent et exécutent des applets Java à partir des nœuds maîtres en utilisant une configuration d'autorisation à ces applets. La figure 1.8 schématise l'organisation hiérarchique des clients dans WebCom.

Au niveau d'un client « Maitre », un thread est créé et se met à l'écoute d'un port

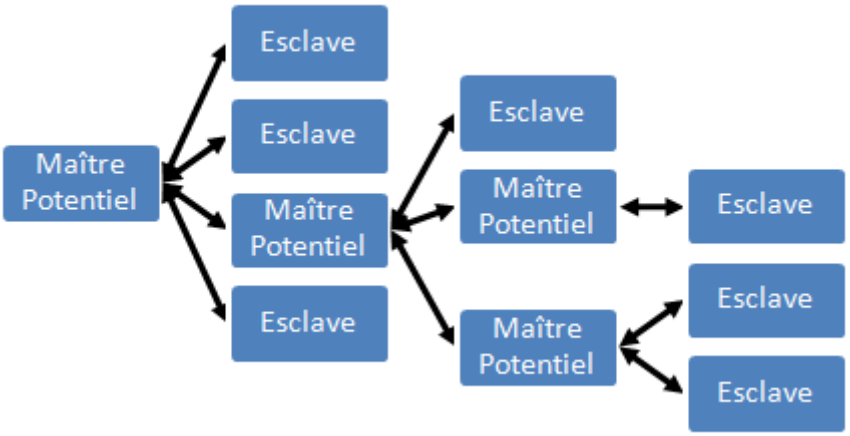

FIGURE 1.8 – Organisation des clients dans WebCom.

TCP, dans le cas ou un client « esclave » se connecte, l'applet contenant l'opération est téléchargée au niveau de ce dernier qui l'exécute et renvoi le résultat au « maitre ».
Malgré que l'organisation hiérarchique présente l'avantage d'équilibrage de charge entre les maitres, de notre point de vue, son inconvénient réside dans la complexité de gérer les calculs et les résultats nécessitant la mise en place de mécanismes de recouvrement très couteux dans un environnement de calcul en ligne extrêmement réactif. L'utilisation des applets est aussi discutable compte tenu des soucis de sécurité qu'elles entrainent.
Des utilisateurs non-experts peuvent exécuter des groupes de tâches gourmandes en temps CPU dans l'environnement GiGi-MR présenté dans [Costa *et al.*, 2012]. Dans leur environnement, l'utilisateur (un client ordinaire) peut créer localement un groupe de tâches Bag Of Task (BOT), et le soumettre à un système de calcul volontaire en l'occurrence BOINC. Le serveur, dans ce cas, est déchargé de la tâche de création des tâches. L'environnement présenté combine le CV avec le paradigme MapReduce selon le schéma de la figure 1.9. Les auteurs proposent également, l'utilisation de machines virtuelles au niveau des nœuds afin de contourner les comportements byzantins de ces derniers à cause de l'hétérogénéité du matériel et/ou logiciel.

L'approche utilisée est intéressante dans le sens où l'un des composant du CV est délégué à des clients ce qui permet d'éviter des goulots d'étranglement au niveau des serveurs. La combinaison du CV avec le paradigme MapReduce reste discutable à notre avis du fait que ce dernier est conçu pour fonctionner dans des clusters où un certain niveau de

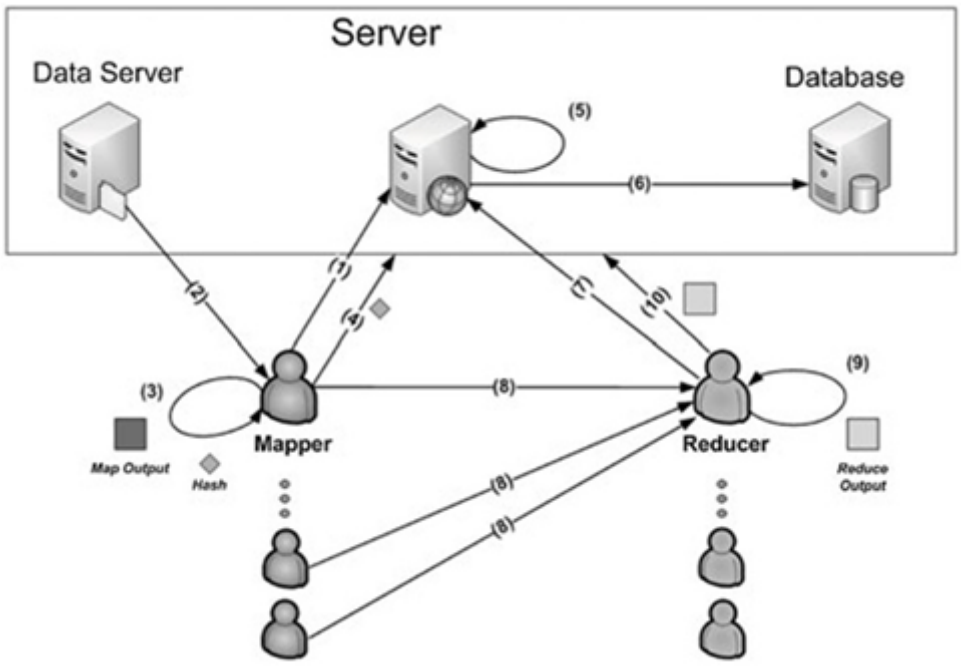

FIGURE 1.9 – GiGi-MR :Combinaison CV et MapReduce [Costa *et al.*, 2012].

fiabilité et de disponibilité des nœuds de calcul est plus au moins assuré. Malgré que les implémentations de MapReduce comme Hadoop prennent en charge la réattribution des tâches. Le fonctionnement dans un environnement de CV, caractérisé par une forte entropie de la disponibilité des volontaires, reste incertain et conduit éventuellement à une dégradation de performance importante vu la complexité des interactions dans de tels systèmes.
Afin de contourner le problème du goulot d'étranglement au niveau des serveurs de calcul volontaire, des travaux de recherche tentent de décentraliser certains éléments du contrôle en les déléguant aux machines des volontaires. Une approche dans ce sens est présentée dans [Li et Franzinelli, 2016]. Les auteurs utilisent, dans leur approche, le réseau de recouvrement Peer To Peer (P2P) Chord basé sur une table d'hachage distribuée Distributed Hash Table (DHT). Ils ont répertorié les éléments de contrôle à distribuer tels que énumérés dans le table 1.5.

Une application spécifique de rendu 3D baptisée RenderWeb 2.0 est développée et testée en combinant l'utilisation du VC et du réseau Facebook comme support publicitaire [MCMAHON et MILENKOVIC, 2011]. Les auteurs ont exploité les API fournies par le framework Blender [7] pour la distribution d'applications de rendu 3D sur Facebook, pour l'exécuter par des amis connectés au réseau social et retourner le rendu au soumissionnaire. Les auteurs ont utilisé blenderrenderer, module de rendu de Blender, à l'intérieur

7. Le framework est disponible et bien documenté sur http ://www.blender.org

TABLE 1.5 – Eléments à décentraliser et exigences [Li et Franzinelli, 2016].

VC coordination	**Master / Worker model**	**Decentralizing goal of P2P**	**Requirment on P2P overlay**
Project advertising	Centralized	Centralized	Maintenance of a pool of bootstrap nodes for each project
Project creation	Centralized	Centralized	Creation of a project overlay by the project owner ; creation of a bootstrap pool of the overlayby the advertising server.
Project participation	Centralized	Distributed	Use a random bootstrap node to join a project overlay.
Volunteer leave	Centralized	Hybrid	Bootstrap pool updating (centralized) ; job checkpointing, job state updating (distributed).
Volunteer crash	Centralized	Hybrid	Bootstrap pool updating (centralized) ; searching for crashed peers (distributed).
Job storage Centralized	Distributed	Distributed	job storage ; search and retrieval on any peer.
Job distribution and reassignment	Centralized	Distributed	Search for jobs and peers ; contact information ; peer communication.
Result return	Centralized	Distributed	Upload results onto the project overlay.
Result storage	Centralized	Distributed	Evenly distribute the uploaded results onto the project overlay.
Job progress monitoring	Centralized	Centralized	Search for available results and update project progress by the project owner.
Result collection and synthesis	Centralized	Centralized	Search and synthesize the available results by the project owner.

d'une applet java signée. RenderWeb utilise également les connections du réseau social Facebook pour exécuter le rendu en calcul volontaire. Le fonctionnement de RenderWeb est illustré dans la figure 1.10, la figure 1.11 montre le paramétrage et l'utilisation de RenderWeb dans Facebook.

L'approche utilisée est plus attractive en particulier pour les artistes et certains domaines scientifiques nécessitant des rendus 3D. L'utilisation de Blender à l'intérieur d'applets Java est à notre point de vue discutable compte tenu des problèmes de sécurité posés par les applets. Des systèmes ont imposé des limitations quant à l'exécution des applets dans les browsers (en imposant que l'applet soit signée par un organisme de confiance), d'autres systèmes (OSX/iOS) ont désactivé carrément leur utilisation. Néanmoins, nous nous

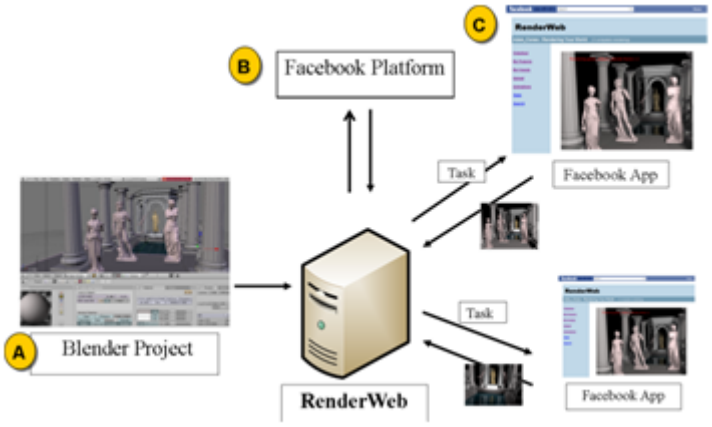

FIGURE 1.10 – Vue générale de RenderWeb [MCMAHON et MILENKOVIC, 2011].

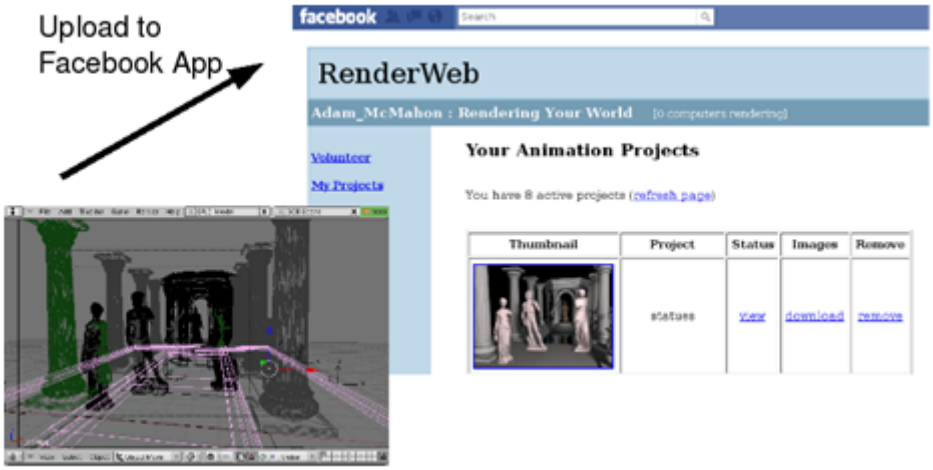

FIGURE 1.11 – Contrôle d'une application blender dans Facebook.

sommes inspirés de cette application de rendu pour bâtir notre environnement détaillé dans le troisième chapitre utilisant le même concept de calcul volontaire. L'environnement qu'on propose diffère dans le sens ou :

1. C'est un environnement générique de calcul,

2. Il ne s'appuie sur aucun middleware autre que ceux supportés nativement par les browsers,

3. Il n'utilise pas d'éléments qui peuvent constituer une menace telle que les applets,

4. L'intégration du réseau social est effectuée selon les récentes mises-à-jour des API Facebook.

1.7 Discussion

Le calcul volontaire, comme outil de calcul à faible coût, n'a cessé de susciter l'engouement des propriétaires de projets gourmands en ressources de calcul. Les plateformes présentées dans ce chapitre, hormis les plus célèbres, sont restés au stade expérimental. Les lacunes principales, à notre avis, qui freinent le développement de cette technique sont liées essentiellement à deux facteurs ; le premier est celui concernant le recrutement des volontaires et le deuxième est lié à l'aspect fortement dynamique de la disponibilité des ressources de calcul. L'amélioration continuelle des techniques utilisées au niveau des différentes fonctionnalités que doit assurer un système de calcul volontaire est impérative. Une étude des réseaux sociaux qui permettent éventuellement de recruter d'avantage de volontaires est présentée dans le chapitre suivant

Chapitre 2

Les réseaux sociaux

2.1 Introdution

Dans ce chapitre, les réseaux sociaux du point de vue théorique sont présentés brièvement dans la section 2.2. Ensuite une définition de quelques métriques servant à analyser les réseaux sociaux (extraire les caractéristiques de individus et des liens) est donnée et finalement une étude technique du réseau Facebook est relatée en mettant l'accès sur la nouvelle architecture d'intégration du réseau social dans les applications par les iFrames.

2.2 Concept du réseau social

Le concept de réseau social, issu des sciences humaines, représente une manière de penser sur les systèmes sociaux d'acteurs ou nœuds qui peuvent être des individus (personnes) ou collectivités (groupes, entreprises, cités, états, etc.) qui concentre l'attention sur les relations entre ces nœuds qui composent (construisent) le système [Everett *et al.*, 2018]. Les nœuds ont des caractéristiques appelés « attributs » qui les distinguent, ceux-ci peuvent être des traits catégoriques comme le trait d'être un masculin ou des attributs continus comme le trait d'avoir 45 ans. Les relations entre les nœuds ont également des caractéristiques. On distingue plusieurs types de réseaux sociaux selon leurs dimensions, La table 2.1 donne une liste non exhaustive des types de réseaux sociaux et exemples d'applications associées à chaque type.

La caractéristique principale des réseaux sociaux réside dans le fait que les liens partagent

TABLE 2.1 – Types de réseaux sociaux et exemples d'applications associées.

Exemples	**Applications**
Friendship networks	Établissement scolaires, organisations ou Web (Facebook, etc.)
Follower networks	Twitter, LinkedIn, ...
Preferencesimilarity networks	Instagram, Twitter, ...
Interactions networks	Messages, Email, Whatsapp, Social Network Analysis (SNA)pshat
Co-authorship networks	Dblp, Base de données scientifiques ...
User-user citations networks	Dblp, Base de données scientifiques ...
Spread networks	Epidemics, Rumeurs, ...
Co-actor networks	IMDB, etc.

des nœuds communs (ex : le lien A -¿ B partage un nœud commun avec le lien B -¿ C), ce qui crée des chaînes ou des chemins de nœuds et de liens. Une partie de la richesse du réseau est celle qui fournit un mécanisme de connexion indirecte par lequel des parties différentes du réseau peuvent être interconnectées.

On trouve plusieurs types de liens entre les nœuds d'un réseau social et chaque type donne lieu à un réseau. Ainsi, si nous mesurons les liens d'amitié, nous avons un réseau d'amitié

et si nous mesurons également les liens de parenté entre ces mêmes personnes on aura un réseau d'amitié et un réseau de parenté. Dans l'analyse, nous pouvons choisir de combiner les réseaux de différentes manières, mais réellement, nous avons deux réseaux. Certes le lien le plus communément étudié pour les personnes est le lien d'amitié, mais plus fondamentalement, la connaissance simple (qui connait qui). Le processus de familiarisation des individus (connaissance) a fait l'objet de nombreuses recherches, y compris dans l'ouvrage fondateur [Newcomb, 1961]. La figure 2.1 donne une vue d'ensemble de la complexité des relations qu'on trouve dans un réseau social.

A partir des liens individuels, d'autres relations plus sophistiquées peuvent être établies

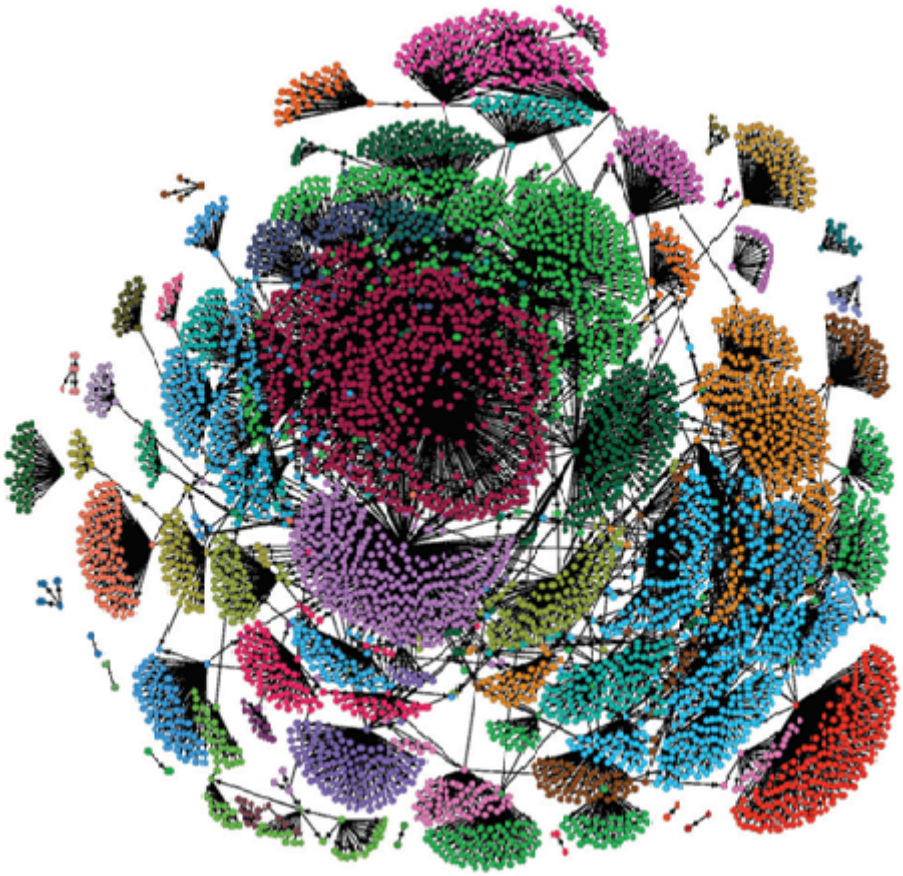

FIGURE 2.1 – Représentation des relations dans un réseau social [Tabassum *et al.*, 2018].

dans les réseaux sociaux, par exemple dans [Zhu *et al.*, 2014] on trouve une étude sur le partitionnement des réseaux sociaux en communautés. Le cloud social en ligne [Dinde et Dixit, 2015] combine les techniques issues du cloud computing avec les réseaux sociaux en ligne Online Social Network (OSN). OSN présente une allocation de ressources en se basant sur la connexion sociale de l'utilisateur. En utilisant cette approche, les utilisateurs peuvent télécharger et installer un middleware qui leurs permet d'exploiter leur réseau social personnel via une application dédiée et par conséquent leur offrir le moyen d'offrir leurs ressources à leurs amis ou d'utiliser des ressources fournies par les autres.

2.3 Analyse des réseaux sociaux

L'analyse des réseaux sociaux Social Network Analysis (SNA) est une discipline qui s'intéresse à l'aspect relationnel et comportemental des individus dans les réseaux sociaux. Les méthodes et techniques du SNA ont été conçues pour découvrir des modèles d'interaction entre les acteurs sociaux [Tabassum *et al.*, 2018] et par conséquent, le SNA se concentre sur les relations établies entre les entités sociales plutôt que les entités sociales elles-mêmes. En fait, l'objectif principal de cette technique est d'examiner à la fois le contenu et les modèles de relations dans les réseaux sociaux afin de comprendre les relations entre les acteurs et les implications de ces relations. Il est à noter que les outils développés aux fin d'analyse des réseaux sociaux sont également utilisés pour les autres réseaux de données (réseaux de capteurs, etc.).
Les tâches communes dans la SNA peuvent être résumées en :

1. Identification des acteurs les plus influents, prestigieux ou centraux, à l'aide de mesures statistiques ;
2. Identification des collecteurs et des autorités, à l'aide d'algorithmes d'analyse de liens ;
3. Découverte de communautés en utilisant des algorithmes de détection des communautés ;
4. Propagation de l'information dans le réseau social, à l'aide d'algorithmes de diffusion.

Ces tâches et les techniques utilisées afin de les accomplir sont extrêmement utiles dans le processus d'extraction des connaissances des réseaux et, par conséquent, dans le processus de résolution des problèmes par la SNA. En raison de la nature attrayante de ces tâches et du potentiel élevé ouvert par ce type d'analyses, la SNA est devenue une approche dans une myriade de domaines, allant de la biologie aux affaires. Par exemple, certaines entreprises utilisent la SNA afin de maximiser le « bouche à oreille » positive de leurs produits en ciblant les clients ayant une plus grande valeur sur le réseau social.
Une multitude de mesures peuvent être extraites d'un réseau social, dans un premier lieu un réseau peut être représenté comme un graphe classique : $G =< V, E >$: où V (Vertex) est l'ensemble des nœuds du réseau et E (Edges) est l'ensemble des liens entre les nœuds. Les structures de données informatiques utilisées pour représenter un réseau social sont les listes et les matrices d'adjacence ; les listes offrent un aspect compact de stockage (pas de perte d'espace) contre une manipulation relativement complexe. quant aux matrices elles offrent un moyen efficace de manipulation et d'inférence des différentes mesures contre

une perte d'espace du à leur aspect creux.
Différentes mesures au niveau des nœuds peuvent être relevées :

— **Le degré (ou la valence) :**est calculé par l'équation (2.1) où le terme a_{ij} désigne l'entrée (i,j) dans la matrice d'adjacence associée au réseau.

$$k_i = \sum_{j=1}^{n} a_{ij} \tag{2.1}$$

Malgré sa simplicité, cette mesure est très importante vu qu'elle reflète l'influence d'un nœud dans le réseau social. Le degré d'un nœud peut être décortiqué en deux valeurs : degré entrant (noté k_i-) exprimant le nombre de liens incidents de l'extérieur vers un nœud (exemple personnes qui connaissent un individu) et le degré sortant (noté k_i+) exprimant le nombre de liens incidents vers l'extérieur du nœud (exemple : les personnes que cette individu connaisse).

— **Interdépendance :** mesure l'aptitude d'un nœud à se situer entre les autres nœuds du réseau et peut être calculé comme le pourcentage de chemins les plus courts qui traversent le nœud. Cette mesure peut être obtenue par l'équation (2.2).

$$\beta_\gamma = \sum_{u,v \in V(G)}^{n} \frac{\sigma_{uv}(\gamma)}{\sigma_{uv}} \tag{2.2}$$

Où σ_{uv} représente les chemins les plus courts entre les deux nœuds u, v. $\sigma_{uv}(\gamma)$ représente les chemins les plus courts passants par γ.

— **Proximité :** c'est une mesure approximative de la position globale d'un nœud dans le réseau. Formellement, c'est la longueur moyenne de tous les chemins les plus courts d'un nœud à tous les autres nœuds du réseau. Dans le contexte des réseaux sociaux, la proximité est une mesure d'accessibilité qui mesure la rapidité par laquelle un nœud donné peut atteindre tout le monde dans le réseau. La proximité peut être obtenue approximativement par l'équation (2.3).

$$Cl_v = \frac{n-1}{\sum_{u \in G(V) \setminus v} d(u,v)} \tag{2.3}$$

Où n est le nombre de nœuds du réseau, v le nœud considéré pour lequel on calcul la proximité, u est n'importe quel autre nœud différent de v, $d(u,v)$ est la distance du chemin le plus court entre u et v.

— **Centralité :** Cette métrique est basée sur l'attribution d'un score relatif à chaque nœud mesuré par l'aptitude d'un nœud donné à se connecter à d'autres nœuds « bien-connectés ». Parmi les mesures de centralité utilisées, celles basées sur le premier vecteur propre de la matrice d'adjacence. La centralité d'un nœud, dans

ce cas, est proportionnelle à la somme des centralités de ses voisins et donnée par l'équation (2.4).

$$Cent_i = \frac{1}{\lambda} \sum_{j=1}^{n} a_{ij} x_i \tag{2.4}$$

Où xi/xj désigne la centralité des nœuds i/j, a_{ij} représente une entrée dans la matrice d'adjacence et λ désigne la plus grande valeur propre de A.
Il est à noter que les métriques précédentes sont non exhaustives et se rapportent à un nœud particulier d'autres métriques concernant le réseau en sa totalité sont à analyser. Ces métriques reposent sur trois concepts fondamentaux à savoir : chemin (séquence de nœuds successif entre deux nœuds donnés), distance géodésique (longueur du chemin le plus court entre deux nœuds donnés) et l'excentricité d'un sommet (la plus grande distance géodésique entre un sommet donné et tous les autres sommets du graphe).

— **Diamètre et rayon :** Le diamètre d'un réseau ou d'un groupe de sommets et la valeur maximale de l'excentricité des sommets. L'excentricité d'un nœud v est donnée par l'équation (2.5).

$$e(v) = max(d_{i \in V(G) \setminus v}(i, v)) \tag{2.5}$$

Le diamètre et le rayon peuvent s'exprimer par les équations (2.6) et (2.7) respectivement :

$$D = max(e(v)/v \in V(G) \tag{2.6}$$

$$R = min(e(v)/v \in V(G) \tag{2.7}$$

— **Moyenne des distances géodésiques :** Cette métrique peut être utilisée pour mesurer l'efficacité du flux d'information au sein du réseau. Elle est calculée par la formule illustrée dans l'équation (2.8)

$$l = \frac{1}{\frac{1}{2}n(n-1)} \sum_{i \geq j} d(i, j) \tag{2.8}$$

— **Réciprocité :** La réciprocité r est une quantité spécifique pour les réseaux dirigés qui mesure la tendance des paires de nœuds à former des connexions mutuelles entre eux. Il existe plusieurs façons de calculer cette métrique. Le moyen le plus populaire et intuitif est de calculer le rapport entre le nombre de connexions mutuelles dans le réseau et le nombre de toutes les connexions, comme indiqué dans l'équation (2.9)

$$r = \frac{\sum a_{ij} \setminus a_{ji} \in E(G)}{\sum a_{ij} \setminus a_{ji} \in E(G) + \sum a_{ij} \setminus a_{ji} \notin E(G)}, 0 < r < 1 \tag{2.9}$$

— **Densité :** La densité ρ est une mesure importante au niveau du réseau, qui

est capable d'expliquer le niveau général de connectivité dans un réseau. Elle est donnée par la proportion de liens dans le réseau par rapport au nombre maximal possible de liens, tel que défini dans l'équation (2.10)

$$\rho = \frac{\sum a_{ij}}{\frac{1}{2}n(n-1)} \tag{2.10}$$

Plusieurs autres mesures peuvent être envisagées afin d'étudier des aspects spécifiques des réseaux sociaux.

2.4 Le réseau social Facebook

Facebook fut certainement le plus grand réseau social actuellement (avec 2.2 Milliards d'inscrits). En outre de sa popularité comme réseau social offrant de multitude de fonctionnalités de partage de contenu, de publication, de création de groupes, etc. Il offre également aux développeurs la possibilité d'intégration de son aspect social à leurs applications qui vont profiter de la valeur ajoutée par l'aspect viral du réseau social (ex partager une application de sondage en utilisant des liens tels que l'amitié).

2.4.1 Intégration

Techniquement, on distingue plusieurs types d'applications qui peuvent intégrer les API de Facebook selon plateforme de développement (iOS, Android, les ex-Canevas FaceBook Modeling Language (FBML),Web, etc.). Les canevas FBML (Facebook Markup Language) utilisent exclusivement les API offertes par Facebook ; cette solution présente l'avantage de disposer de toute la force et la richesse du Facebook (widgets, canevas, etc.) son inconvénient principal réside dans les limitations des canevas à l'HTML pur comme l'absence des extensions Web telles que javascript. Officiellement, les canevas FBML, dont le principe est illustré par la figure 2.3 , sont dépréciés au profit des Web iFrame.
Pour les applications Web ou iFrame, le site de l'application intègre directement des API Facebook tout en gardant la richesse de programmation Web classique comme l'utilisation du javascript. Dans une iFrame on ne peut pas interpréter le FBML, mais il est possible d'utiliser du XFBML avec les framework Javascript (jQuery, Ajax, etc.). La figure 2.3 représente la nouvelle interaction avec la plateforme facebook selon le schéma des iFrames.
La nouvelle solution iFrame préconisée par la plateforme permet aux deux parties de l'application (serveur et client) d'échanger directement des données via les protocoles classiques tels http et websocket, ce qui présente d'une part une meilleure performance pour les applications en plus de l'utilisation des possibilités des frameworksjavascript, d'autre part, elle permet de soulager la plateforme Facebook, encombrée déjà par les données des deux (02) milliards d'abonnées, des interactions des applications hors volet social.

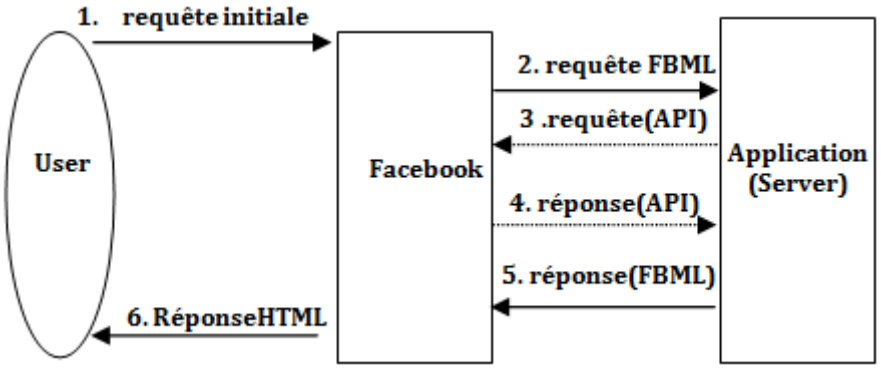

FIGURE 2.2 – Interaction FBML Facebook (Déprécié).

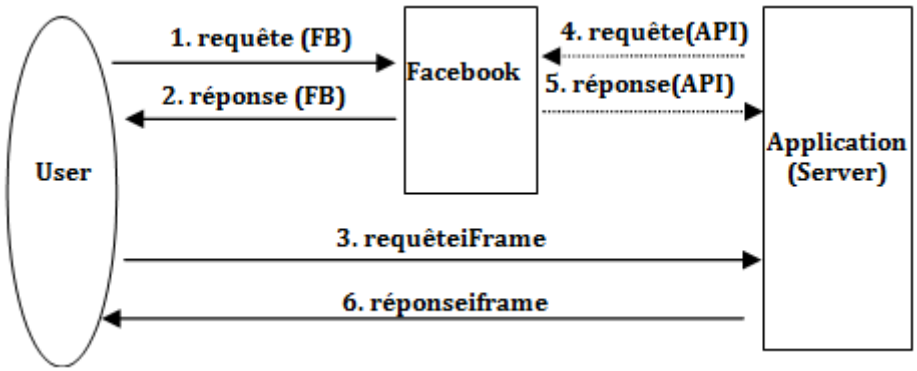

FIGURE 2.3 – interaction iFrame Facebook/Application/User.

Néanmoins, la communication classique directe entre le client et le serveur en plus des API Facebook qui récupèrent des informations personnelles et parfois sensibles, posent un grand problème de confidentialité ; Comment protéger la confidentialité des utilisateurs d'une application sans la priver de l'aspect social offert par le réseau ? Depuis le scandale de Cambridge Analytica Data [1] où des informations confidentielles d'utilisateurs de Facebook, notamment leurs flux d'actualités, leurs calendriers et leurs messages ainsi que leurs emplacements ont été utilisées à des fins politiques à leur insu, Les responsables de Facebook ont porté des modifications de taille aux API avec lesquelles une application accède aux informations de leurs utilisateurs. Une application Facebook iFrame doit, à priori, s'inscrire sur le site Facebook developper [2] . Plusieurs informations doivent être fournies comme l'url de l'iFrame (adresse du serveur) ainsi que le domaine, la politique de confidentialité et d'autres informations concernant son auteur. Facebook offre plusieurs produits à ajouter, le produit essentiel à toute application est FBLogin qui permet aux utilisateurs de se connecter à l'application avec leur identifiant FB et permettant à l'application de les identifier.

1. https ://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal
2. https ://developers.facebook.com/docs

Le contenu des pages des utilisateurs peut être récupéré via un autre produit : le GraphApi, une page facebook est organisée comme un graphe avec des nœuds correspondants aux différentes zones alimentées par l'utilisateur, ses amis ou des suggestions Facebook tel que le mur, le fil d'actualité, la liste d'amis, etc. Le GraphApi récupère les valeurs de ces nœuds via des requêtes spécifiant l'identificateur du nœud et un token d'accès obtenu à partir de FBLogin, ce token correspond à une autorisation de l'utilisateur à l'application pour consulter ce contenu. De plus, l'application inscrite sur Facebook doit avoir une autorisation explicite pour consulter un type de nœuds de GraphApi. La figure 2.4 illustre le tableau de bord permettant de configurer les applications Facebook.

Par exemple, pour qu'une application accède à la liste d'amis (nœud user_friends) d'un

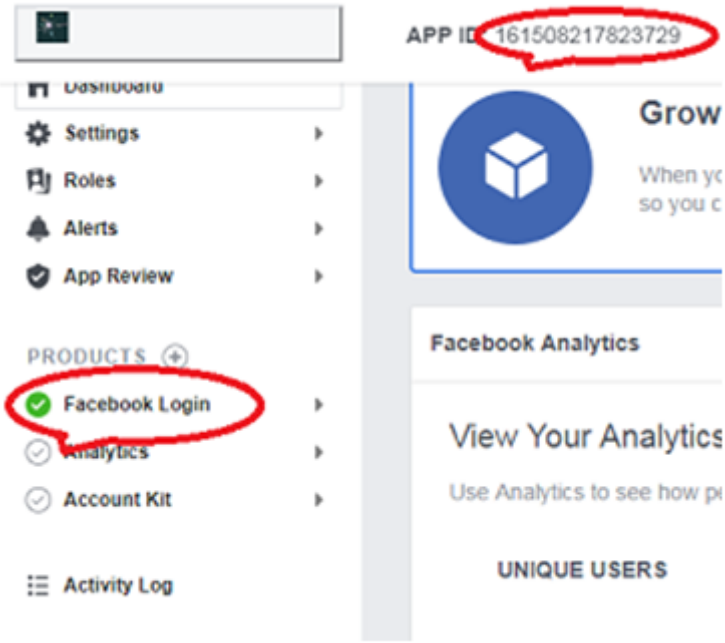

FIGURE 2.4 – Tableau de bord d'une application Facebook.

utilisateur, l'application doit avoir une permission explicite d'accès aux nœuds de type « user_friends » par un paramétrage au niveau de la plateforme Facebook qui impose des règles et des conditions pour avoir cette permission. De plus, chaque utilisateur est sollicité individuellement pour autoriser l'application à accéder au contenu de son « user_friends ». Cette procédure vise à renforcer la confidentialité dans le réseau social.

2.5 Conclusion

Dans cette partie nous avons présenté une discipline en plein expansion ces dernières années ; les réseaux sociaux informatiques. Ce type de réseaux peut être abordé selon plusieurs angles de vue. Par exemple l'analyse permet de porter des réponses sur

des caractéristiques comportementales de communautés particulières. Nous nous sommes intéressés aux réseaux sociaux comme un ensemble de ressources connectées par des relations ayant un certain degré de confiance. Nous nous somme penché sur le réseau Facebook ; le plus grand réseau social actuellement. Dans le chapitre suivant, l'environnement de calcul volontaire social qu'on a proposé est présenté.

Chapitre 3

SVCE : Environnement de calcul volontaire social

3.1 Introdution

Dans ce chapitre, notre environnement de calcul volontaire social qu'on a nommé SCVE est présenté, son architecture et fonctionnement sont détaillés. On s'est penché également sur la problématique de l'ordonnancement des tâches dans les systèmes de calcul volontaire et on a contribué par la proposition d'une nouvelle politique d'ordonnancement qu'on a baptisée Task-Adapted Scheduling Policy (TASP). On a expérimenté notre environnement par une application pilote sur la conjecture de Collatz. La comparaison en termes de recrutement des volontaires ou l'efficacité de la politique d'ordonnancement a montré la validité de nos propositions.

3.2 Modèle du calcul

Nous reprenons le modèle du calcul volontaire décrit dans la plupart des travaux y afférents cités dans le Chapitre 1. Dans ce modèle, un projet de calcul volontaire consiste en un ensemble de petites tâches élémentaires suffisamment grand pour justifier l'utilisation des ressources des participants. Chaque tâche peut être vue comme un code exécutable opérant sur des données en entrée et produisant un résultat(équations (3.1) et (3.2)).

$$project = task_i/i = 1..n \tag{3.1}$$

$$task_i = f(p_i1, p_i2, \ldots p_ik) = result_i \tag{3.2}$$

Dans la plupart des projets de calcul volontaire, les tâches partagent un code unique, et opèrent sur de grandes intervalles de données (paramètres). C'est d'ailleurs le type d'application ou le calcul volontaire est parfaitement adapté (parallélisme de données) et a atteint des performances semblables aux mainframes (Seti@home, PrimeGrid, GIMPS, etc.). Dans ce type d'application, l'ordonnanceur comporte entre autre, un générateur de données (paramètres) à dispatcher sur les volontaires au fur et à mesure. Il peut s'agir aussi de tâches différentes et indépendantes, dans ce cas un générateur de tâches est implémenté coté serveur. Deux techniques sont utilisées pour la génération des tâches [Nouman Durrani et Shamsi, 2014] ; l'une consiste à générer en une seule passe toutes les tâches du projet, l'autre consiste à générer à la demande un pool de tâches (S_i). Dans notre environnement, on a utilisé la deuxième car elle présente l'avantage d'alléger le serveur (stockage d'un petit nombre de tâches, accélération des requêtes de recherche, d'ordonnancement et de validation des résultats, etc.). La figure 3.1 schématise le modèle de calcul décrit précédemment.

Pour guider les algorithmes d'ordonnancement, certaines informations sont associées aux tâches comme l'estimation du temps d'exécution d'une tâche w_{ti}, l'estimation du temps

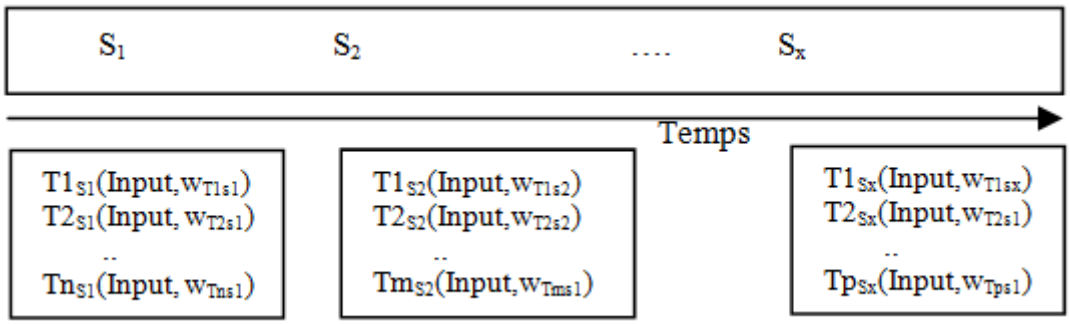

FIGURE 3.1 – Modèle de calcul utilsé dans le CV.

d'exécution d'une tâche sur une ressource r_j qu'on note (w_{ti}, r_j) et le temps max accordée à une tâche $maxt_i$, au dela de ce temps, le volontaire exécutant cette tâche est considéré comme défaillant et la tâche est réattribuée à un autre volontaire.

3.3 Architecture

L'environnement que nous proposons [Kadache et Seghir, 2021] est une application Client/Serveur basée sur le Web, les participants doivent être connectés en permanence au serveur. Le mode connecté, malgré son apparence contraignante, offre une meilleure gestion des participants, une tâche non exécutée à cause d'une déconnexion ou autre défaillance est détectée et réattribuée à temps. Le choix de la taille de la tâche est primordial dans ce mode « connecté » des participants, des tâches trop longues peuvent ne pas être exécutées à cause d'une déconnexion, des tâches trop petites dégradent considérablement le rendement du système par augmentation du coût de communication au niveau de la machine du participant et par une lourde gestion au niveau du serveur (ordonnancement, stockage du résultat, etc.). La figure 3.2 illustre l'architecture de SCVE et ses composants.

3.3.1 Coté serveur

Le serveur est scindé en deux composants : le premier « HttpServer » utilise le protocole http classique afin de fournir les différents composants du client nécessaires à l'exécution des tâches (le frontend et le code des tâches ainsi que les bibliothèques liées). Le deuxième composant utilise le protocole webSocket [IETF, 2018] afin d'échanger les données et les résultats des différentes tâches avec le frontend du client. Le schéma d'échange est décrit dans la section 3.3.2. Le composant comporte aussi les fonctionnalités essentielles à tout système de calcul volontaire à savoir : l'ordonnancement, la validation des résultats et la purge de la base de données. La structure des données utilisée pour le

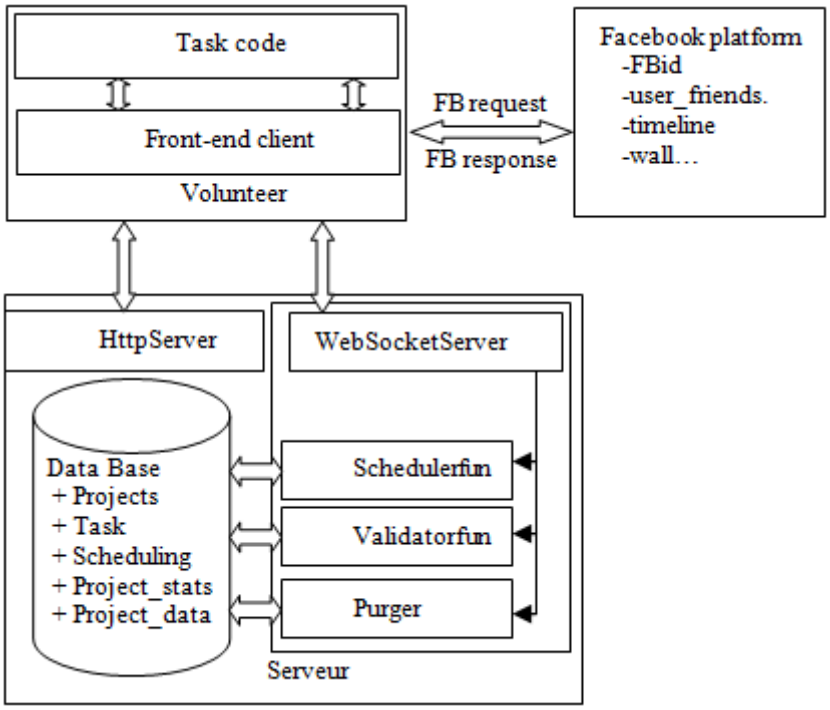

FIGURE 3.2 – Architecture de SCVE.

stockage des résultats envoyés par les volontaires est donnée par les tables 3.1 et 3.2.

L'ordonnanceur utilisé dans SCVE est chargé de distribuer les tâches sur les différents

TABLE 3.1 – Table Project.

Project_id	Task_id	turnaround	worker

TABLE 3.2 – Table Result.

Project_id	Task_id	InputData	result

volontaires disponibles à un instant donné. Plusieurs techniques sont envisageables, l'ordonnancement est discuté dans la section 3.4 où une nouvelle politique d'ordonnancement est proposée et évaluée.

3.3.2 Coté client

Le client de notre environnement consiste en une page Web principale qui est affichée en premier lieu à la demande de l'url du serveur, ensuite le composant frontend est chargé. Ce composant constitue le réceptacle d'exécution des tâches et de communication avec le serveur, le frontend assure également la fonction de connexion au réseau social et le partage de l'application via les API de ce dernier. La figure 3.3 montre l'interaction entre les clients SCVE (Volontaires) et le serveur SCVE ainsi que Facebook.

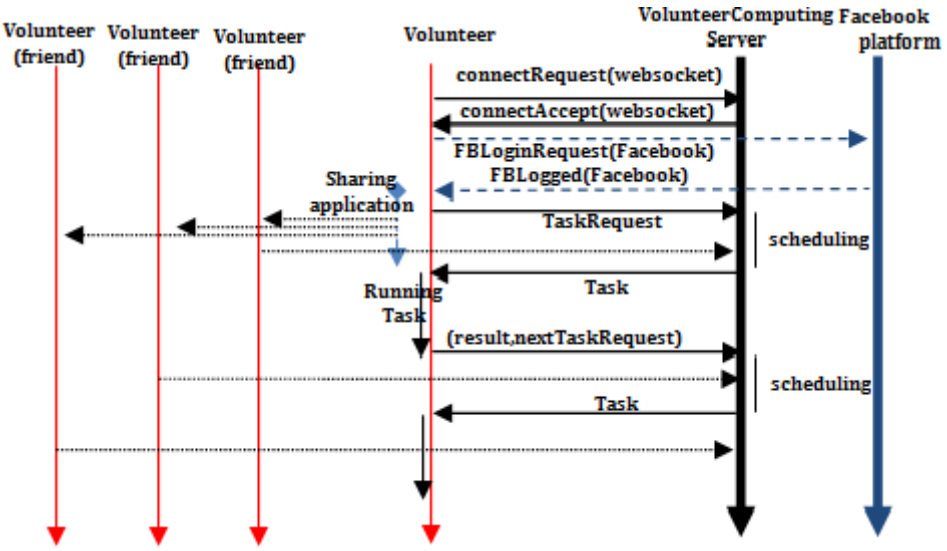

FIGURE 3.3 – Communication Volontaire/Serveur CV/Facebook.

3.3.3 Détails d'implémentation

Nous avons utilisé le framework nodejs [1] qui est un serveur d'application monothreadé contrairement à la plupart des autres serveurs (PHP, Apache, etc.), sa puissance réside dans le fait qu'il est orienté événement, ce qui permet des temps de réponse très court pour les requêtes des clients. Nodejs permet d'exécuter du javascript coté serveur. Il est consolidé par un très grand nombre de plugins développés également en javascript et pouvant être incorporés à la demande (par téléchargement manuelle ou grâce au gestionnaire de paquet npm). JQuery, express, fs, websocket sont des exemples de plugins les plus utilisés permettant d'écrire aisément des applications rapides et très puissantes. Pour stocker les données nécessaires à notre environnement (projets, tâches, données et résultats), nous avons utilisé le fameux serveur de base de données MySql d'Oracle® avec le plugin mysql.js permettant à l'application nodejs de communiquer avec le serveur Mysql. Notre environnement expérimental est hébergé sur un Virtual Private Server (VPS) (2 cœurs, 4GO RAM, 250GO DD, connexion illimitée), bien entendu, pour mettre en place un système de CV pouvant traiter des centaines de milliers de volontaires, une infrastructure beaucoup plus importante devra être mise en place. La mesure du temps d'exécution des tâches au niveau des machines des bénévoles est échantillonnée par la fonction window.performance() dont la précision théorique est de l'ordre de 5 microsecondes, cependant, elle dépend fortement du type de browser et de la machine adjacente.

1. https ://nodejs.org/en/docs/

3.4 Ordonnancement dans SCVE

Dans SCVE, nous avons proposé une nouvelle politique d'ordonnancement qui s'appuie sur les capacités instantanées de calcule des volontaires. Avant de détailler son fonctionnement, une brève revue des politiques d'ordonnancement utilisées dans les systèmes de CV est présenté afin de mettre en évidence l'efficacité de notre approche.

3.4.1 Politiques d'ordonnancement existantes

L'ordonnancement est la fonction centrale dans un système de CV qui a fait l'objet de plusieurs travaux ; les politiques d'ordonnancement existantes peuvent être catégorisées en deux classes [Nouman Durrani et Shamsi, 2014] ; les politiques naïves (aveugles) qui attribuent les tâches aux participants sans accorder d'importance à l'historique des activités des participants. Dans cette classe on peut citer :

- **FCFS : premier arrivé premier servi,**
- **Ordonnancement local :** utilisé comme alternative dans BOINC et qui consiste à attribuer les tâches au volontaire qui possède les données nécessaires à leur exécution,
- **Ordonnancement aléatoire : attribue aléatoirement des tâches aux volontaires.**

Les politiques d'ordonnancement naïves mènent à des répartitions de tâches sur des participants avec une faible capacité de calcul et de moindre fiabilité.
La deuxième classe, celle des politiques informées, basée sur l'historique d'exécution des volontaires. Dans Ibm World community grid [Nouman Durrani et Shamsi, 2014], [Toth et Finkel, 2009] qui utilise la plateforme BOINC, les tâches sont réparties en fonction du temps moyen dont un client a besoin pour renvoyer un résultat avant la date limite.
Une autre technique consiste à ce que les workers anticipent le téléchargement d'un nombre suffisant de tâches de plusieurs projets recouvrant l'intervalle de connexion avec le serveur. Chaque projet aura toujours au moins une tâche en file d'attente ou en cours d'exécution. Cette politique fonctionne bien dans certains cas, mais il y a d'autres cas dans lesquels elle échoue terriblement [Anderson et McLeod, 2007], [Kondo *et al.*, 2007].
D'autres algorithmes d'ordonnancement utilisent deux seuils [Estrada *et al.*, 2009] : le premier lié à la disponibilité des workers et le second à leur fiabilité. Les tâches ne sont pas attribuées à un Worker avec des valeurs de fiabilité/disponibilité inférieures aux seuils utilisés qui peuvent être fixes ou variables. Les workers avec des valeurs constamment faibles de disponibilité et de fiabilité sont considérés inaptes pour faire des calculs et sont supprimés (ce qui est considéré comme inconvénient de ce type de stratégies). Dans les stratégies informées, on trouve aussi, les politiques à base de tourniquet avec priorité [Ngo *et al.*, 2008] où les workers disposant de ressources similaires sont regroupés dans

des niveaux de priorité. Une approche proactive de réattribution des tâches est proposée dans [Lee *et al.*, 2010], où les tâches attribuées à des volontaires peuvent être réattribuées à de nouveaux bénévoles offrant des capacités supérieures. Dans cette approche, une certaine dégradation de performance due à la réattribution est tolérée. D'autres travaux tentent de décentraliser certaines fonctions y compris le stockage et l'ordonnancement sur des nœuds de calculs [Li et Franzinelli, 2016]. Une simulation des cinq différentes techniques d'ordonnancement à savoir Buffer none, Download early, Buffer one task, Buffer multiple Task et super-optimal sur des machines respectivement avec un seul processeur, multicoeurs et des machines en veille [Toth et Finkel, 2009].

3.4.2 L'algorithme d'ordonnancement TASP

Les algorithmes d'ordonnancement évoqués dans la section précédente tentent de trouver le meilleur mapping entre un ensemble de tâches pondérées existantes (déjà générées) avec les capacités des ressources disponibles à un instant donné. L'idée de l'algorithme que nous proposons est de générer des tâches avec des pondérations adéquates reflétant au mieux les capacités des ressources de calcul disponibles. Dans ce cas, le modèle du calcul volontaire peut être assimilé à la théorie de charge divisible Divisible Load Theory (DLT) décrite dans [Bharadwaj *et al.*, 2003].
Dans ce contexte, le travail présenté dans [Beaumont *et al.*, 2005] applique deux modèles sur les calculs et les communications à un réseau à topologie virtuelle en étoile : le modèle linéaire appelé « STARLINEAR » et le modèle affine « STARAFFINE » où un facteur de latence est ajouté au modèle linéaire. Le nœud maître peut distribuer des « morceaux » de calcul en une round ou plusieurs dans le cas d'un grand projet de calcul, ce qui permet de mettre les calculs en pipeline en plus d'un recouvrement communication/calcul.
Dans le contexte du calcul volontaire, générer une tâche avec une complexité préalable peut apparaitre difficile à accomplir. Afin de surmonter cette lacune, les projets doivent être divisés en un nombre très grand de tâches très minuscules (élémentaires) de telle manière à ce que tout volontaire, et en faisant abstraction de ses capacités de calcul, peut exécuter un nombre significatif de ces petites tâches élémentaires. Ceci est possible avec la majorité des projets de calcul volontaire impliquant une quantité colossale de données à traiter.
Une autre supposition à tenir en compte, est que les conditions d'exécution au niveau des volontaires demeurent stables pour une certaine durée t_{stab}. En dehors de cette supposition, les capacités des ressources de calcul fluctuent aléatoirement et aucune politique d'ordonnancement ne peut être justifiée.
TASP (Task-Adpated Scheduling Policy) est une adaptation du modèle STARLINEAR multi-rounds aux environnements de calcul volontaire. Le modèle affine peut être appliqué à son tour pour les calculs et pour les communications. Sans perte de généralité, on ne

va considéré que le modèle STARLINEAR pour les calculs. Il a été démontré que toute politique d'ordonnancement optimale pour ce modèle doit satisfaire les deux conditions suivantes :

— **Condition 1 :** Tous les workers participent au calcul.
— **Condition 2 :** Tous les workers terminent les calculs simultanément.

L'application du modèle STARLINEAR à l'environnement CV doit tenir compte deux contraintes :

1. La disponibilité des ressources de calcul est dynamique et varie continuellement.
2. Les capacités de calcul des volontaires varient aussi continuellement du fait qu'elles ne sont pas dédiées exclusivement aux calculs.

Soit $R = \{r_1, r_2, \ldots, r_i, \ldots, r_p\}$ l'ensemble des ressources disponibles à un moment observable i.e. où un cycle d'ordonnancement est activé au niveau du serveur de calcul volontaire $R_0, T = \{t_1, t_2, ...t_i, ...t_q\}$ un ensemble très grand de tâches élémentaires équivalentes en complexité. Sans perte de généralité, on suppose que ces tâches sont suffisamment petites de telle manière que n'importe quelle ressource peut en exécuter un nombre significatif de ces dernières. Considérons les trois ensembles :
$W = \{w_1, w_2, ...w_i, ...w_p\}$ où w_i est le temps d'exécution d'une tâche t_i sur la ressource $r_i, Q = \{q_1, q_2, ...q_i, ...q_p\}$ où q_i est le temps nécessaire d'envoi d'une tâche élémentaire et ses données du serveur R_0 vers la ressource r_i et $S = \{s_1, s_2, ...s_i, ...s_p\}$ où s_i est le temps nécessaire pour récupérer le résultat de l'exécution de la tâche t_i sur la ressource rivers serveur R_0.
Selon le modèle linéaire, le temps nécessaire à l'exécution de α_i tâches élémentaires est donné par l'équation (3.3).

$$t_i = \alpha_i(q_i + w_i + s_i) \tag{3.3}$$

Soit $Coeff = \{\alpha_1, \alpha_2, ...\alpha_i, ...\alpha_p\}$ les coefficients correspondants à l'ensemble des tâches exécutées par toutes les ressources respectivement. Le temps total d'exécution correspondant à cet ensemble est donné par l'équation (3.4).

$$T_{total} = \sum_{i=1}^{p} \alpha_i(q_i + w_i + s_i) \tag{3.4}$$

Le problème d'ordonnancement revient à déterminer les coefficients α_i qui minimisent T_{total}, et calculer $q_i, w_i et s_i$ qui varient fortement dans un environnement de calcul volontaire.
Pour simplifier, on ne va prendre en considération que les grandeurs du calcul w_i, le même raisonnement peut être étalé sur les grandeurs de communications.

Estimation des facteurs w_i

Les facteurs w_i qui correspondent aux capacités de calcul des volontaires sont calculés dans le round D_{k-1} par la moyenne des temps d'exécution des tâches accomplies pour chaque ressource.
Soit :
$w_i(D_k)$: le temps d'exécution d'une tâche élémentaire sur la ressource r_i à calculer à partir du round D_k,
$\alpha_i(D_{k-1})$: le nombre de tâches élémentaires accomplies par la même ressource r_i pour le round D_{k-1}
et $tr_i(D_{k-1})$: le temps d'exécution de $\alpha_i(D_{k-1})$ tâches élémentaires par ladite ressource r_i. La performance $w_i(D_k)$ est calculée par l'équation (3.5).

$$w_i(D_k) = \frac{tr_i(D_{k-1})}{\alpha_i(D_{k-1})} \tag{3.5}$$

A chaque arrivée d'une nouvelle ressource (connexion par un nouveau volontaire), un petit ensemble de tâches élémentaires est affecté à cette dernière. Après la réception des résultats, la ressource est insérée dans l'ensemble R et la valeur de w est déterminée et sera ensuite recalculée à chaque round par l'équation (3.5).

Classification des ressources

Avec un grand nombre de ressources en ligne dans notre environnement SCV, il est intéressant de regrouper les ressources équivalentes en capacité de calcul/communication dans des classes de ressources et les gérer de la même manière pour deux raisons :

1. Alléger le processus d'ordonnancement par l'utilisation de la politique efficace FCFS pour les ressources de la même classe,
2. Equilibrage de charge entre les volontaires de la même classe ; en effet les volontaires les plus relativement puissants en capacité de calcul génèrent un temps additionnel et peuvent exécuter les tâches attribuées à des volontaires défaillants ou ceux avec des performances dégradés.

Un petit seuil est choisi judicieusement afin de répartir l'ensemble $W(D_{k-1})$ en sous-ensembles de ressources ayant des capacités de calcul similaires. Le partitionnement des ressources est effectué selon l'algorithme 1.

Algorithm 1 W_Part

Require: $R = \{r_1, r_2, .., r_p\}$ a set of resources at the round D_{k-1}
$W = \{w_1, w_2, ..w_p\}/w_i \neq 0$ set of task's weight returned at the round D_{k-1}
tr : a threshold
Ensure: 1- A partition on R :
$PR = \{Pr_1, Pr_2, ...Pr_m\}$
2- A partition's weights $PRW = \{prw_1, prw_2, ..., prw_m\}$
while R is not empty **do**
take the first element r_{first} from R
$R = R - \{r_{first}\}$ // remove w_{first} from R
$Pr = \{r_{first}\}$ // initialize the Pr with r_{first}
$prw = w_{first}$
for each r_z in R **do**
if $|w_z - w_{first}| <$ tr **then**
$Pr = Pr + \{r_z\}$
$R = R - \{r_z\}$
$prw = prw + w_z$
end if
end for
$PRW = PRW + \{prw/card(Pr)\}$ // put the average of the new partition's weight into the PRW set, card(Pr) is the cardinality // of Pr
$PR = PR + Pr$ // put the partition Pr in PR
end while

Calcul des coefficients α_i

Les facteurs $\alpha_i(D_k)$ peuvent être maintenant calculés pour toutes les partitions Pr_i appartenant à PR par l'équation (3.6).

$$\alpha_i(D_k) = \frac{t_{stab} * |Pr_i|}{prw_i(D_{k-1})} \tag{3.6}$$

Où $|Pri|$ est la cardinalité de Pr_i. L'équation (3.6) signifie qu'à chaque partition Pr_i correspond le nombre de $|Pri|$ de ressources identiques avec une capacité de calcul total de $\alpha_i * prw_i$. Cette équation garantie théoriquement la seconde condition d'optimalité d'une politique d'ordonnancement, en effet tous les volontaires terminent, en théorie, leurs tâches simultanément dans la fenêtre temporelle de t_{stab}.
L'algorithme 2 récapitule la politique d'ordonnancement proposée dans SVCE.

Algorithm 2 TASP

Require: $R = r_1, r_2, .., r_n$ a set of available resources
$N = n_1, n_2, ..n_k / n_i \neq 0$ a set of the number of tasks successfully completed by each r_i
$T = t_1, t_2, .., tk$ a set of total turnaround returned by resource r_i running n_i tasks at the last round
t_{stab} an interval of time in witch execution conditions remain stable
Ensure: Task assignation
updating R, T, N
for (each new resource r_k in R) **do**
send t_{test} to r_k
$R = R - \{r_k\}$
end for
$W = \{\}$
for each t_i in T **do**
let $w_i = t_i / n_i$ the average of turnaround time of a single task for r_i
put w_i in W
end for
call W_PART($R, W, threshold$) the result is a partition PR with a corresponding set of weights PRW
for each s in PR **do**
let prw_s the corresponding weight of s in PRW
generate $(card(s) * t_{stab}/prw_s)$ elementary task for s, card(s) is the cardinality of s
end for
send tasks to their respective partition of resources using FCFS policy.

L'algorithme TASP proposé peut être récapitulé par les étapes suivantes :

— Un projet est divisé "ou émietté" en un nombre très grand de tâches élémentaires équivalentes en terme de temps d'exécution.
— Pour chaque nouvelle arrivée d'une ressource, un petit nombre de tâches élémentaires est envoyé, le temps d'exécution moyen d'une tâche élémentaire est calculé.
— On choisi une intervalle de temps (temps de stabilité).
— Pour chaque ressource disponible :

1. on calcul le nombre N de tâches élémentaires qu'elle peut exécuter durant le temps de stabilité,
2. les N tâches sont acheminées vers la ressource,
3. au retour des résultats, sa capacité est mise à jour pour le round suivant.

Les avantages de notre algorithme peuvent être résumés en :

— Éviter la résolution du problème d'attribution de tâches existantes avec des poids variables à des ressources avec des capacités aussi variables ce qui se présente comme un problème NP-complet et par conséquence accélérer la procédure d'ordonnancement.

— La partition du projet en petites tâches garantie de servir tous les bénévoles disponibles ce qui évite le problème de famine et contribue à la satisfaction de la première condition d'optimalité de toute politique d'ordonnancement.
— La génération des tâches selon les capacités des volontaires évite de stocker des tâches inutilement au niveau du serveur, ce qui allège considérablement son fonctionnement.

En revanche, l'inconvénient réside dans le fait de générer un grand nombre de tâches, ce qui implique un surcout concernant leur gestion (suivi de leur états, stockage des résultats, etc. Cet inconvénient peut être contourné en utilisant plusieurs serveurs pour équilibrer la charge liée.

3.5 Expérimentation

3.5.1 La conjecture de Collatz

Nous avons choisi comme projet pilote de test de notre environnement la conjecture de Collatz pour deux raisons ; la première est qu'il requiert un calcul pratiquement à l'infini, la deuxième est la simplicité de sa mise en œuvre. La suite de collatz est définie par l'équation (3.7).

$$u_{n+1} = \begin{cases} u_n/2 & \text{if } u_n \text{ is even} \\ 3 * u_n + 1 & \text{otherwise} \end{cases} \tag{3.7}$$

On remarque que pour $u_i = 4$, la suite de Collatz atteint un cycle de (2,1,4). La conjecture affirme que pour tout N, il existe un indice i dans la suite tel que $u_i = 1$. Actuellement, il n'y a aucune preuve à cette affirmation. Plusieurs approches on été utilisées pour tenter de statuer sur cette conjecture. L'approche probabiliste par exemple tente de déterminer le facteur d'évolution de la suite. L'approche calculatoire consiste en l'étude du comportement de la suite pour des nombres suffisamment grands. L'étude porte sur certaines caractéristiques de la suite telles :

- **Le temps de vol :** le plus petit indice $i/u_i = 1$
- **Le temps de vol en altitude :** le plus petit indice $i/u_{i+1} < u_0$
- **L'altitude maximale :** la valeur maximale de la suite.

Un projet implémenté sous la plateforme BOINC tourne depuis 07/2009 à nos jours. Notre idée consiste à porter ce projet sur notre environnement, et de donner accès particulièrement aux analystes / mathématiciens à la base de données contenant les valeurs brutes de la suite en offrant éventuellement des outils statistiques adéquats.

3.5.2 Résultats

Evaluation de la participabilité dans SCVE

Dans une première expérience nous avons exposé notre environnement pour une courte durée en communiquant l'url du serveur à quelques utilisateurs, nous avons, par la suite, extrait l'évolution du nombre de volontaires quotidien et global depuis le panneau de notre application Facebook. La figure 3.4 montre les résultats obtenus comparés avec les données similaires du projet tournant sous BOINC.

On estime que les résultats sont très encourageants et ce malgré la courte durée d'exécution

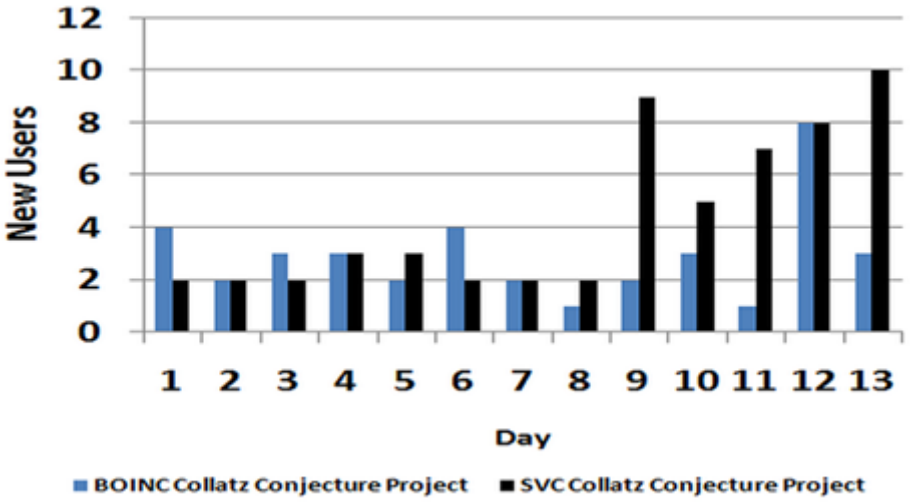

FIGURE 3.4 – Nouveaux Utilisateurs/jours dans SCVE vs BOINC/Collatz.

de notre environnement vu les restrictions dues à la sécurité imposées par la plateforme

Facebook.

Evaluation de l'algorithme TASP

Nous avons également évalué les performances de l'algorithme proposé TASP par rapport à deux politiques d'ordonnancement : FCFS (aveugle) et world communitygrid-like notée Knowledge Based (KB). Une bonne évaluation nécessite l'utilisation de ressources couteuses dont on ne dispose pas, néanmoins, et afin de procéder à une comparaison objective, nous avons adopté la méthodologie suivante :
Pour reproduire l'aspect hétérogène du calcul CV, nous avons opté pour une expérience consistant à calculer la suite de Collatz pour les premiers 100.000 entiers en utilisant trois machines ; un PC de bureau (DuoCore, 4Go de RAM), un laptop (core i3 4ème génération, 4GO de RAM) et un smartphone (moyen de gamme). Le choix des machines est justifié par le fait que ce sont des machines équivalentes à la plupart utilisées par la majorité des volontaire. Le calcul est divisé en 2000 tâches élémentaires, générées à la demande en deux étapes et qui calculent, chacune, la suite de Collatz pour 50 nombres successifs. Lorsque les tâches sont assignées aux volontaires, le calcul se déroule en pseudo-alternance afin d'assurer au mieux l'exécution de tâches équivalentes par les volontaires. En premier lieu, nous avons exécuté le projet en entier d'une manière individuelle sur les trois machines pour avoir une idée préliminaire sur la performance des trois machines. La figure 3.5 montre clairement que l'ordre décroissant M2,M1,M3 en termes de performances.

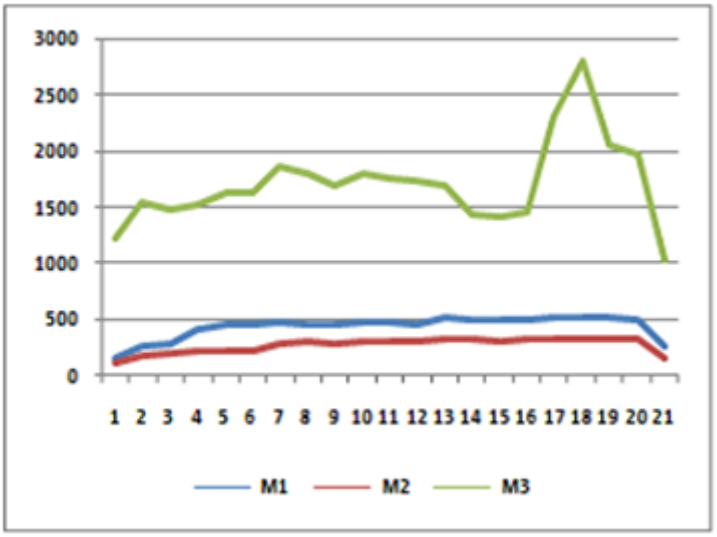

FIGURE 3.5 – Performances des trois machines utilisées.

Pour la politique FCFS, l'ordonnanceur a affecté un nombre conséquent de tâches à la machine la moins performante M3 (700 tâches pour M1, 505 pour M2 et 795 pour M3)

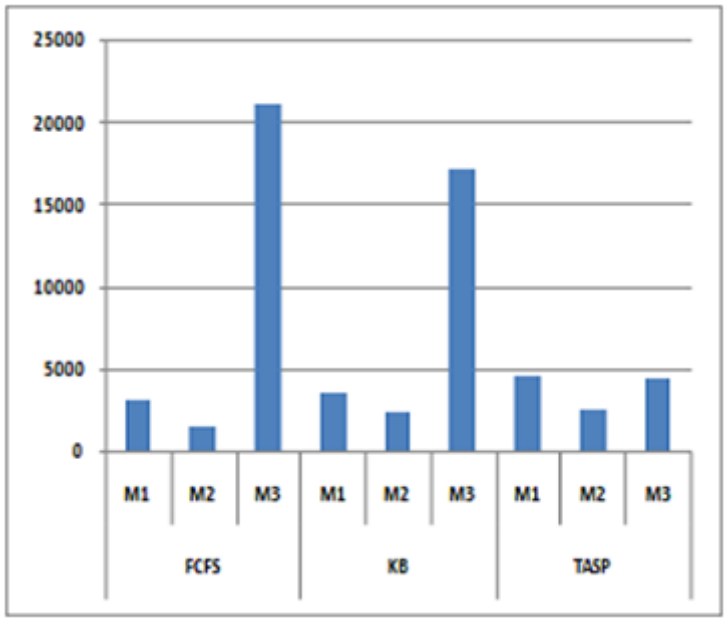

FIGURE 3.6 – Temps d'exécution (secondes) des trois politiques d'ordonnancement.

qui consommait du temps exorbitant (3242,2s pour M1, 1570,8s pour M2 et 21155,8s pour M3). Dans la KB, même si les groupes sont correctement distribués aux bons volontaires adéquats (machines) (700 tâches pour M1, 805 pour M2 et 495 pour M3), la machine la moins puissante M3 a également pris trop de tâches élémentaires car les tâches pré-générées ont une taille fixe et par conséquent, a consommé beaucoup de temps d'exécution (3 635,3s pour M1, 2 423s pour M2 et 17 191,5s pour M3). Contrairement aux deux politiques précédentes, TASP calcule exactement le nombre de tâches élémentaires adéquats en fonction de la puissance réelle de la machine utilisée (946 tâches pour M1, 910 pour M2 et 143 pour M3) et donne un meilleur temps d'exécution (4694,8 pour M1, 2643 pour M2 et seulement 4 542,5s pour M3).

3.6 Conclusion

Dans ce chapitre, nous avons présenté notre deuxième contribution à savoir une politique d'ordonnancement optimale dans les systèmes de calcul volontaire. L'algorithme que nous avons introduit baptisé TASP détermine d'une manière optimale le nombre de tâches élémentaires (résultant d'une décomposition fine du projet de CV) à attribuer pour chaque volontaire selon ses capacités de calcul temporelles. A travers une expérimentation sur un projet mathématique portant sur la conjecture de Collatz, nous avons illustré la suprématie de notre politique par rapport à deux politiques en cours d'utilisation dans les systèmes CV actuels.
La partie suivante de la thèse, divisée en deux chapitres, concerne la simulation distribuée

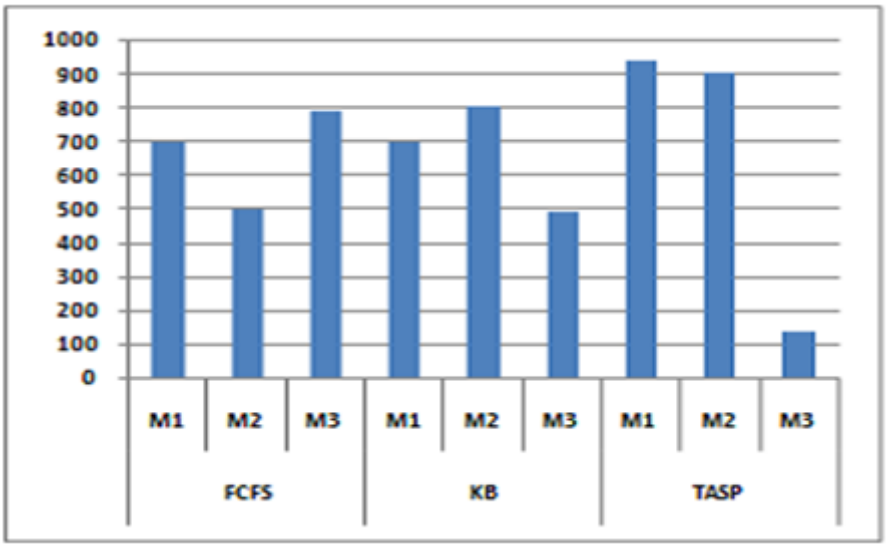

FIGURE 3.7 – Distribution des tâches des trois politiques d'ordonnancement.

et l'utilisation de VolSIM qui est une version adaptée de SVCE pour la simulation.

Deuxième partie

Environnement de simulation basé sur le calcul volontaire

Chapitre 4

La simulation distribuée

4.1 Introdution

La SD est la discipline résultante de l'exécution d'une simulation sur un système distribué. Elle intègre les concepts de la simulation (évènements discrets, états, entités, file d'attentes, etc.) en s'appuyant sur les techniques issues des systèmes distribués (horodatage, horloge distribuée, synchronisation et coordination, passage de messages, consensus, etc.). Dans ce chapitre, après la présentation de la simulation et la simulation distribuée, le standard HLA est étudié ainsi que quelques travaux de recherche récents portant sur la SD.

4.2 Généralités sur la simulation

On trouve plusieurs définitions de la simulation, une discipline assez vieille, dans la littérature. L'idée commune à ces définitions c'est le remplacement fonctionnel d'un système (réel ou virtuel) par un modèle mathématique. L'expression M&S est utilisée très fréquemment pour joindre toujours la technique de simulation avec une activité primordiale qui est celle de la modélisation des systèmes. L'utilisation de la simulation est répandue dans plusieurs domaines (prototypage, évaluation de performances, SIAD, training, réseaux informatiques, applications militaires, aéronautique, etc.). Il est clair que les techniques de simulation offrent des avantages indéniables par rapport à des expériences sur des systèmes réels qui peuvent être coûteuses, dangereuses et voir même impossibles à réaliser comme dans un système où l'échelle de temps de simulation est de quelques centaines d'années.
Le processus de modélisation/simulation peut être représenté par la figure 4.1. Les notions essentielles qu'on trouve dans la M&S sont [Chaudron, 2012] :

- **Le système source :** est l'objet à étudier. Il peut être vu comme une source de données observable (un phénomène physique) ou d'un objet à concevoir,
- **Le cadre expérimentale :** désigne les conditions dans lesquelles l'observation du système est effectuée. Ces conditions doivent être spécifiées de manière à garantir la correspondance avec le modèle en adéquation,
- **Le modèle :** c'est une représentation du système sous des conditions données, il correspond à l'objet manipulable afin de réaliser la simulation. Le modèle est une abstraction du système obtenue à partir d'un ensemble de règles (équations, contraintes logiques, etc.) qui définissent le comportement du système,
- **Le simulateur :** c'est l'entité capable de modéliser l'évolution du système en fonction des spécifications du modèle ; c'est un système de calcul qui peut être physique ou logiciel.

Il est primordial d'avoir un modèle pertinent avec ledit système pour obtenir une simula-

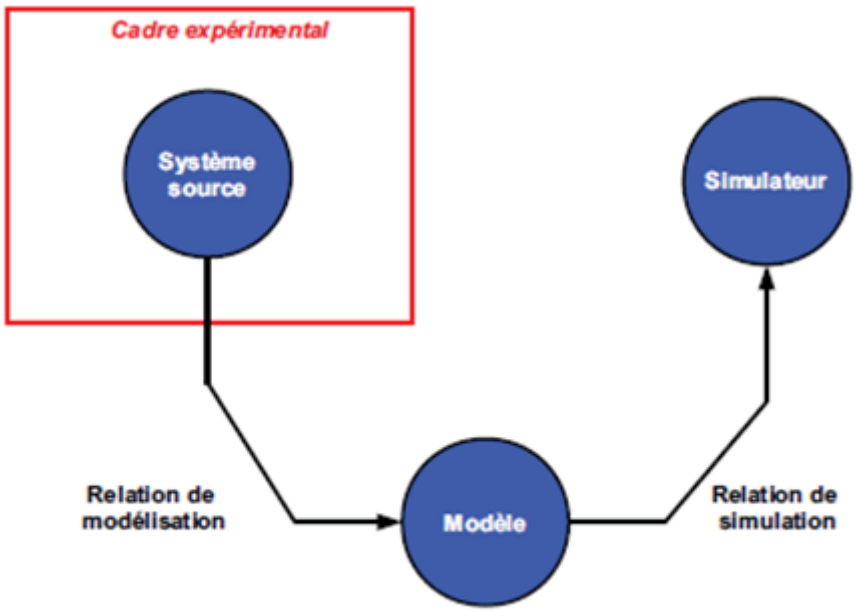

FIGURE 4.1 – modélisation et simulation [Chaudron, 2012].

tion pertinente. Le choix d'un modèle s'effectue selon plusieurs critères et le comportement peut s'exprimer dans divers formalismes reposant sur des propriétés mathématiques selon deux dimensions : spatiale et temporelle. Les propriétés spatiales peuvent être exprimées par des variables avec des valeurs finies ou infinies. L'ensemble des valeurs des variables ,à un instant donné, est appelé état du modèle. Le temps peut être continu ou discret. On distingue plusieurs concepts de temps dans une simulation [Fujimoto, 1990] :

- **Temps physique :** temps de référence du système physique que l'on veut modéliser et simuler,
- **Temps simulé :** est une représentation du temps physique pendant la simulation,
- **Temps absolu :** temps qui s'écoule pendant l'exécution de la simulation.

L'évolution d'une simulation à temps discret peut se faire par deux approches :
La première dirigée par le temps lui-même (synchrone) où on choisit un temps simulé fixe T (qui peut éventuellement être synchronisé sur une horloge de temps absolu) et faire évoluer pas à pas le modèle et ses composants. L'intervalle de temps T doit être choisi judicieusement afin de garantir la cohérence du modèle d'une part et la performance globale de la simulation d'autre part. Un temps T trop petit amène à exécuter des boucles vides, idem pour un temps T trop grand ce qui débouche sur le traitement de plusieurs actions dispersées temporellement en les assumons au même temps, ce qui mène à des incohérences de causalité (précédence cause-effet).
La deuxième approche dite à évènements discrets (asynchrone) n'impose aucun pas de temps aux actions ; le temps n'est avancé que par l'arrivée d'un évènement qui progresse selon une estampille dédiée aux évènements (temps de chaque évènements).
Afin d'assurer la cohérence globale, la gestion de l'entrelacement des actions dans le temps

fait appel à des algorithmes de synchronisation spécifiques.
Une simulation à évènements discrets (DES) est une simulation où le modèle utilisé repose sur un ensemble fini de variables qui évoluent selon un temps discret. Un évènement dans une DES correspond à un changement d'état (changement d'une ou plusieurs valeurs des variables). L'exécution d'une simulation peut être vue comme une succession d'états. Les règles temporelles de la simulation sont :

1. **Déterminisme :** Ce principe fondamental stipule que le futur d'un système peut être (doit-être ?) déterminé à -partir de son état présent et ses états passés. Ce principe peut être traduit par l'équation (4.1).

$$E_{futur}(s) = \sigma(E_{présent}(S), E_{présent}(s)) \tag{4.1}$$

2. **Causalité :** Ce deuxième principe fondamental impose à ce que le futur d'un système ne peut en aucun cas influencer son passé. L'état d'un système à l'instant présent t est indépendant de tout état à l'instant t' tel que $t' > t$. Tout simulateur doit respecter ce principe lors de l'émission des évènements qui devra se faire selon un ordre chronologique ascendant et ce pour refléter le système physique réel.

4.3 La simulation distribuée

4.3.1 Objectifs

L'objectif principal de la simulation distribuée SD est l'utilisation des techniques de programmation parallèle et distribuée et des nœuds multiples afin d'accélérer l'exécution des programmes de simulation et/ou connecter différentes simulations ensembles. Dans certain cas, l'objectif est tout simplement d'exécuter plusieurs scénarios (instances) de la même simulation en parallèle.

4.3.2 Motivations

L'exemple relaté dans [Taylor, 2019] montre bien une des motivations de l'utilisation de la Simulation Distribuée ; Deux sociétés A et B fabriquent une gamme de widgets à vendre pour différents consommateurs. La société A produit les corps widget quant à la société B elle s'occupe de la finition de ces derniers. Les deux entreprises doivent faire face au problème d'équilibrage de la chaine de production, i.e. la société A ne doit pas produire trop de widget (saturation de capacité de stockage de B) et la société B ne doit jamais tomber en rupture de stock. Les deux entreprises utilisent déjà la simulation pour gérer leurs propres processus de production et elles décident d'en créer une seule commune

pour gérer une production équilibrée. Souvent, les deux simulations sont écrites dans deux environnements différents et il est extrêmement difficile de les fusionner en une seule. Il est de même si elles sont écrites dans le même environnement, leur intégration prendrait du temps et très probablement a besoin d'un certain effort de recodage, par exemple dans les cas où il y a :

— Ambigüité des noms de variables et/ou méthodes,
— Différences dans les structures et/ou sources de données,
— Problèmes liés à la gestion de la simulation commune.

De plus, une fois la nouvelle simulation créée, chacune des deux entreprises peut voir le fonctionnement interne de l'autre, ce qui pose un problème de sécurité des données confidentielles de chacune des entreprises.
Dans un tel scénario, la SD prend toute sa place, elle permettra de maintenir les deux simulations séparées et d'échanger uniquement les données qui devront être coordonnées l'une avec l'autre. Plusieurs auteurs ont identifié les arguments d'utilisation de la SD [Boer *et al.*, 2009], [Lendermann *et al.*, 2007], [Mustafee *et al.*, 2012] et [Taylor *et al.*, 2012] :

— **Temps d'exécution :** Une simulation de grande envergure peut être très lente. La SD peut être utilisée afin de diviser cette dernière sur plusieurs nœuds d'exécution pour exploiter le parallélisme et accélérer l'exécution. Elle peut aussi accélérer l'exécution d'une simulation en exécutant plusieurs expérience sur des ordinateurs différents.
— **Composition de modèles de simulation et réutilisation :** cet argument se justifie par le coût de développement des simulations qui peut être très significatif. La réutilisation de simulations existantes dans le cadre d'une nouvelle SD est très attractive dans ce sens. De plus, Il est mieux convenable de lier deux simulation ensemble que leur combinaison avec tous les problèmes de génie logiciel qui accompagnent un développement « from scratch ».
— **Maintenance propriétaire :** La SD permet aux développeurs de maintenir et/ou de mettre-à-jour leurs simulations individuelles indépendamment des autres.
— **Confidentialité technologique :** La SD permet de maintenir la confidentialité des secrets technologiques liés à ses différentes simulations individuelles qui la composent (fonctionnement interne d'une usine, d'un hôpital ou d'un système militaire). Une intégration en une seule simulation centralisée rendra visible des parties confidentielles à toute personne exécutant cette simulation. Donc la SD permet l'interaction entre plusieurs autres simulations qui peuvent rester dans leur état de « Boite noire ».
— **Intégrité et Confidentialité des données :** Similaire au problème précédent avec les données des simulations individuelles qui peuvent comporter du code pour accès à des données hautement confidentielles (mot de passes d'une base de données

ou d'une session sur réseau). L'intégration dans une simulation centralisée rend ce code visible, par contre, la SD permet aux différentes parties individuelles qui la composent de garder leur code d'accès comme privé et n'échanger avec les autres parties que les données concernées par l'intégration.

— **Simulation Hybride :** Cet argument est évident du fait de l'existence de plusieurs packages commerciales permettant de réaliser des simulations, ces derniers sont très variés et d'une grande hétérogénéité. La SD permet une grande interopérabilité entre des simulations issues d'environnements divers (à base d'agents, à évènement discret, etc.).

4.3.3 Les modes de distribution de simulation

Les modes de distribution de simulations [Robinson *et al.*, 2004] peuvent être répartis en trois catégories :

— **Mode A :** Une simulation qui devrait se dérouler sur un seul nœud est divisée en plusieurs parties ou sous-simulations qui s'exécutent sur différents nœuds et interagissant via un réseau de communication.

— **Mode B :** Il s'agit dans ce mode de permettre à plusieurs simulations à part entières et s'exécutant sur des nœuds différents de communiquer et d'interagir. Ce mode permet la réutilisation des simulations existantes et d'épargner l'effort de redéveloppement de nouvelles simulations.

— **Mode C :** Dans ce cas, il s'agit d'exécuter en parallèle plusieurs expérimentations (scénarios) de la même simulation sur plusieurs nœuds, autrefois exécutées séquentiellement sur un seul nœud. La figure 4.2 illustre les trois modes de distributions discutés.

Il convient de noter que le simple fait de diviser une simulation entre différents processeurs ne garantit pas automatiquement l'accélération. Les frais généraux de coordination entre les simulations pourraient entraîner une exécution plus lente qu'une seule simulation. De même, la réutilisation d'une simulation dans le cadre d'une SD n'est pas non plus automatique, surtout si cette simulation n'a pas été conçue à cette fin [Robinson *et al.*, 2004]. Actuellement, aucun progiciel de simulation commercial ne possède de fonctionnalités de la SD prêtes à l'emploi. Celles-ci doivent être modifiées pour permettre aux modèles d'interagir sur un réseau de communication (qui peut être complexe).
En outre, pour réaliser un projet de SD réussi dans le domaine de la recherche opérationnelle par exemple, des connaissances sont nécessaires qui combinent la SD, la recherche opérationnelle, les réseaux et technologies informatiques ainsi qu'une expérience en programmation significative.

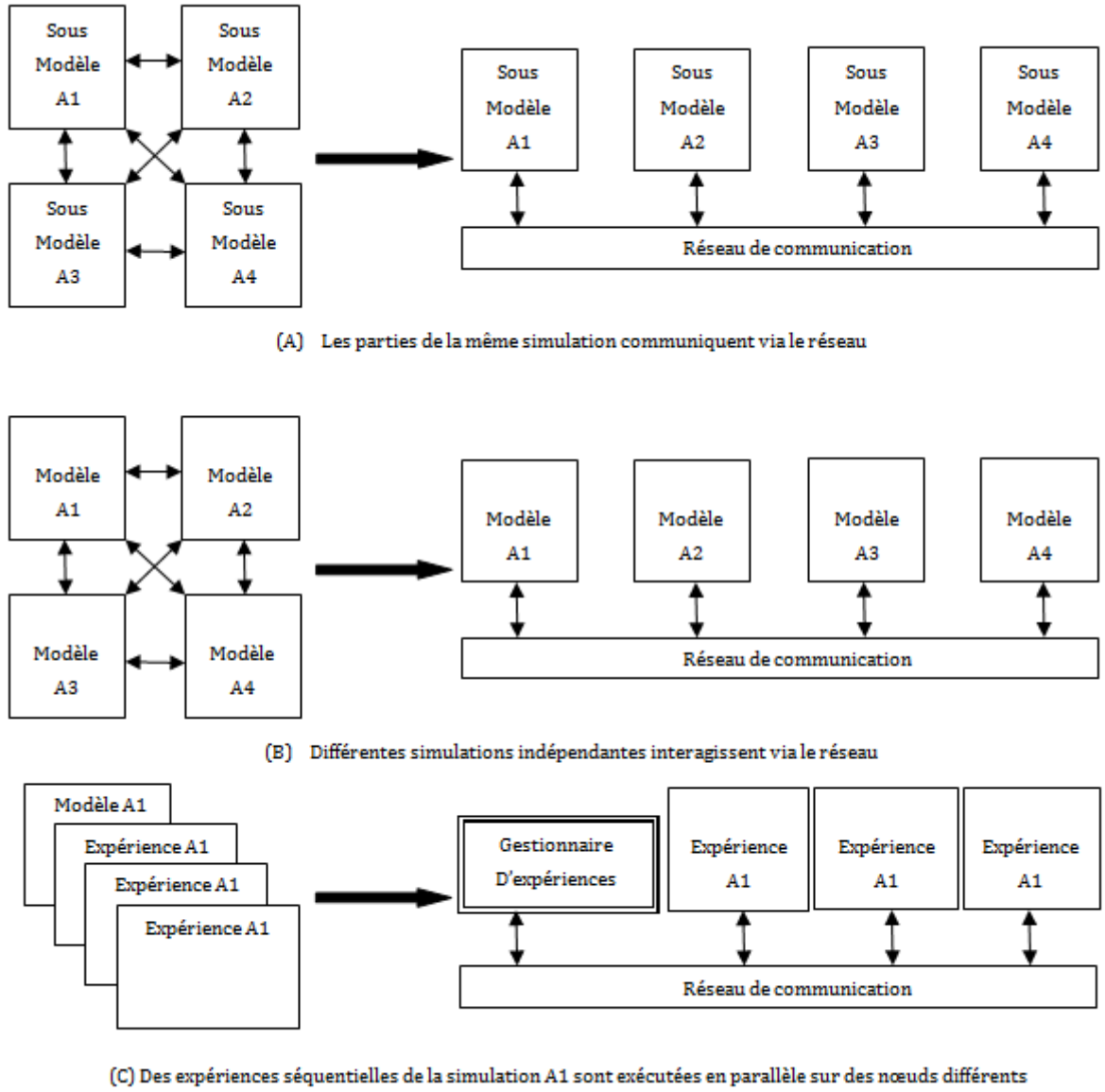

FIGURE 4.2 – Modes de distribution de simulations.

4.4 Fonctionnement d'une simulation distribuée

Pour le mode (C), détaillé dans la section précédente, où la SD peut être utilisée pour accélérer l'expérimentation d'une simulation donnée, le principal problème est d'ordre conceptuel ; comment distribuer et coordonner l'exécution des expérimentations de simulation sur différents ordinateurs ?

L'un des premiers travaux qui ont proposé une base théorique pour l'exécution des expérimentations d'une simulation est proposé dans [Heidelberger, 1986]. Bien que l'utilisation de nombreux ordinateurs pour exécuter des expérimentations en parallèle soit attrayante, le coût de parallélisation et/ou de distribution est considérable (ce coût inclus la mise en place du simulateur ainsi que l'envoi de la simulation et de ses paramètres).

Dans une SD selon les deux modes (A) et (B), le principal problème réside dans la synchronisation du temps de simulation entre les unités d'exécution (ordinateurs) distribuées. Les approches utilisées sont celles dérivées de la coordination et de la synchronisation des processus dans les systèmes distribués [Tanenbaum et van Steen, 2006]. Trois classes d'algorithmes sont à distinguer : La première classe porte sur la synchronisation des horloges physiques par communication pour coordonner l'exécution en temps réel des processus, tandis que la deuxième classe s'articule sur un ordre logique des actions effectuées par chacune des entités d'un système distribué. L'absence d'interaction entre deux entités d'un tel système implique l'absence de synchronisation entre elles, toutefois les actions de ces deux dernières qui pourraient affecter d'autres entités du système doivent être ordonnées. [Lamport, 1978] décrit un algorithme utilisant les horloges logiques, l'horodatage des actions et les relations de précédence pour imposer un ordre partiel d'actions dans les systèmes distribués.
Un exemple de problème de synchronisation d'horloges propre à la simulation distribuée est relaté dans [Taylor, 2019]. Trois files d'attente à serveur unique (SSQ) : SSQ1, SSQ2 et SSQ3 où chaque entité est traitée en fonction d'un temps de service au niveau de SSQ1 ou SSQ2 puis envoyée à SSQ3. Chaque SSQ s'exécute sur un nœud différent (3 ordinateurs dans ce cas). Chaque simulation possède ses propres éléments à part entière (état, routines d'événements, liste d'événements, horloge, environnement d'exécution, etc.) Le transfert d'une entité d'un serveur vers un autre se traduit par un message d'événement horodaté au serveur destinataire.
Supposons que SSQ1 et SSQ2 traitent des entités dans un temps de 10 et 12 minutes respectivement. Lorsque chacun des deux serveurs termine le traitement d'une entité, il l'envoi immédiatement au serveur SSQ3 (on suppose que le temps de transite est nul). Si on commence la simulation au temps t=0 ; les entités arriveront dans l'ordre suivant à SSQ3 : e1 à 10 (instant $t1$), e2 à 12 (instant $t2$), e3 à 20 (instant $t3$), e4 à 24 (instant t_4), e5 à 30 (instant $t5$), etc. (c'est-à-dire $t1 < t2 < t3 \ldots$). Ces interactions sont illustrées dans la figure 4.3. TEM1 et TEM2 représentent les messages de transfert des évènements e1 et e2.

Étant donné qu'il n'y a pas de temps de coordination d'horloge de simulation unique sur la SD, Aucune supposition ne peut être faite sur le temps d'arrivée des messages TEM1 et TEM2 au serveur SSQ3. Il est nécessaire de mettre en place un mécanisme de coordination permettant de garantir le traitement des entités émises par SSQ1 et SSQ2 dans l'ordre logique en fin de simulation.
En l'absence d'horloge commune, rien n'oblige SSQ3 à traiter l'évènement e2 avant l'évènement e1 (ceci dépendra éventuellement des temps d'arrivée des messages TEM1 et TEM2.
Dans une SD, les entités SSQ1, SSQ2 et SSQ3 sont complètement indépendante et peuvent s'exécuter sur des machines différentes reliées par un réseau de communication avec un

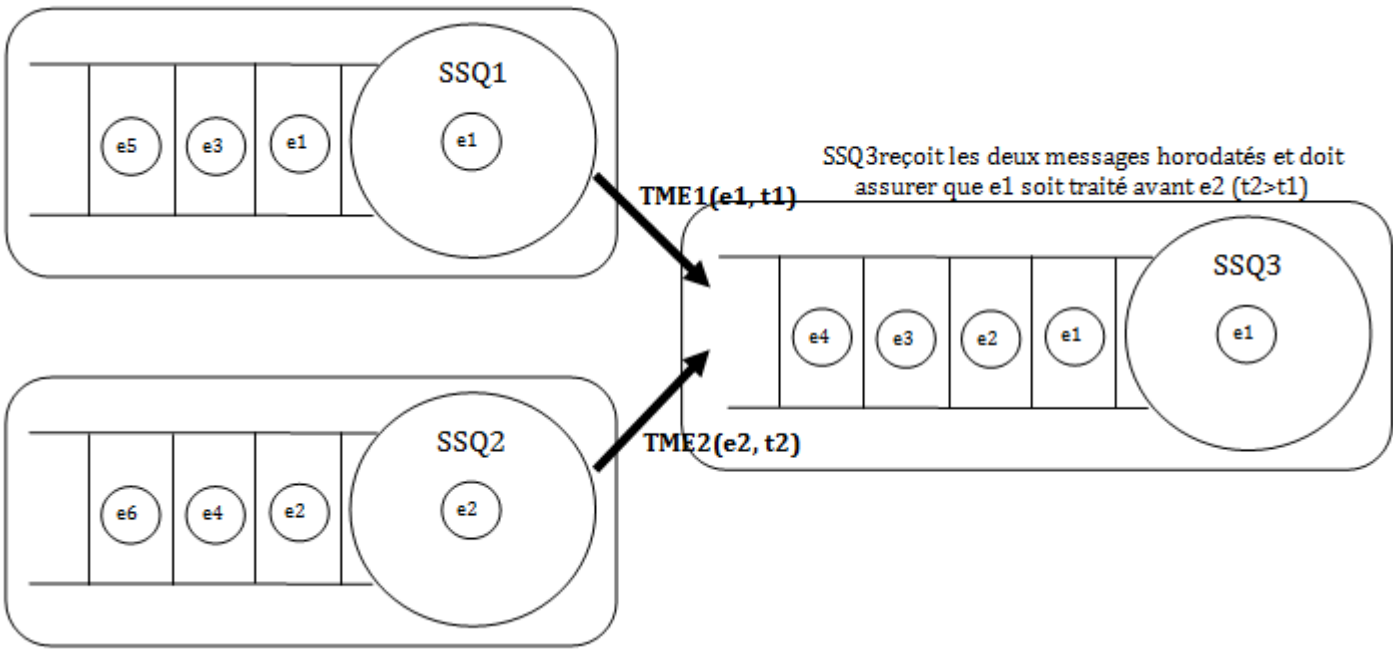

FIGURE 4.3 – Exemple de simulation distribuée simple [Taylor, 2019].

protocole de remise arbitraire.

Dans une SD, un Processus Logique (PL) correspond à une partie de ladite simulation. Les PL communiquent via des messages horodatés. Le but d'un algorithme de gestion du temps est de s'assurer qu'à la fin de la simulation, tous les PL ont traité leurs événements dans l'ordre logique préconisé, et pour les événements « internes » produits par le PL et pour les évènements « externes » reçus par le PL via les messages horodatés. Cette contrainte est appelée la contrainte de causalité. La simulation globale sera correcte si tous les PL qui la composent respectent, chacun localement, la contrainte de causalité.

Un autre exemple du problème de causalité dans le cas d'une simulation distribuée est relaté dans [Fujimoto, 1990] illustré dans la figure 4.4. trois PL P1, P2 et P3 modélisant une simulation militaire et évoluant indépendamment les uns des autres. Les temps de simulation des trois processus sont, bien entendu, différents. P1 émis un évènement T correspondant au tir d'un projectile, P2 et P3 reçoivent cet évènement RT à des temps différents. P3 émis un évènement D correspondant à la destruction du projectile P1 et P3 reçoivent l'évènement RD. Pour P2 l'évènement de destruction RD est reçu avant l'évènement de tir RT du projectile ce qui met en cause la cohérence de la simulation.

Selon la manière adoptée pour gérer la causalité, on peut distinguer trois grandes approches :

1. **Approche Préventive :**
   La première approche, dite préventive ou conservatrice, où il est imposé aux PL de respecter scrupuleusement la causalité, Ces algorithmes ont été les premiers à être

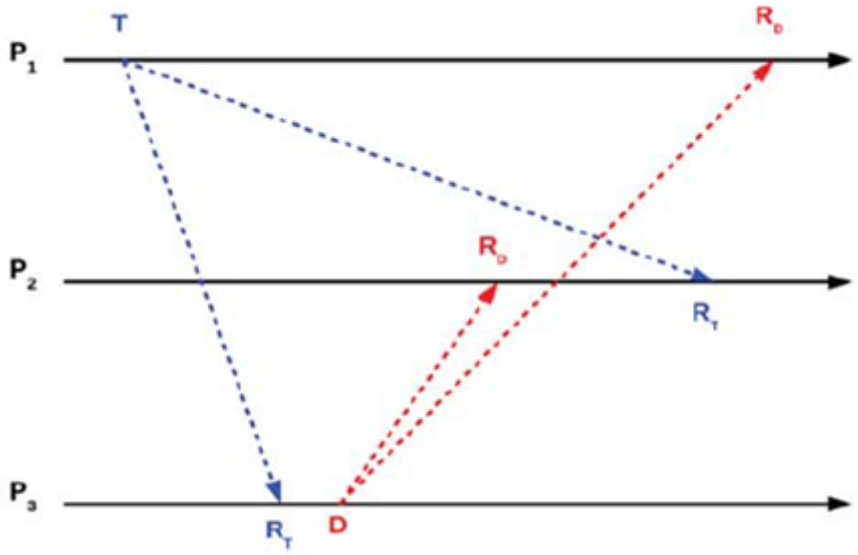

FIGURE 4.4 – Exemple de problème de causalité dans la SD.

développés à la fin des années 1970 [Bryant, 1977], [Chandy et Misra, 1979], [Misra, 1986]. Le qualificatif préventif de ces algorithmes est dans le sens où ils évitent les erreurs de gestion du temps. Dans l'exemple précédent de la figure 4.3, si SSQ3 est à l'instant 10, reçoit TEM1 qui représente l'arrivée de e1 à t1 et la traite, puis il reçoit TEM2 représentant l'arrivée de e2 à t2 (temps de e2=12), il ne traitera pas TEM2 sans confirmer la sûreté de l'action (c'est-à-dire aucune occurrence d'événements avant le temps 12). Cette approche suppose que les PLs et leur interconnexion (topologie) sont fixés et que chaque PL envoie uniquement des messages dans un ordre chronologique croissant et que le réseau de communication fourni les messages dans leur ordre d'envoi. Par conséquent, chaque PL doit gérer localement un ensemble de files d'attente d'entrée pour les messages des autres PL. Chacune de ces files contient un ensemble de messages dans un ordre d'horodatage croissant. Cet ordre est maintenu afin de choisir le prochain message sûr à traiter selon un cycle de vérification répété (traitement de l'ancien message le plus sûr sans précédents, avancer le temps de simulation puis traiter le suivant et ainsi de suite d'une manière identique à une simulation séquentielle).
Outre l'horodatage, la solution de Chandy et Misra [Chandy et Misra, 1979] repose sur la connaissance de la topologie locale du système. Chaque processus doit en effet identifier l'ensemble des PLs à l'origine des messages qu'il reçoit. Chaque canal de communication entre processus logiques est matérialisé par une file d'attente côté consommateur où viennent s'empiler, dans l'ordre de leur date, les messages non-encore consommés. A chaque canal on lui attribue la date de son message de tête qui est par conséquent le plus ancien. A défaut de message, il conserve la date du dernier message qui a été émis. A chaque cycle de simulation, le processus consomme

le message de tête du canal le plus ancien. L'exécution de l'évènement correspondant produit éventuellement de nouveaux messages dont la date est nécessairement postérieure au message initial. Si le canal le plus ancien est vide, le processus est bloqué jusqu'à réception d'un nouveau message. Ce mécanisme prémunit totalement contre les violations de causalité. Un processus ne peut en effet progresser que si ses PLs voisins (en interaction) sont temporellement en avance par rapport au processus en question.
Comme il est possible qu'un inter-blocage se produit dans ce schéma (files d'attente vide et attente cyclique entre PLs), des messages nuls horodatés sont envoyés pour permettre aux PLs d'avancer leur temps de simulation et émettront éventuellement des évènements pour les autres, en l'absence d'évènements, le PL génère à son tour un message nul. Le choix du temps d'envoi des messages nuls(appelé anticipation) doit être judicieusement choisi. Un temps mal choisi peut généré un grand nombre de messages nuls et cause une dégradation brutale des performances (de calcul et de communication).
De nombreuses variantes sont proposées dans ce sens afin d'améliorer le mécanisme d'anticipation dans les algorithmes préventifs appliqués à des applications particulières telles que la topologie d'interconnexion des PLs [Kumar, 1986] « lookahead conditionnel » [Fu *et al.*, 2014]. La gestion efficace du temps de simulation et de la causalité est un domaine de recherche encore actif.

2. **Approche Optimiste :**
La deuxième approche, appelée l'approche optimiste ou curative, est celle où l'on suppose que les évènements arrivent dans le bon ordre, en cas d'erreur de chronologie, des mécanismes d'annulation et de reprises des actions depuis l'évènement erroné. Contrairement aux algorithmes préventifs, les algorithmes curatifs permettent aux erreurs de causalité de se produire, et procéder aux corrections nécessaires de telle façon que tous les événements soient correctement traités dans l'ordre à la fin de la simulation. L'avantage principal de cette approche est de générer plus de parallélisme pour traiter la simulation plus rapidement sous la supposition que les erreurs d'horodatage sont moins fréquentes. L'algorithme curatif le plus connu est l'application du concept de temps virtuel détaillé dans [Jefferson, 1985] dans la SD dans le mécanisme « Time Warp » ; chaque PL maintient localement trois files d'attente : une pour les messages entrants, une pour les messages sortants et une troisième pour les états. Les messages d'événement sont échangés entre les PLs et sont stockés dans leurs files d'attente. Les messages d'événement sont traités au fur et à mesure qu'ils arrivent dans l'espoir qu'ils arriveront dans le bon ordre. Chaque fois qu'un message d'événement est traité, un nouvel état est créé (reflétant le contexte du PL à l'instant t) et stocké dans la file d'attente d'état.

Si un événement hors temps est rencontré, le message est traité puis le PL revient au dernier état sûr avant l'horodatage de l'événement fautif (comme pour les roll-back dans les SGBD transactionnels). Les messages sortants depuis l'évènement en question sont réémis avec un indicateur négatif, un message avec un indicateur négatif, appelé anti-message, provoque à son tour un « rollback » au niveau du PL destinataire et ainsi de suite. Les inconvénients principaux de l'approche curative peuvent être résumés en :

— Le stockage et l'accès à un grand nombre d'états de simulation
— Les cascades de restauration de contexte des PL dus aux rollbacks imbriqués.

Afin de limiter les avalanches des rollbacks, un protocole assurant un temps de référence appelé Global Virtual Time (GVT) peut être utilisé pour établir une limite inférieure sur le temps global - cela signifie que les PLs ne reviendront pas à une date antérieure à ce temps et donc les états de simulation avec un horodatage antérieur peuvent être éliminés. De nombreuses variantes ont été proposées dont une bonne partie entre elles visent à réduire le nombre de restaurations en cascade. A titre d'exemple des solutions apportées ; l'ignoration des restaurations inutiles en cas d'arrivée d'un événement hors temps qui ne change pas l'état de la simulation et l'introduction des méthodes plus efficaces pour le calcul du GVT. Ces variantes peuvent être documentées à titre non exhaustif dans les travaux [Fujimoto, 1990], [Rönngren et Ayani, 1994], [Damani *et al.*, 1997], [Chetlur *et al.*, 1998], [Moreira *et al.*, 2005] et [Venu et Joe, 2014].

3. **Approche Temps réel :**
Une alternative au deux approches préventives et curatives est la simulation en temps réel où une simulation est exécutée si c'était le monde réel (ou au moins donner une réponse en temps réel aux actions d'un utilisateur). Plusieurs simulations doivent interagir avec plusieurs autres simulations aussi tôt que possible. Cet objectif nécessite différentes approches pour équilibrer les traitements et les communications. Ces algorithmes n'utilisent pas la gestion du temps comme vu précédemment.
Historiquement, l'introduciton de l'approche temps réel en simulation distribuée était motivée par la nécessité de connecter et réutiliser des simulations militaires très coûteuses. Le projet Simulation Networking Programm (SIMNET) a été financé par la Defense Advanced Research Project Agency (DARPA) en 1983 (puis par l'armée américaine) et représente une étape majeure dans l'évolution de la SD en temps réel. Le projet consiste à l'interopérabilité entre 250 simulateurs répartis sur neufs sites de training (5 aux USA et 4 en Europe). SIMNET représente le premier effort de normalisation de la SD temps réel en spécifiant le protocole d'interaction des simulations, le format des données échangées et le support logiciel. Les

protocoles de communication développés dans SIMNET ont conduit à la norme IEEE 1278-1993 du protocole standard de simulation interactive distribuée [IEEE, 1993] approuvé puis remplacé par de nouvelles versions et extensions.
Sur la base de ces efforts, le Defense Modeling and Simulation Office (DMSO) a introduit en 1996 une proposition initiale pour l'architecture de haut niveau HLA, une norme pour soutenir la réutilisation et l'interopérabilité des simulations distribuées [Dahmann *et al.*, 1997]. La proposition a été ratifiée officiellement en tant que standard pour la modélisation et la simulation en 2000 (mise à jour en 2010). HLA est en fait une suite de normes spécifiant un framework et des règles [IEEE, 2010b], la définition d'un support logiciel en l'occourence le (RunTime Infrastructure (RTI)) [IEEE, 2010a], la définition des données (Object Model Template (OMT)) [IEEE, 2010c] et le processus de développement [IEEE, 2011]. La suite de normes est organisée et élaborée par un IEEE Task-group tandis que l'activité de normalisation de l'organisation des normes d'interopérabilité de simulation est assurée par le comité SISO (SISO SAC).
Il est normal qu'un ensemble de normes ait une communauté de soutien. SISO a également produit plusieurs normes à l'appui de la HLA. La majorité de ces normes est spécifique à la défense (par exemple, les formats de données pour les militaires véhicules, armes, etc.). Vu son importance et son utilisation large, la norme HLA est détaillée séparément dans la section 4.5.

4.5 HLA : Standard pour la simulation distribuée temps réel

L'architecture de haut niveau (HLA) fournit un cadre général dans lequel les développeurs de simulation peuvent structurer et décrire leurs applications de simulation [IEEE, 2010b]. La flexibilité est au cœur de cette norme qui vise à satisfaire deux objectifs clés :

1. Promouvoir l'interopérabilité entre les simulations.
2. Faciliter la réutilisation des modèles dans des différents contextes.

Dans la terminologie HLA, une fédération désigne la simulation globale, une simulation fédérée ou tout simplement un fédéré désigne d'une manière générale un des composants de la fédération ; il peut être une application de simulation unique ou un collecteur de données ou un moniteur de visualisation ou tout autre composant.
Trois composants principaux (plus le RTI) sont décrits dans l'ensemble des produits formant le HLA :

1. Le premier composant est le cadre HLA et la spécification des règles qui assurent la

bonne interaction des fédérés dans une fédération et définissent les responsabilités des fédérations et des fédérés,

2. Le deuxième composant est l'OMT. C'est une base nécessaire pour réutiliser et former une norme de documentation décrivant les données utilisées par un modèle particulier.
3. Le troisième,c'est la spécification d'interface des fédérés, traite l'interopérabilité et décrit une interface générique de communication permettant de connecter et de coordonner des modèles de simulation. Bien que le HLA soit une architecture, pas un logiciel, l'utilisation d'un logiciel d'infrastructure RTI est nécessaire pour prendre en charge les échanges dans une fédération.
4. Le logiciel RTI fournit un ensemble de services, tels que définis par la spécification d'interface des fédérés. Le RTI est utilisé par les fédérés pour coordonner les opérations et échanger des données pendant l'exécution.

HLA est une architecture qui comprend des spécifications et des règles à respecter par les implémentations. A titre documentaire, la norme 2010 a inclut dix règles ; les cinq premières concernent les fédérations et les cinq dernières s'appliquent aux fédérés :

— R1 : Les fédérations doivent avoir un Federation Object Model (FOM) HLA, documenté conformément au HLA OMT.
— R2 : dans une fédération, toutes les instances des objets associés à la simulation doivent être dans les fédérés, et pas dans le RTI.
— R3 : Durant l'exécution d'une fédération, tout échange de données FOM entre les fédérés participants doit se faire via le RTI.
— R4 : Durant l'exécution d'une fédération, les fédérés participants doivent interagir avec la RTI conformément à la spécification d'interface HLA.
— R5 : Durant l'exécution d'une fédération, un attribut d'instance doit appartenir au plus à un fédéré participant à un moment donné.
— R6 : Les fédérés doivent avoir un Simulation Object Model (SOM) HLA documenté conformément au HLA OMT.
— R7 : Les fédérés doivent pouvoir consulter/mettre à jour les attributs d'instance et envoyer/recevoir des interactions, comme spécifié dans leurs SOM.
— R8 : Les fédérés doivent pouvoir transférer et/ou accepter l'appropriation des attributs d'instance dynamiquement pendant l'exécution de la fédération, comme spécifié dans leurs SOM.
— R9 : Les fédérés doivent pouvoir modifier les conditions (par exemple, les seuils) dans lesquelles ils fournissent des mises à jour des attributs d'instance, comme spécifié dans leurs SOM.
— R10 : Les fédérés doivent être capables de gérer l'heure locale de manière à leur permettre de coordonner l'échange de données avec les autres membres d'une

fédération.

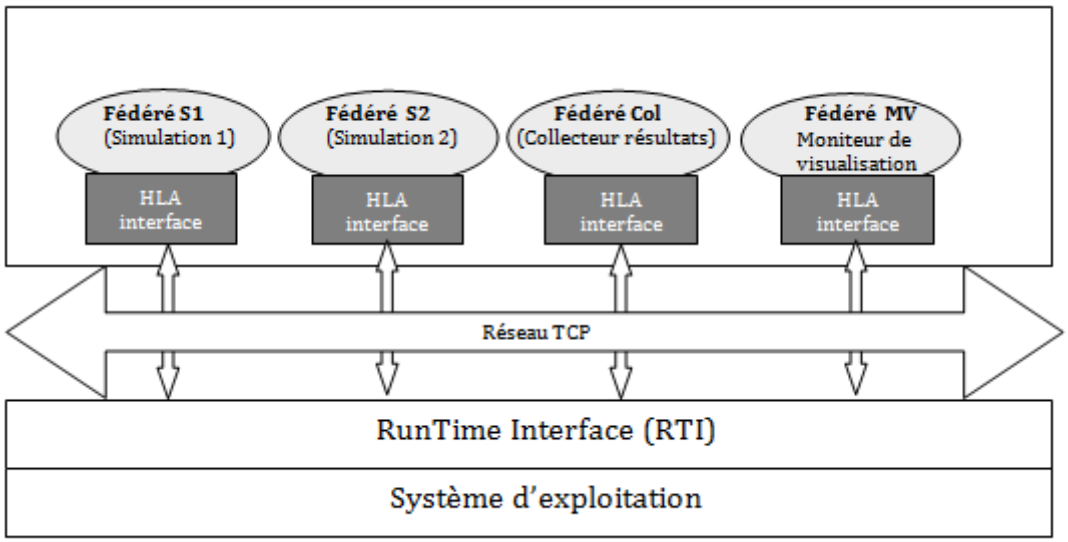

FIGURE 4.5 – Architecture HLA.

La figure 4.5 représente une vue de l'interaction entre les différents composants HLA. Les échanges d'informations entre fédérés s'effectuent exclusivement à travers le RTI. Les règles régissant ces échanges sont spécifiées dans [IEEE, 2010a] qui inclut des définitions des concepts d'interaction entre fédérés HLA dont les plus importants :

- **L'OMT :** spécification système définie principalement par des caractéristiques et des relations de classe. L'OMT HLA est similaire au concept qu'on trouve dans la littérature de l'approche orientée objet mais pas identique à cette dernière.
- **Le FOM :** spécification définissant les informations échangées au moment de l'exécution d'une fédération. Ces informations incluent les classes d'objets, les attributs de classe d'objets, classes d'interaction, paramètres d'interaction et d'autres informations pertinentes à la simulation en question. Le FOM est spécifié au niveau du RTI en utilisant un ou plusieurs modules. Le RTI assemble un FOM en utilisant ces modules et un gestionnaire du modèle d'objet Management Object Model (MOM) et module d'initialisation Management Initialization Module (MIM), qui est fourni automatiquement par le RTI ou, éventuellement, fourni par le RTI lors de la création de la fédération.
- **Le SOM :** spécification des types d'informations qu'un fédère individuel pourrait fournir aux autres fédérés ainsi que les informations qu'un fédéré peut recevoir d'autres fédérés dans les fédérations HLA. Le SOM est spécifié à l'aide d'un ou plusieurs modules SOM. Le format standard dans lequel les SOM sont exprimées facilite la participation du fédéré dans les fédérations.
- **Souscrire (ou s'abonner) :** annoncer les informations qu'un fédéré souhaite obtenir d'une fédération.

— **Publier :** Pour annoncer les informations qu'un fédéré peut fournir à la fédération.
— **interaction :** action explicite prise par un fédéré qui peut avoir un effet ou un impact sur un autre fédéré dans une exécution de fédération.
— **Axe temporel HLA :** une séquence de valeurs totalement ordonnée dans laquelle chaque valeur représente généralement un instant HLA dans le système physique modélisé. Pour deux points quelconques T1 et T2, sur l'axe du temps HLA, si T1 ¡T2 alors T1 est l'instant qui précède T2.
— **Ordre d'horodatage Time Stamp Order (TSO) :** un ensemble de messages ordonnés fournis par la RTI pour les fédérés et activant l'utilisation des services de gestion du temps de la fédération au niveau de ces derniers. Les messages ayant différents horodatages seraient délivrés selon TSO pour le même fédéré. Les messages ayant le même horodatage seront livrés dans un ordre arbitraire (c'est-à-dire, pas de mécanisme d'arbitrage dans le RTI).
— **Temps logique :** point actuel d'un fédéré sur l'axe du temps HLA. Un fédéré utilisant des services de gestion du temps d'une fédération suit les restrictions sur les horodatages TSO (par rapport à leur temps logique) pour garantir que les fédérés qui reçoivent ces messages les reçoivent selon le TSO.

D'autres concepts tels que les attributs, les dimensions, les classes d'interaction ... sont également définis pour la HLA. Une description détaillée du OMT est relatée dans le standard HLA [IEEE, 2010c]. La description comporte le format et la syntaxe d'enregistrement des informations dans l'OMT HLA, pour inclure des objets, des attributs, des interactions et des paramètres. L'OMT est la brique de base conceptuelle de définition des SOM (fédérés) et du FOM (fédération).

4.5.1 Fonctionnement

Gestion des fédérations et des fédérés

La gestion des fédérations fait référence à la création, à l'exécution, au contrôle dynamique, à la modification et à la suppression de fédérations. Avant toute interaction qui peut y avoir lieu entre un fédéré et le RTI, comme la création d'une instance (exécution) d'une fédération ou rejoindre une fédération existante, le fédéré non joint doit effectuer une connexion à la RTI. Le fédéré se déconnecte du RTI lorsqu'il termine son exécution ou dans le cas où son interaction avec les autres n'est plus pertinente. Un fédéré peut invoquer la création d'une instance d'une fédération soit immédiatement ou en différé (IMMEDIATE ou EVOKE callbacks). La figure 4.6 illustre le cycle de vie d'une fédération. Comme le montre la figure 4.7, l'état initial de tout fédéré est «Non connecté». Avant qu'un fédéré puisse interagir avec un RTI, afin de créer par exemple une instance de

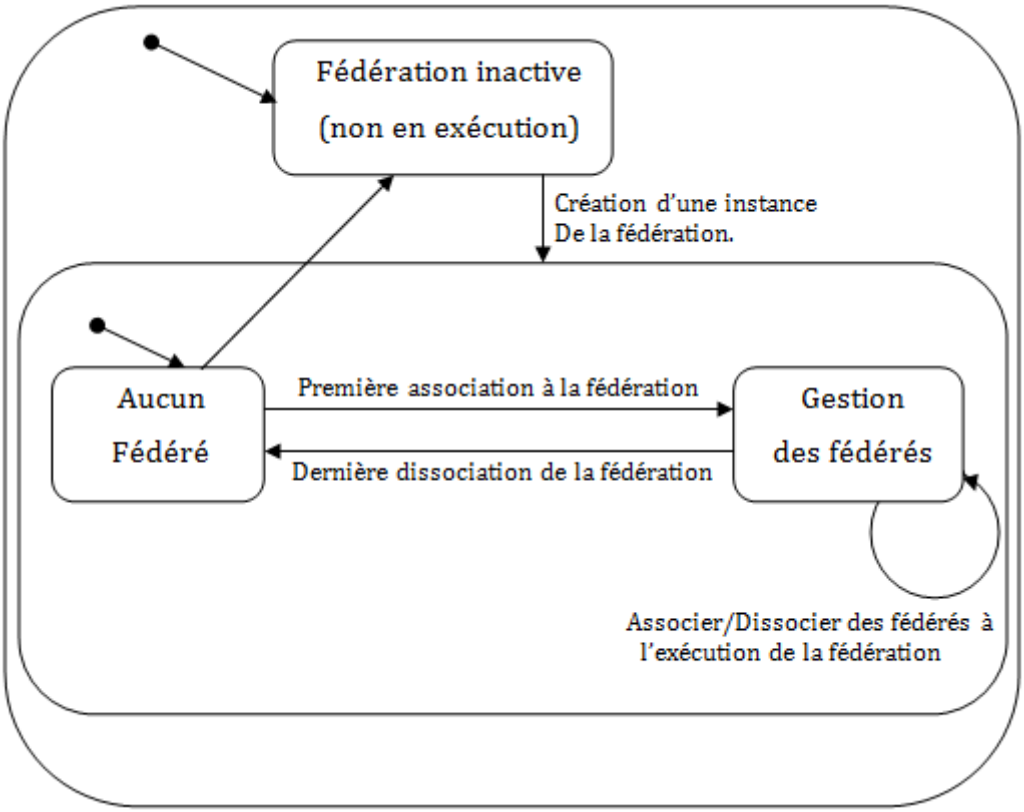

FIGURE 4.6 – Cycle de vie d'une fédération HLA.

fédération ou d'en joindre une, le fédéré doit effectuer une connexion au RTI. Le service « Connect » permet au RTI de configurer et d'initialiser son infrastructure de communication. Le fédéré passe à l'état connecté mais « Dissocié » de toute fédération. Après une invocation réussie du service « Join Federation Execution », un fédéré passe à l'état « Associé » où se déroulent les activités de simulation distribuée. Si un problème d'infrastructure est détecté entretemps, le service « Connection Lost » informe le fédéré qu'un défaut a été détecté, ce dernier revient à l'état « Non connecté ». Si ce fédéré est associé à une fédération, il sera considéré comme dissocié automatiquement. Après des activités de simulation réussies, le fédéré fait appel au service « Resign Federation Execution » afin d'abandonner l'instance de la fédération et passe à l'état « Dissocié ». S'il n'a pas l'intention de rejoindre de nouveau une fédération, il doit effectuer une déconnexion du RTI qui procède au nettoyage adéquat de son infrastructure de communication. De plus, la gestion des fédérations et des fédérés inclut plusieurs fonctions supplémentaires, aussi importantes que les précédentes, telles que l'annonce des points de synchronisation, la sauvegarde et la 'restauration des états des fédérés associés.

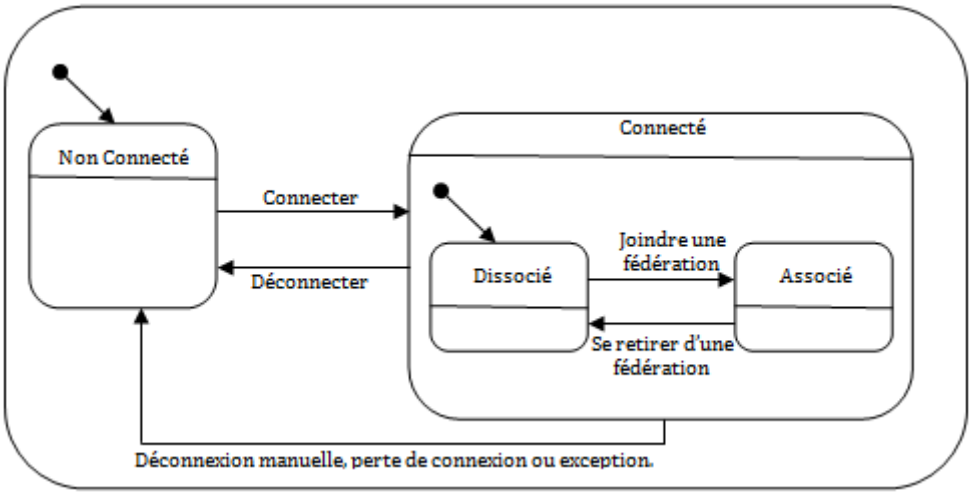

FIGURE 4.7 – Cycle de vie d'un fédéré.

Exemples de services HLA

La Table 4.1 illustre, d'une manière non exhaustive, quelques services avec les informations nécessaires à leur invocation (paramètres, argument de retour, pré et post conditions et les exceptions susceptibles d'être levées à l'appel des services).

TABLE 4.1: Exemple de services HLA : gestion des Fédérations/Fédérés.

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Connect	Modèle de callback, paramètres locaux (option)	Néant	Fédéré non connecté	Fédéré connecté
Disconnect	Néant	Néant	Fédéré connecté, fédéré dissocié.	Fédéré déconnecté.
Connection Lost	Texte descriptif de l'erreur.	Néant.	Fédéré associé	Fédéré notifié
Create Federation Execution	Nom de fédération, Ensemble de FOM, MIM (option), Implémentation, TL(option)	Néant	Fédéré connecté au RTI	Instance d'une fédération avec un nom
Destroy Federation Execution	Nom de fédération	Néant	Fédéré connecté au RTI, instance de fédération existante, pas de fédéré associé à la fédération à supprimer	Fédération supprimée (inexistante)
List Federation Execution	Néant	Néant	Fédéré connecté au RTI	Néant

Suite sur page suivante ...

TABLE 4.1 – *Suite...*

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Join Federation Execution	Nom du fédéré(optionnel, peut être attribué par RTI), type de Fédéré, nom de fédération, ensemble de FOM (optionnel)	Handle de fédéré associé	Fédéré connecté au RTI, Instance de fédération existante, fédéré non associé avec ladite fédération, nom de fédéré unique, fédéré non en état de sauvegarde, fédéré non en état de restauration	fédéré associé à l'instance de fédération en question
Resign Federation Execution	Céder la propriété de tous les attributs d'instance possédés, supprimez toutes les instances d'objet pour lesquelles le fédéré associé a le privilège de suppression, annulez toutes les acquisitions de propriété d'attribut d'instance en attente.	Néant	Le fédéré est connecté au RTI, l'exécution de la fédération existe, le fédéré est joint à cette exécution de la fédération.	Le fédéré n'est plus membre de la fédération, objets propriétés du fédérés supprimés, tous les attributs d'attribut d'instance en attente d'acquisition pour le fédéré sont annulées.
Register Federation Synchronization Point	Label du point de synchronisation, étiquette des informations associées au point, Liste des fédérés concernés (si vide alors tous les fédérés par défaut).	Néant	Le fédéré est connecté à l'RTI, l'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, liste des fédérés valide, si spécifiée, l'étiquette de point de synchronisation fournie n'est pas utilisée avant, Fédéré non en cours de sauvegarde ou 'restauration.	Point de synchronisation connue par le RTI

Suite sur page suivante ...

TABLE 4.1 – *Suite...*

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Initiate Federate Save	Federation save label, Optional timestamp.	Néant	L'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, une sauvegarde du fédéré a été planifiée par un service ≪ Request Federation Save ≫, le fédéré n'est pas en état de sauvegarde/'restauration	Le fédéré associé est instruit pour commencer une sauvegarde d'état
Federation Saved	Indicateur de succès de la sauvegarde, cause d'erreur (option).	Néant	L'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, le fédéré n'est pas en état de 'restauration	Fédéré informé du succès ou échec de la tentative de sauvegarde, le fédéré peut relancer avec de nouveaux paramètres
Abort Federation Save	Néant	Néant	Fédéré connecté au RTI, l'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, fédéré dans l'état sauvegarde en cours	Abandon de la sauvegarde d'état du fédéré.
Request Federation Restore	Label de sauvegarde du fédéré	Néant	Fédéré connecté au RTI, l'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, fédéré possède une sauvegarde avec un label, fédéré n'est pas dans l'état sauvegarde/'restauration en cours.	Le RTI est notifié de la demande de 'restauration d'état du fédéré
Federation Restored	Indicateur de succès de la 'restauration, cause d'erreur (option).	Néant	L'instance de la fédération existe, le fédéré est associé à l'instance de la fédération, fédéré possède une sauvegarde avec un label, fédéré n'est pas dans l'état sauvegarde/'restauration en cours	Le fédéré est informé du succès ou l'échec de la tentative de 'restauration, le fédéré peut éventuellement relancer l'opération de 'restauration avec de nouveaux paramètres

Gestion des déclarations

Les fédérés associés aux instances de fédérations doivent utiliser les services des déclarations (DM) pour déclarer leur intention d'émettre et/ou de recevoir des informations. Les services de déclaration doivent être invoqués avant de pouvoir :

1. Enregistrer des instances d'objet, mettre à jour la valeur d'un attribut d'instances et envoyer des interactions.
2. Découvrir des objets.
3. Utiliser les services de gestion d'objets et de propriété.

Les objets à déterminer par le service de déclaration (avec les autres services de la HLA) peuvent être répartis-en :

— Classes d'objets dans lesquelles des instances d'objets peuvent être enregistrées
— Classes d'objets dans lesquelles les instances d'objets sont découvertes
— Attributs d'instance disponibles pour être mis à jour et reflétés
— Interactions pouvant être envoyées
— Classes d'interaction auxquelles les interactions sont reçues
— Paramètres disponibles pour l'envoi et la réception

Les effets des services DM sont indépendants de l'heure logique de tout fédéré affilié à l'instance de la fédération.

TABLE 4.2: Exemple de services HLA : gestion des déclarations.

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Publish Object Class Attributes	Id de la classe, ensemble d'identificateurs des attributs	Néant	Le fédéré est connecté au RTI, l'instance de la fédération existe, le fédéré est associé, l'Object class est dans le FDD-FOM, Les attributs sont valables pour la classe, le fédéré n'est pas en état de sauvegarde ou de 'restauration	Les attributs de la classe sont publiés pour le fédéré, si un attribut au moins est publié pour le fédéré, l'ID de l'objet est publié pour le fédéré, si un attribut au moins est publié pour le fédéré, le privilège HLAprivilegeToDeleteObjectclass attribute est publié implicitement pour ce fédéré, le fédéré s'approprie des attributs publiés.

Suite sur page suivante ...

TABLE 4.2 – *Suite...*

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Unpublish Object Class Attributes	Id de la classe, Ensemble d'identificateurs des attributs (optionnel)	Néant	Fédéré connecté, la fédération existe, fédéré associé, classe défini dans FDD, si l'ensemble d'identificateurs d'attributs est spécifié ils doivent être valides, les attributs ne doivent pas être appropriés par d'autres fédérés, fédéré n'est pas en état de sauvegarde ou 'restauration.	Si l'ensemble d'attributs est spécifié, ces attributs ne sont plus publiés par le fédéré, si un privilège de suppression est publié pour le fédéré sur des attributs de la classe en question il est implicitement non publié pour le même fédéré, si l'ensemble d'attributs optionnel n'est pas spécifié alors tous les attributs de la classe seront non publiés, le fédéré n'est plus propriétaire des attributs dans la fédération.
Publish Interaction Class	Id de classe d'interaction	Néant	Fédéré connecté, instance de fédération existe, fédéré associé à la fédération, classe d'interaction définie dans FDD, fédéré n'est pas en état de sauvegarde ou 'restauration	Le fédéré peut envoyer les interactions de la classe publiée
Unpublish Interaction Class service	Id de classe d'interaction	Néant.	Fédéré connecté, instance de fédération existe, fédéré associé à la fédération, classe d'interaction définie dans FDD, fédéré n'est pas en état de sauvegarde ou 'restauration.	Le fédéré n'envoi plus les interactions de la classe un-publiée
Subscribe Object Class Attributes service.	Id de la classe d'objets, ensemble d'identificateurs des attributs, indicateur de souscription passive(option), cadence de mise -à-jour(option).	Néant	Le fédéré est connecté au RTI, l'instance de la fédération existe, le fédéré est associé, l'Object class est dans le FDD- FOM, le fédéré n'est pas en état de sauvegarde ou de Restauration	RTI notifié de la demande de souscription du fédéré à la classe d'objet.

Suite sur page suivante ...

TABLE 4.2 – *Suite...*

Service	**Paramètres requis**	**Argument-retour**	**Pré-Conditions**	**Post Conditions**
Unsubscribe Object Class Attributes service	Id de la classe d'objets, ensemble d'identificateurs des attributs.	Néant	Fédéré connecté, la fédération existe, fédéré associé, class défini dans FDD, si l'ensemble d'identificateurs d'attributs est spécifié ils doivent être valides, fédéré n'est pas en état de sauvegarde ou restauration.	Si l'ensemble d'attributs est spécifié, ces attributs ne sont plus souscrits par le fédéré, si l'ensemble d'attributs optionnel n'est pas spécifié alors tous les attributs de la classe seront non souscrits. Le fédéré ne reçoit plus les service de découverte et de réfléxion pour cette classe si tous les attributs sont non souscrits.
Subscribe Interaction class service	Id de classe d'interaction, indicateur de souscription passive(optionnel)	Néant	Fédéré connecté, instance de fédération existe, fédéré associé à la fédération, classe d'interaction définie dans FDD, fédéré n'est pas en état de sauvegarde ou restauration.	Notification RTI de la demande de souscription du fédéré.
Unsubscribe Interaction Class service	Id de classe d'interaction	Néant	fédéré connecté, instance de fédération existe, fédéré associé à la fédération, classe d'interaction définie dans FDD, fédéré n'est pas en état de sauvegarde ou restauration.	Le RTI ne fournie plus d'interaction spécifiée au fédéré.

Il est à noter qu'après une invocation aux services RTI de souscription (classe d'objets et d'interactions), le fédéré peut contrôler la réception effective des informations grâce à des routines d'activation/désactivation.

Gestion du temps

Cette classe de services HLA vise à implémenter une gestion temporelle de l'exécution de la simulation distribuée dans HLA, quoique HLA soit une norme de type temps réel (les évènements sont traités à leurs temps d'occurrence), la norme prévoit d'une manière facultative une gestion causale à la demande des simulations fédérées connectées au RTI et membre de fédération.
Pour cela, HLA différencie deux types de messages : TSO (timestamped order) et RO (received-order). Les premiers obéissent à la règle de causalité tandis que les deuxième sont reçus à n'importe quel ordre (temps réel). Seul les fédérés ayant demandé préalablement une gestion de régulation de temps HLA peuvent émettre des messages TSO, de la même manière, seuls les fédérés ayant demandé préalablement à contraindre les messages par l'horodatage peuvent recevoir des messages TSO.

HLA prévoie un ensemble de routines permettant de demander l'activation de la gestion (régulation et contrainte) du temps logique HLA, sa désactivation et les routines concernant la gestion temporelle telle que l'avancement de l'heure logique pour un fédéré, la rétraction d'un message, etc. La liste non exhaustive suivante énumère quelques routines de temps :

— EnableTimeRegulation, TimeRegulationEnabled et DisableTimeRegulation,
— EnableTimeConstrained, TimeConstrainedEnabled et DisableTimeConstrained,
— TimeAdvanceRequest et TiemAdvanceRequestAvailable,
— NextMessageRequest et NextMessageRequestAvailable,
— FlushQueueRequest,
— RetractService,
— QueryLogicalTime, etc.

Autres Fonctionnalités

HLA prévoie d'autres routines supplémentaires telles que la gestion de la propriété des objets qui est utilisée par les fédérés associés à une fédération et le RTI pour transférer la propriété des attributs d'instance entre ces derniers. La possibilité de transférer la propriété des attributs d'instance entre les fédérés joints doit être requise pour prendre en charge la modélisation coopérative d'une instance d'objet donnée dans une fédération. Seulement le fédéré associé qui possède un attribut d'instance pourra :

— invoquer le service « UpdateAttributeValues » pour fournir une nouvelle valeur pour cet attribut d'instance
— Recevoir des appels du service « ProvideAttributeValueUpdate » pour cet attribut d'instance
— Recevoir des invocations des options : « TurnUpdatesOn » et « TurnUpdatesOff » relatif à un attribut d'objet d'instance.

Une autre fonctionnalité est la gestion de la distribution grâce à la définition des concepts de dimension, de rang et de région (ensemble d'attributs et de classes agrégée ensemble). Ces concepts limitent la portée de toutes les routines de gestion des objets (publication et souscription) ce qui permet de réduire le temps d'échange d'informations dans HLA en se passant des échanges impertinents.
Toutes les interfaces de la HLA vues précédemment peuvent être documentées d'une manière détaillée dans [IEEE, 2010a] quant à la spécification de l'OMT et particulièrement ses deux dérivés ; leFOM et le SOM est bien illustrée dans [IEEE, 2010c].

4.5.2 Travaux d'extension sur la HLA

Les auteurs dans [Zacharewicz *et al.*, 2005] tentent d'implémenter une solution conservatrice au problème de causalité et de gestion des horodatages dans HLA en s'appuyant sur les routines de gestion de temps vu précédemment. Une proposition d'une nouvelle méthode qui relie les pratiques méthodologiques de la recherche opérationnelle et de la gestion des connaissances (OR/MS) et de la simulation distribuée est relatée dans [Anagnostou et Taylor, 2017]. A travers une étude de cas dans un service médicale, les auteurs ont montré que leur cadre méthodologique simplifie significativement l'implémentation de la SD. Le travail présenté dans [Falcone *et al.*, 2017] tente d'étendre le framework HLA par un mécanisme de réactivité baptisé RxHLA qui est un ensemble de services indépendants de HLA et qui visent à offrir un cadre réactif pour la simulation distribuée basé sur le modèle de conception Observateur. Les auteurs introduisent la classe template Observable¡DataPacket¿ permettant de gérer les communications entre les différentes fédérations HLA sans faire recours au API HLA (RTI) en permettant de s'abonner à des données générées par des fédérations dans HLA. RxHLA est toujours indépendant et son intégration dans HLA reste comme perspective. La figure 4.8 illustre le schéma de communication dans RxHLA tandis que le diagramme de classe est donné par la Figure 4.9.

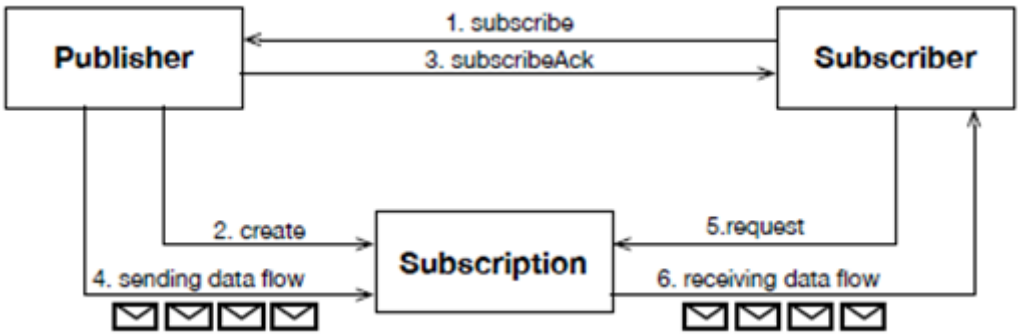

FIGURE 4.8 – Mécanisme de communication Réactive RxHLA [Falcone *et al.*, 2017].

L'utilisation du modèle maître/esclave Master-Worker (M-W) dans l'exécution d'une simulation à évènements discrets parallèle Parallel Discret Event Simulation (PDES) est discuté dans [Park et Fujimoto, 2009]. Afin de réduire la latence et le surcout induit par la distribution d'une simulation sur plusieurs machines (Workers). Les auteurs proposent quatre améliorations à savoir :

— Mise en cache des jobs de simulation qui permet de réduire les états à transférer entre un job sortant et un autre entrant au niveau d'un Worker, disposant d'un vecteur d'état valide. Une maintenance de la cohérence du cache est indispensable
— Mise à jour anticipée du vecteur d'état de simulation au niveau des Workers pour éviter les erreurs irrécupérables en cas de pannes,

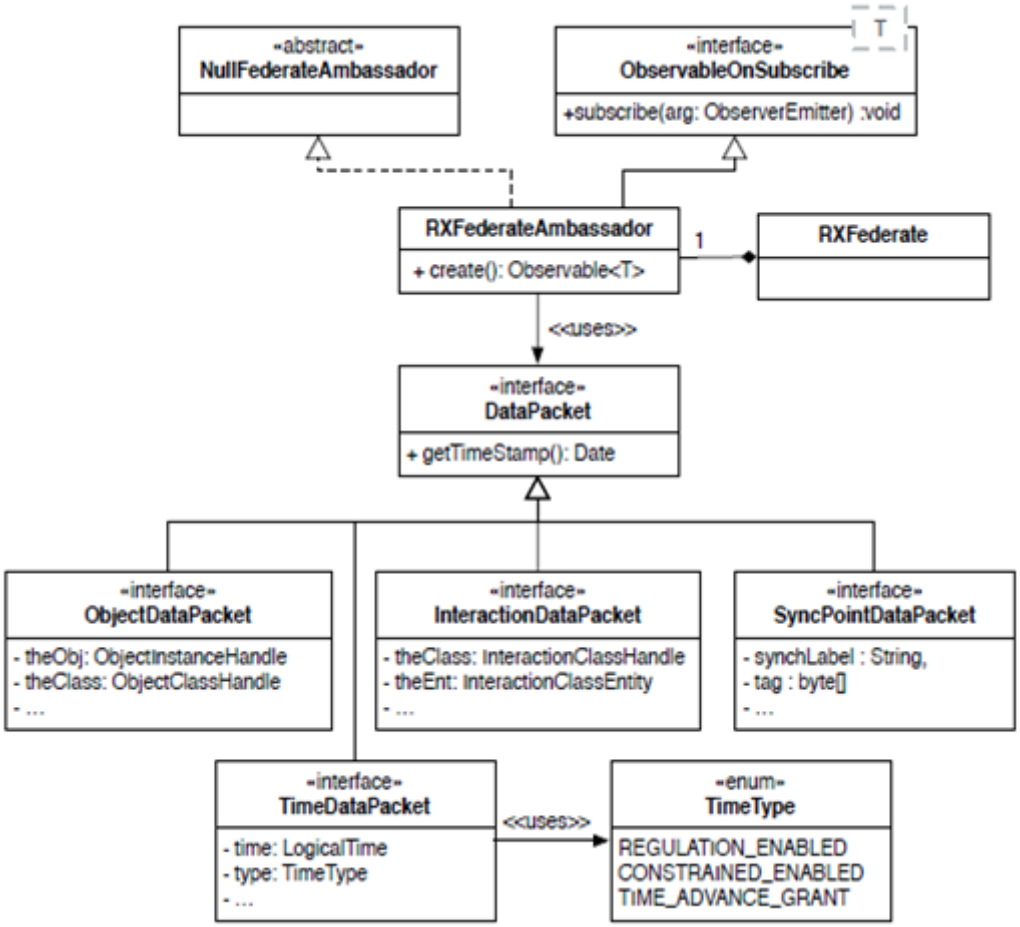

FIGURE 4.9 – Diagramme de classes de RxHLA [Falcone *et al.*, 2017].

— Livraison accélérée des messages selon une approche pro-active.
— Assignation des jobs aux Workers basée sur la disposition de ces derniers d'un vecteur d'état valide.

Dans [Durak *et al.*, 2020], les recommandations du standards de la simulation distribuée HLA sont appliquées au domaine de simulation de vol, des mises au points concernant l'implémentation des fédérés représentant des aéronefs en simulation sont avantageusement enrichies.

4.6 Conclusion

Dans ce chapitre, après l'introduction des concepts principaux de la simulation, on a présenté la simulation distribuée : une discipline à l'intersection de la simulation classique et des systèmes distribués. On a présenté les différentes approches et modes de distribution de simulation. Une attention particulière est donnée à la simulation distribuée temps réel ainsi qu'une étude détaillée du standard HLA est relatée avec quelques travaux de recherche relativement récents sur ce standard de simulation distribuée. Le chapitre

suivant est consacré à la présentation de VolSIM notre environnement d'exécution de simulation basé sur le calcul volontaire étudié dans la première partie de cette thèse.

Chapitre 5

VolSIM : Un environnement de simulation à base du calcul volontaire

5.1 Introdution

Dans ce chapitre une spécification d'un environnement de simulation à base du calcul volontaire nommé VolSIM est donnée. L'idée s'articule sur un serveur de calcul volontaire qui stocke des tâches propres à des simulations pouvant s'exécuter d'une manière distribuée. La gestion des différents aspects de simulation (gestion du temps, interopérabilité, récupération des données, etc.) est complètement assurée d'une manière centralisée par le serveur.

5.2 Exigences et fonctionnalités requises de l'environnement cible

L'environnement de simulation à base du calcul volontaire doit offrir d'une part toutes les fonctionnalités requises par les environnements de calcul volontaire vu à la section 1.5 du chapitre 1 à savoir : (i) Une répartition efficace des tâches aux volontaires, (ii) Robustesse, i.e. détection des volontaires défaillants et reprise des tâches non exécutées, (iii) Confiance au middleware et (iv) la sécurité et d'autre part aux exigences de la simulation distribuée à savoir le respect du principe de causalité et la cohérence.
Les fonctionnalités requises par un environnement de simulation à base du calcul volontaire peuvent être résumsées par :

1. Ordonnancement et répartition des unités de simulation sur les volontaires selon une politique d'ordonnancement efficace.

2. Offrir aux différents processus logiques de simulation s'exécutant sur des machines de volontaires un service de publication/Abonnement d'objets de simulation similaire à celui offert par le standard HLA détaillé précédemment (chapitre 4).

3. Récupération des états des simulations exécutées sur des volontaires suite à la défaillance de ces derniers et pouvoir les réattribuer à d'autres volontaires

4. Offrir aux unités de simulation sur les machines des volontaires un mécanisme de gestion du temps logique

5. Récupération des résultats de simulation (Les machines des volontaires sont utilisées uniquement comme support d'exécution de simulation).

6. Eventuellement des services de synchronisation.

5.3 VolSIM : Un environnement de simulation basé sur le CV

5.3.1 Contraintes de modélisation

Avant de détailler l'architecture de notre environnement de simulation basé sur le CV, nous allons discuter un volet primordial dans notre environnement ; c'est la modélisation préliminaire de la simulation.
Comme on l'a mentionné dans la première partie dédiée au CV (chapitre 1), les projets adéquats pour ce type d'infrastructure de calcul libre-coût sont ceux où il y a possibilité de décomposition en une multitude de tâches minuscules, de préférences indépendantes entre elles, c'est dans ce cas de figure où le calcul volontaire trouve sa justification et sa vraie grandeur. Dans le cas des simulations, même les plus complexes entre elles, il est souvent le cas de constater une dépendance entre les parties simulées, dans le meilleur des cas, ces parties doivent interagir. La situation s'empire si ces parties requièrent une synchronisation. La nature volatile des ressources de calcul dans les systèmes de CV rend l'exécution d'une simulation distribuée avec des contraintes temporelles extrêmement difficile.

5.3.2 Décomposition

Afin d'exécuter une simulation complexe en utilisant un système de CV, il est incontournable de procéder à une décomposition préalable de la simulation. Une simulation individuelle peut être modélisée formellement par l'équation (5.1).

$$M =< X, Y, S, ta, \delta_{ext}, \delta_{int}, \lambda > \tag{5.1}$$

Où :

- X : est l'ensemble des évènements entrants.
- Y : est l'ensemble des évènements sortants.
- S : ensemble d'états séquentielles.
- $ta : S \longrightarrow T^{\infty}$ est la fonction qui détermine la durée de vie d'un état.
- $delta_{ext} : Q \times X \longrightarrow S$ est la fonction de transition externe qui définie l'influence d'un évènement externe sur l'état du système tel que :
  $Q = (s, t_e)|s \in S, t_e \in (T \cap [0, ta(s)])$ est l'ensemble de tous les états, t_e est le temps écoulé depuis le dernier évènement.
- $delta_{ext}$:La fonction de transition interne qui définie la façon avec laquelle le système change intérieurement.

— $\lambda : S \longrightarrow Y^{\emptyset} | Y^{\emptyset} = Y \cup \emptyset$ est la fonction de sortie, $\emptyset \notin Y$ est l'évènement nul (ou l'évènement non observable).

Dans ce formalisme, plusieurs éléments peuvent être pris en considération afin de décomposer une simulation unique. Les mécanismes de décomposition envisageables sont [Ray, 2003], [Righter et Walrand, 1989] :

— Décomposition par parallélisation automatique : cette décomposition consiste à soumettre le code d'une simulation à un compilateur parallèle qui va procéder à une décomposition automatique. Cette méthode est dédiée à des parties de simulation qui s'exécuterons sur une machine parallèle d'une part, d'autre part les résultats sont souvent mitigés ; le recourt à des méthodes semi-automatiques où l'utilisateur indique explicitement au compilateur les parties pouvant être parallélisées est d'usage fréquent. La détection des parties intrinsèquement parallèle d'un grand code de simulation est pratiquement irréalisable par ce mécanisme.
— Distribution des fonctions du simulateur : Les différentes fonctions du simulateur peuvent être affectées à des nœuds de calcul différents, par exemple ; un nœud pour la génération des paramètres d'entrée du simulateur, un deuxième nœud pour la vérification du modèle, un troisième pour la collection et la consolidation des résultats et un quatrième pour la visualisation. Les nœuds d'exécution (machines éventuellement) doivent travailler en coopération et non en concurrence comme dans le premier mécanisme de parallélisation automatique.
— Distribution des évènements : pour ce mécanisme on distingue deux catégories : La distribution centralisée des évènements où un nœud dispose d'une liste globale de tous les évènements liés à la simulation et procède à leur distribution sur les autres nœuds. Ce mécanisme est parfait pour les systèmes à mémoire partagée à cause de la liste globale des évènements. La distribution décentralisée des évènements consiste à attribuer un sous ensemble d'évènements à chaque nœud qui les traite localement, les évènements peuvent engendrer d'autre évènements locaux au nœud ou à envoyer à d'autres nœuds (évènements sortants). Un mécanisme de cohérence doit être mis en place dans ce cas
— Distribution des éléments du modèle : consiste en une décomposition en plusieurs composants du modèle de simulation faiblement couplés (peu d'interactions entre les composants). C'est la méthode de décomposition cartésienne des systèmes complexes et son intérêt réside dans le fait qu'elle exploite le parallélisme inhérent au modèle et les parties qui le composent. La décomposition en éléments du modèle génère un graphe de composants où les nœuds représentent les composants et les arcs représentent les interactions entre composants. Le mode de communication dans ce type de composition est le passage de messages (Une implémentation en MPI est envisageable)

— Décomposition par distribution des expériences : une expérience correspond à une exécution unique d'une simulation. Souvent les systèmes ont besoin de multiples expériences de simulation afin de valider le modèle et les hypothèses de simulation. Une méthode intuitive consiste à exécuter la même simulation (avec les mêmes paramètres initiaux) et d'en prendre la moyenne des variables d'état comme résultats finaux, l'autre possibilité est l'exécution de plusieurs expériences de simulation avec des paramètres initiaux différents c'est le cas de plusieurs scénarios de simulation.

Dans le système qu'on propose VolSIM, l'utilisation des machines des volontaires, ressources de calcul dont la pérennité est très mise en cause, exige qu'il soit utilisé exclusivement, à notre avis, dans la décomposition par distribution des expériences ou scénarios. Rappelons que le modèle admis du calcul volontaire est celui d'un système centralisé où aucune coopération entre volontaires n'est prévue (exception faite pour quelques propositions de calcul volontaire P2P). A cet effet les interactions entre les parties de simulations ne sont pas permises ce qui élimine les mécanismes de parallélisation et de distribution des fonctions du simulateur, des composants de modèles et des évènements.

5.4 Architecture de VolSIM

Par analogie à SVCE, VolSIM consiste en un serveur http/WebSocket tournant sous nodejs abritant une base de données stockant :

— plusieurs simulations écrites en HTML/JavaScript interprétable directement à la volée,

— les paramètres et les données de chaque expérience relative aux simulations précédentes

— les résultats de l'exécution des expériences/scénarios retournés par les volontaires.

Le serveur encapsule aussi un dispatcher dont le rôle est de distribuer les différents scénarios/expériences sur les volontaires connectés en plus d'un Collecteur/Validateur de résultats et d'un module de visualisation des données de simulation.

La figure 5.1 illustre l'architecture et les composants de VolSIM. Au niveau des machines des volontaire, une fois connecté, un module "frontend" est uploadé. Ce module est responsable de mener le reste de l'interaction avec le serveur (réception des simulations et des paramètres ainsi que le transfert des résultats). Ensuite, le code d'une simulation est choisi au niveau du serveur par le dispatcher et envoyé vers la machine du volontaire.

Dans l'étape suivante, les paramètres initiaux des expériences et/ou scénarios propres à la simulation en question sont envoyés successivement et au fur et à mesure que le volontaire exécute avec succès les expériences/scénarios qui lui ont été envoyés. Le processus dure autant que le volontaire est connecté. A la déconnexion du volontaire, l'expérience ou le scénario en cours est toujours indiqué comme inachevé au niveau du serveur. Après un délai fixé en tenant compte de la durée maximale estimée de chaque expérience/scénario,

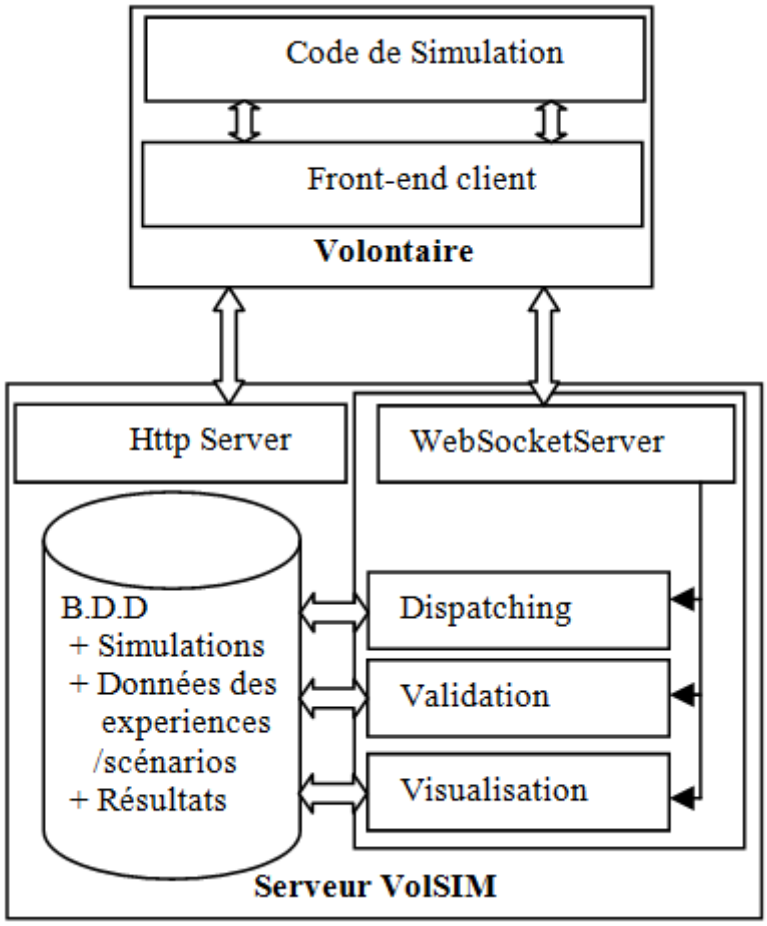

FIGURE 5.1 – Architecture de VolSIM.

le volontaire est déclaré défaillant, et l'expérience/scénario de simulation est réattribué à un autre volontaire de manière identique au fonctionnement des système de calcul volontaire classiques.
La figure 5.2 montre le déroulement d'une session de participation d'un volontaire depuis sa connexion jusqu'à son départ dans l'environnement VolSIM.

5.5 Expérimentation de VolSIM

Afin de tester notre environnement, nous avons choisi une simulation très connu dans le domaine des automates cellulaires (AC) dont les modèles de calcul permettent de décrire la dynamique de systèmes complexes évoluant dans le temps tels que les systèmes physiques (dissipation de la chaleur), biologiques (ingénierie tissulaire, évolution des masses témorales, etc.).

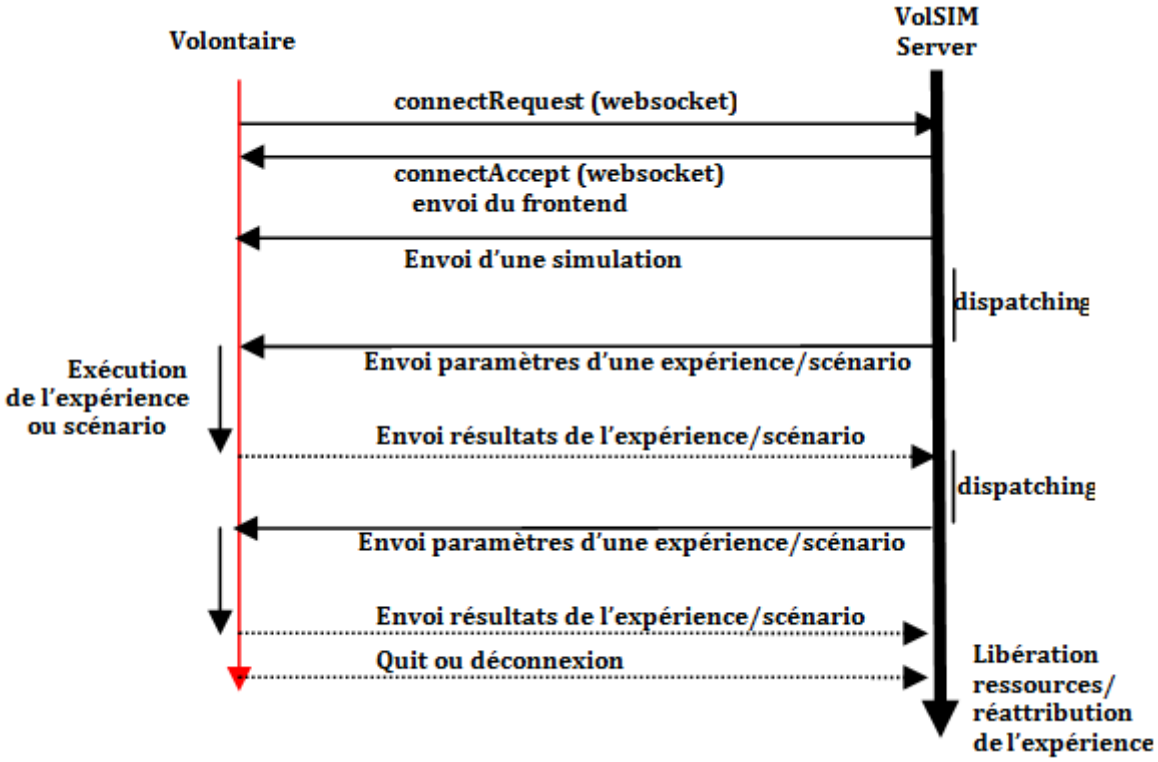

FIGURE 5.2 – Interaction volontaire/serveur dans VolSIM.

5.5.1 Brève aperçu sur les automates cellulaires

Les Automate Cellulaire (AC) sont des systèmes dynamiques, dans lesquels l'espace et le temps sont discrets, constitués d'un certain nombre de cellules identiques dans un réseau régulier de dimension quelconque. Le concept d'AC a été initialement introduit par John Von Neumann et Stan Ulam comme une conceptualisation possible des systèmes biologiques dans le but particulier de modéliser l'auto-reproduction [Ben Youssef, 2015], [Wolfram, 1994]. Chaque cellule de l'espace cellulaire (réseau ou grille) peut être dans un nombre fini d'états. L'état suivant de chaque cellule est déterminé, à des intervalles de temps discrets, en fonction de son état actuel, de l'état actuel de ses cellules voisines et d'une règle ou fonction de transition d'état suivant. Les automates cellulaires fournissent une technique de calcul efficace pour analyser les propriétés collectives d'un réseau de cellules interconnectées.

La théorie des automates cellulaires allant de la formulation mathématique, passant par les propriétés, la dynamique, la convergence, la réversibilité, la décidabilité, etc. a été détaillée dans une multitude d'ouvrages tels que [Formenti *et al.*, 2009], [Kari, 2005], [Luczak et Cohen, 1991].

Afin de mieux cerner le concept de AC d'une manière pratique, on va illustrer la dynamique des automates cellulaires binaires à deux dimensions dans le fameux jeu de la vie introduit par John Conway qui consiste à commencer avec une grille infinie de cases

dans le plan 2D. Chaque case individuelle possède 8 voisins. Une case peut être en « vie » ou « morte » (états binaires). Certaines cases sont en vie à l'initialisation. L'intervalle de temps discret est appelé le temps de génération, la grille évolue selon les deux règles suivantes (d'autres règles peuvent être envisagées) :

— Si une cellule est en « vie » possède 2 ou 3 cellules voisines en « vie » elle demeure en « vie » dans la génération suivante, sinon elle passe à l'état « morte ».

— Si une cellule à l'état « morte» possède exactement trois cellules voisines en « vie », elle prend « vie » à la prochaine génération (naissance).

La figure 5.3 illustre l'évolution d'une configuration simple de cellules en « vie » pour les règles précédentes.

Prenons par exemple la configuration suivante (cellules mortes en blanc et celles en « vie » en noir) :

A noter que la forme de la figure 5.3 se répète, avec une période de deux. Une telle

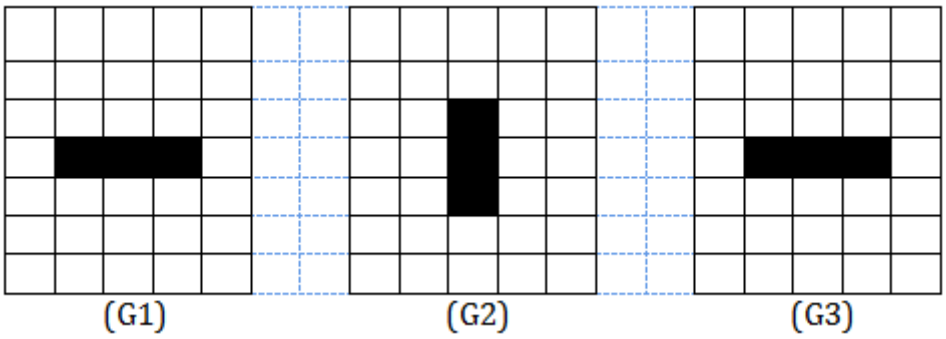

FIGURE 5.3 – Evolution de l'automte cellulaire, avec trois cellules vivantes en ligne au départ (G1).

forme répétitive est appelée un oscillateur. Les formes stationnaires (qui ne changent pas) peuvent être considérées comme des oscillateurs avec une période de 1. Dans la figure 5.4 plusieurs types d'oscillateurs se déplacent à travers la grille à mesure qu'ils changent de génération en génération. Ces formes sont appelées oscillateurs en translation ou planeurs. Un grand nombre de formes ont été découverts et exploités. Ces formes ont été nommées selon leur apparence telle que «bateaux», «ruches» et ainsi de suite.

L'utilisation de moyens de calculs performants a permis de trouver des formes plus complexes comme celle illustrée dans la figure 5.5 où le planeur au sommet a une période de 5, la grande forme à gauche s'appelle un "Wickstretcher" et grandit indéfiniment (représenté après 83 générations) ; la partie en haut de la mèche se déplace vers le haut tandis que la goutte en bas reste en place (avec des turbulences). La «mèche» dans le milieu augmente de longueur et ondule vers le bas. L'utilisation d'automates cellulaires dans la modélisation de divers systèmes, y compris des systèmes biologiques, présente un certain nombre d'avantages, notamment le fait que les AC sont suffisamment simples pour per-

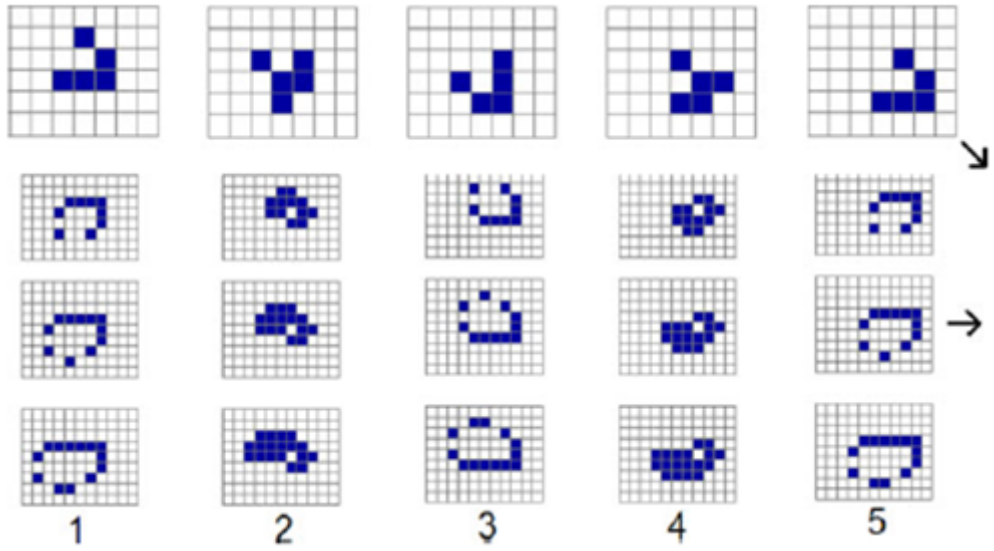

FIGURE 5.4 – Quelques AC de type oscillateur.

mettre une analyse mathématique détaillée, mais suffisamment complexes pour présenter une grande variété de phénomènes complexes. Les modèles basés sur des automates cellulaires fournissent une approche alternative impliquant des coordonnées et des variables discrètes pour représenter un système dynamique complexe. De plus, les algorithmes basés sur les automates cellulaires conviennent également au traitement parallèle [Ben Youssef, 2015], [Cagigas-Muñiz *et al.*, 2020], [Deutsch *et al.*, 2005]. La simulation correspondante à ces modèles nécessite un temps d'exécution exorbitant de nombreux travaux de recherche se sont pencher sur l'utilisation des machines parallèles et des clusters hybrides (multi-cores CPU/GPU) afin de réduire le temps de simulation d'un automate cellulaire. Nous présentons ci-dessous notre approche qui consiste à utiliser le calcul volontaire pour exécuter librement et sans frais supplémentaires une simulation fastidieuse des automates cellulaires via notre environnement VoliSIM.

5.5.2 Spécification de la simulation cible

La simulation cible choisie à exécuter sur VoliSIM consiste en une grille de 128x128 Cellules (cases). Chaque cellule est codée sur un bit (AC binaire). On choisi au départ un nombre de cellules en « Vie » soit N, et on essaye de faire tourner toutes les configurations possibles afin de trouver des structures particulières. Pour cela, On procède comme suit :

Génération des configurations

La génération des configurations est une opération cruciale car ces dernières constituent les données de la simulation à exécuter au niveau des volontaires ; deux options se

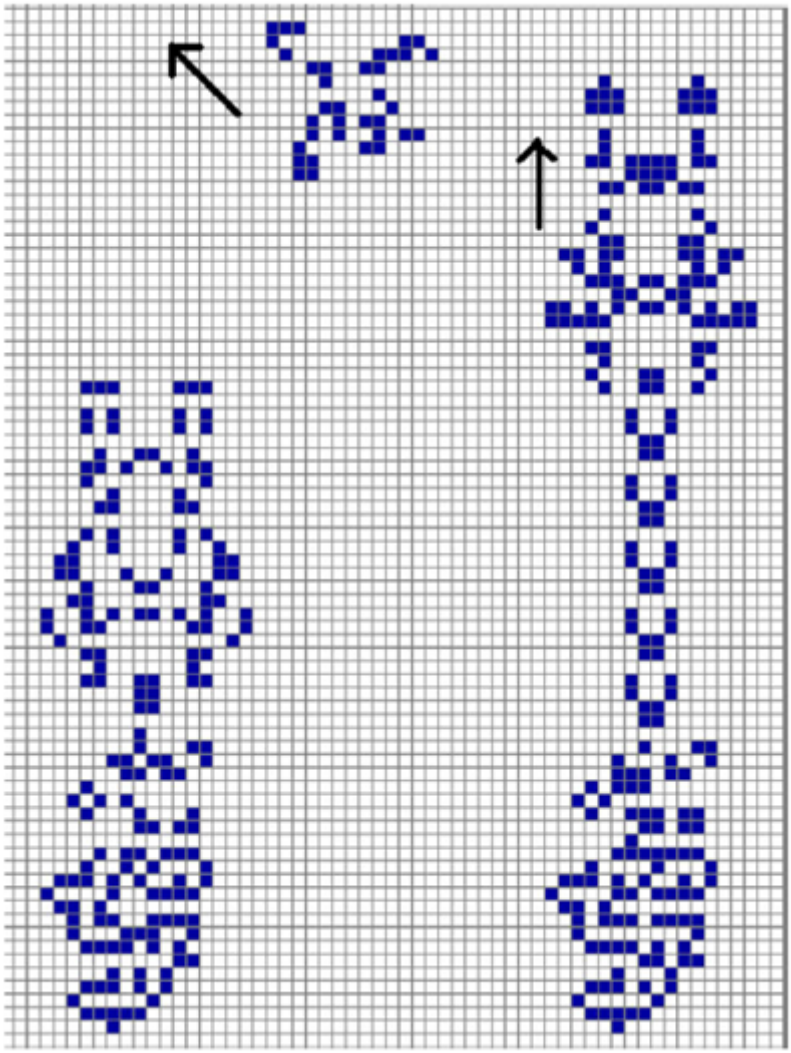

FIGURE 5.5 – Exemple de configuration complexes présentant des propriétés intéréssantes.

présentent : la première est de générer formellement au fur et à mesure toutes les configurations possibles, dans ce cas le nombre de configurations sera de l'ordre donné par l'équation (5.2) :

$$NB_{conf} = C_{128^2}^{N} = \frac{(128^2)!}{N!(128^2 - N)!} \tag{5.2}$$

à titre indicatif, si on prend N=8, le nombre total de configurations initiales possibles est de
NB_{conf} = 126548572519667500019184695328 2 configurations possibles (les mêmes combinaison à des endroits différents sont comptées). L'inconvénient de cette solution est d'une part l'explosion combinatoire qu'elle engendre et d'autre part générer beaucoup de configurations inutiles telles que les configurations où les cellules en "vie" sont éparpillées (la grille se videra à la génération G2).
Nous avons opté pour une solution où :

Algorithm 3 VolSIM-Serv

Require: $R = r_1, r_2, .., r_n$ a set of available resources
SC : Simulation Code
Ensure: Simulation's experience assignation
updating R
for (each new resource r_k in R) **do**
send SC to r_k
R=R+r_k
end for
for (each r_j in R) **do**
generate a set S_e of initial configurations for resource r_j according to their weight.
send s_e to r_j
CALL ReceiveResults(Results r,Status s)
end for
Procedure ReceiveResults(Resource r_i,Result res,Status st)
if (st=error) **then**
R=$R - r_i$
else
check(res) // save the configuration if it is interesting, delete it else.
update(r_i) // update the resource's weight.
end if
End Procedure

— On génère aléatoirement une configuration,
— On test préalablement la configuration générée, s'il s'agit d'une configuration triviale on régénère aléatoirement une autre.

Cette solution, en dépit de son indéterminisme, présente l'avantage de sa simplicité et de sa rapidité. En revanche son inconvénient réside dans le fait qu'on peut générer deux configurations identiques (déjà exécutée par un volontaire). Ce problème, malgré sa faible probabilité d'occurrence (vu que les générateurs aléatoires implémentés sont efficace) peut arriver, il est résolu tout simplement lors de l'enregistrement des résultats ou un contrôle de duplicité des configurations est effectué. Quant au nombre de configurations à générer pour un volontaire donné, l'utilisation de l'algorithme TASP proposé dans [Kadache et Seghir, 2021] vu dans la section (3.4.2) trouve pleinement sa justification en considérant l'exécution d'une configuration comme une tâche élémentaire.

Exécution de la simulation au niveau du volontaire

Une fois le volontaire dispose du code et reçoit des configurations, l'exécution commence immédiatement. Nous avons fixé la terminaison de l'exécution d'une configuration dans l'un des trois cas suivant :

1. Une grille vide (une configuration qui s'auto-détruit)

2. Périodicité : les états intermédiaires sont stockés jusqu'à un certain nombre k. Chaque état résultant du calcul d'une génération de l'automate est comparé avec les k derniers états, si l'état courant concorde avec l'un des k états précédent il s'agit d'un AC périodique, les k états sont renvoyés comme résultats d'exécution.
3. Après un certain temps ; dans ce cas il s'agit d'un AC qui est en perpétuel turbulence (non stationnaire). La tâche est terminée et le résultat des k dernières configurations est retourné.

Stockage des résultats au niveau du serveur

Les résultats retournés sont stockés au niveau du serveur pour éventuel examen ultérieur plus approfondi. L'enregistrement doit comporter la configuration initiale, les K configurations retournées par le volontaire et éventuellement un indicateur sur la terminaison de la tâche (Grille Vide, Périodicité ou hors-temps).

Exigences techniques

La simulation des AC qu'on a choisi d'implémenter sur VolSIM requiert une infrastructure hardware très conséquente particulièrement en matière de stockage. A titre d'exemple, le stockage d'une configuration donnée de l'automate avec ses k configurations résultantes nécessite en moyenne $2 * (k + 1)$ Kilo octet. En vue du nombre astronomique de configurations initiales à examiner, une infrastructure de stockage à part entière doit être envisagée. En plus, plusieurs serveurs doivent être mis en place afin de traiter les demandes des volontaires en un temps acceptable pour ces derniers, en plus d'une connexion onéreuse d'une haute qualité est à déployer. Ces exigences techniques seront destinées au fonctionnement de VolSIM. Leur coût reste négligeable en comparaison avec le coût d'une machine parallèle ou un cluster récent capable d'exécuter une telle simulation. Cette comparaison, en termes de coût, illustre bien l'intérêt d'utilisation du calcul volontaire dans de telles simulations.

5.6 Conclusion

Dans ce chapitre, après discussion des mécanismes de distribution d'une simulation, on a statué sur un mécanisme faisable pour exécuter une simulation dans un système de calcul volontaire qui est celui de la distribution des expériences et/ou scénarios. On a proposé une architecture d'un environnement qu'on a baptisé VolSIM en spécifiant les différentes fonctionnalités dans le serveur VolSIM ainsi que la partie client qui assure l'exécution d'une expérience/scénario au niveau du volontaire.

Conclusion

Synthèse du travail

Dans cette thèse, intitulée « Un environnement d'exécution de simulation basé sur le calcul volontaire », notre travail s'est porté sur les environnements de calcul volontaires et leur éventuelle utilisation dans l'exécution de simulation distribuée. On a subdivisé méthodiquement notre travail en deux parties la première sur les environnements de calcul volontaires (CV) et l'autre sur la simulation distribuée (SD) et la possibilité d'utilisation du concept de CV dans la SD.

Dans la première partie de la thèse, nous avons commencé par un état de l'art concernant les systèmes de calcul volontaire qui ne cessent d'évoluer et d'attirer l'attention de la communauté scientifique vu leur aspect de calcul cost-free n'impliquant pas une consommation aussi excessive d'énergie comme pour les clusters. En effet, la collecte des temps de repos des ordinateurs individuels connectés sur internet fait émerger une puissance de calcul considérable. Les travaux récents dans le domaine mettent le focus sur les différentes problématiques et défis soulevés par ce type de systèmes. Dans un premier temps, nous avons porté nos deux premières contributions en l'occurrence un système de calcul volontaire social et un algorithme d'ordonnancement performant (SVCE et TASP).

Dans la deuxième partie, nous avons introduit la vieille discipline de simulation ainsi que sa descendante directe : La simulation distribuée. Nous avons mis l'accent sur un standard de simulation distribuée qui est le HLA (High Level Architecture) en détaillant les différentes fonctionnalités que doivent satisfaire toute implémentation du standard HLA Dans le but de sa vulgarisation pédagogique pour d'éventuels futurs travaux. Nous avons ensuite proposé notre troisième contribution qui consiste en un environnement baptisé VolSIM qui permet l'exécution d'expériences de simulation en utilisant le concept de calcul volontaire étudié dans la première partie.

La recherche effectuée durant l'élaboration de ce travail nous a permis d'approfondir nos connaissances dans les deux disciplines évoquées (CV et SD) ainsi que dans plusieurs disciplines et sous-disciplines sous-adjacentes à savoir l'ordonnancement, les réseaux sociaux, les systèmes distribués et bien d'autres. Néanmoins, ça nous a permis d'être à la pointe des innovations technologiques dans les deux domaines étudiés. Outre nos contributions résumées ci-après, L'étude menée nous a permis d'effleurer des problématiques récentes

novatrices en matière d'utilisation du calcul volontaire dans la simulation distribuée, ce qui nous ouvrent surement la voie pour de futurs travaux dans ce même contexte.

Contributions

Nos contributions peuvent se résumer en :

— Un environnement de calcul volontaire SVCE utilisant les capacités des réseaux sociaux comme moyen de diffusion et de recrutement des volontaires (ressources de calcul).

— Une nouvelle politique d'ordonnancement qu'on a baptisé TASP et qui est à notre avis optimale dans le sens où elle assigne exactement le nombre de tâches nécessaires à un volontaire en fonction de ses capacités de calcul instantanées.

— Une architecture pour l'environnement VolSIM de simulation qui tire profit des capacités de calcul des volontaires afin d'effectuer des expériences et/ou des scénarios de simulation assez longs et gourmands en terme de ressources de calcul.

— Spécification complète d'une simulation à grande échelle à exécuter par des volontaires. La simulation des automates cellulaires par CV est très prometteuse pour des futurs travaux.

L'objectif escompté de la thèse étant, à notre avis, atteint. Notre travail nous a permis également de mettre en ligne de mire quelques futurs travaux.

Perspectives

Les perspectives de notre travail peuvent être fixées selon les axes suivants :

— Pour l'environnement SVCE, des volontaires spécifiques peuvent être choisis selon des critères qui peuvent être déterminés en interrogeant leurs profils sociaux. Nous pensons que cela améliorera sensiblement le recrutement de nouveaux volontaires. En s'appuyant sur les métriques vues dans le chapitre 2 telle que l'excentricité, nous pensons que le procédé de recrutement des volontaires ciblera des individus à grande potentialité d'influence afin de faire adhérer un maximum de participants.

— Spécialisation des middlewares du calcul volontaire pour les volontaires en possession d'infrastructures de calcul d'envergure (cluster, grille de GPUs, etc). En effet, plusieurs institutions et particulièrement les institutions éducatives (universités, centres de recherche, laboratoires,etc.) manifestent une grande disponibilité à participer aux projets de recherche basés sur le CV en offrant des capacités importantes de calcul. Les algorithmes d'ordonnancement y compris TASP les traitent comme des volontaires individuels. Un ordonnancement local spécifique en adéquation avec ce type de ressources de calcul s'avère plus que nécessaire. Cette mise en adéquation

pour ce type de volontaires peut constituer une voie d'inspection et de recherche très intéressante.

— La mise en œuvre du projet de la simulation des automates cellulaires par le concept du calcul volontaire via VolSIM, le projet est à notre avis très fertile en termes de problématiques et de solutions à envisager. A titre d'exemple, la détermination formelle des configurations "intéressantes" peut éliminer un bon nombre de simulations à exécuter. Une coopération faisant intervenir des spécialistes théoriciens dans les automates cellulaires s'avère très prometteuse.

— Nous pensons que l'intégration directe des API permettant l'utilisation de capacités de bénévoles dans le standard HLA est une autre voie intéressante dans le sens où on réutilise un standard largement adopté en simulation distribuée. La problématique dans ce cas, revient à réadapter les différentes fonctionnalités HLA vues dans le chapitre 4 au contexte du calcul volontaire. La tolérance aux pannes et la synchronisation vont constituer des obstacles de taille pour un tel travail.

Par ce travail, nous estimons avoir fait un trait d'union entre deux disciplines, complètement disjointes auparavant, celle du concept de calcul volontaire et de la simulation distribuée.

Bibliographie

[Anagnostou et Taylor, 2017] Anagnostou, A. et Taylor, S. J. (2017). A distributed simulation methodological framework for OR/MS applications. *Simulation Modelling Practice and Theory*, 70:101–119.

[Anderson, 2004] Anderson, D. P. (2004). BOINC : a system for public-resource computing and storage. *In Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, Pittsburgh, PARIS(66), France.

[Anderson et McLeod, 2007] Anderson, D. P. et McLeod, J. (2007). Local Scheduling for Volunteer Computing. *In 2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, Rome (499),Italy.

[Anderson et Reed, 2009] Anderson, D. P. et Reed, K. (2009). Celebrating diversity in volunteer computing. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*.

[Beaumont *et al.*, 2005] Beaumont, O., Casanova, H., Legrand, A., Robert, Y. et Yang, Y. (2005). Scheduling divisible loads on star and tree networks : results and open problems. *IEEE Transactions on Parallel and Distributed Systems*, 16(3):207–218.

[Ben Youssef, 2015] Ben Youssef, B. (2015). A parallel cellular automata algorithm for the deterministic simulation of 3-D multicellular tissue growth. *Cluster Computing*, 18(4):1561–1579.

[Bethune, 2015] Bethune, I. (2015). PrimeGrid : a Volunteer Computing Platform for Number Theory. *In 4th Annual International Conference on Computational Mathematics, Computational Geometry & Statistics (CMCGS 2015)*.

[Bharadwaj *et al.*, 2003] Bharadwaj, V., Ghose, D. et Robertazzi, T. G. (2003). Divisible load theory : A new paradigm for load scheduling in distributed systems. *Cluster Computing*, 6(1):7–17.

[Boer *et al.*, 2009] Boer, C. A., de Bruin, A. et Verbraeck, A. (2009). A survey on distributed simulation in industry. *Journal of Simulation*, 3(1):3–16.

[Bryant, 1977] Bryant, R. E. (1977). Simulation of packet communication architecture computer systems. Rapport technique, Massachusetts Institute of Technology, Cambridge, MA, USA.

[Cagigas-Muñiz *et al.*, 2020] Cagigas-Muñiz, D., Diaz-Del-Rio, F., López-Torres, M. R., Jiménez-Morales, F. et Guisado, J. L. (2020). Developing efficient discrete simulations on multicore and GPU architectures. *Electronics (Switzerland)*, 9(1):1–17.

[Chandy et Misra, 1979] Chandy, K. et Misra, J. (1979). Distributed Simulation : A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452.

[Chaudron, 2012] Chaudron, J.-b. (2012). *Architecture de simulation distribuée temps réel*. Thèse de doctorat, Université de Toulouse.

[Chetlur *et al.*, 1998] Chetlur, M., Abu-Ghazaleh, N., Radhakrishnan, R. et Wilsey, P. A. (1998). Optimizing communication in Time-Warp simulators. *In Proceedings. 12th Workshop on Parallel and Distributed Simulation PADS '98*, pages 64–71, Banff Alberta, Canada.

[Chorazyk *et al.*, 2017] Chorazyk, P., Godzik, M., Pietak, K., Turek, W., Kisiel-Dorohinicki, M. et Byrski, A. (2017). Lightweight Volunteer Computing Platform using Web Workers. *Procedia Computer Science*, 108:948–957.

[Costa *et al.*, 2012] Costa, F., Silva, J. N., Veiga, L. et Ferreira, P. (2012). Large-scale volunteer computing over the internet. *Journal of Internet Services and Applications*, 3(3):329–346.

[Dahmann *et al.*, 1997] Dahmann, J. S., Fujimoto, R. M. et Weatherly, R. M. (1997). The Department of Defense High Level Architecture. *In Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 142–149, Washington, DC, USA. IEEE Computer Society.

[Damani *et al.*, 1997] Damani, O. P., Yi-Min Wang et Garg, V. K. (1997). Optimistic distributed simulation based on transitive dependency tracking. *In Proceedings 11th Workshop on Parallel and Distributed Simulation*, pages 90–97, Lockenhaus, Austria (28).

[Deutsch *et al.*, 2005] Deutsch, A., Dormann, S. et Others (2005). *Cellular automaton modeling of biological pattern formation*. Springer.

[Dinde et Dixit, 2015] Dinde, S. H. et Dixit, A. M. (2015). On Sharing Infrastructure Resources using Online Social Networks. *Procedia Computer Science"*, 4:948–957.

[Durak *et al.*, 2020] Durak, U., D'Ambrogio, A. et Gerlach, T. (2020). Applying ieee recommended practice for distributed simulation engineering and execution process for modeling and simulation based airborne systems engineering. *In AIAA Scitech 2020 Forum*, Hyatt Regency Orlando, Orlando, Florida,USA.

[Estrada *et al.*, 2009] Estrada, T., Taufer, M. et Anderson, D. P. (2009). Performance Prediction and Analysis of BOINC Projects : An Empirical Study with EmBOINC. *Journal of Grid Computing*, 7(4):537.

[Everett *et al.*, 2018] Everett, M. G., Borgatti, S. P. et Johnson, J. C. (2018). *Analyzing social networks*. Sage, second édition.

[Falcone *et al.*, 2017] Falcone, A., Garro, A., Anagnostou, A. et Taylor, S. J. (2017). An introduction to developing federations with the High Level Architecture (HLA). *In Proceedings - Winter Simulation Conference*, Las Vegas, NV (476), USA.

[Flynn, 1972] Flynn, M. (1972). Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21:948–960.

[Formenti *et al.*, 2009] Formenti, E., Dennunzio, A. et Weiss, M. (2009). 2d cellular automata : expansivity and decidability issues. *CoRR*, abs/0906.0857.

[Fu *et al.*, 2014] Fu, D., Becker, M. et Szczerbicka, H. (2014). Accelerating Distributed Discrete Event Simulation through Exchange of Conditional Look-Ahead. *In 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*, pages 183–189, Toulouse (39),France.

[Fujimoto, 1990] Fujimoto, R. M. (1990). Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53.

[Goossens, 2002] Goossens, B. (2002). *Architecture et micro-architecture des processeurs*. Collection IRIS. Springer Paris.

[Heidelberger, 1986] Heidelberger, P. (1986). Statistical analysis of parallel simulations. *In Proceedings of the 18th conference on Winter simulation - WSC '86*, pages 290–295, New York, New York, USA. ACM Press.

[IEEE, 1993] IEEE (1993). IEEE Standard for Information Technology – Protocols for Distributed Interactive Simulation Applications–Entity Information and Interaction. *IEEE Std 1278-1993*, pages 1–64.

[IEEE, 2010a] IEEE (2010a). IEEE Standard for Modeling and Simulation (M amp ;S) High Level Architecture (HLA)– Federate Interface Specification - Redline. *IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000) - Redline*, pages 1–378.

[IEEE, 2010b] IEEE (2010b). IEEE Standard for Modeling and Simulation (M amp ;S) High Level Architecture (HLA)– Framework and Rules - Redline. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000) - Redline*, pages 1–38.

[IEEE, 2010c] IEEE (2010c). IEEE Standard for Modeling and Simulation (M amp ;S) High Level Architecture (HLA)– Object Model Template (OMT) Specification - Redline. *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000) - Redline*, pages 1–112.

[IEEE, 2011] IEEE (2011). IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP). *IEEE Std 1730-2010 (Revision of IEEE Std 1516.3-2003)*, pages 1–79.

[IETF, 2018] IETF (2018). RFC 6455 - The WebSocket Protocol. `https://tools.ietf.org/html/rfc6455`, accédé le 24/09/2018.

[Jefferson, 1985] Jefferson, D. R. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425.

[Kadache et Seghir, 2021] Kadache, N. et Seghir, R. (2021). A New Social Volunteer Computing Environment with Task-Adapted Scheduling Policy (TASP). *International Journal of Grid and High Performance Computing (IJGHPC)*, 13(2):47–63.

[Kari, 2005] Kari, J. (2005). Theory of cellular automata : A survey. *Theoretical Computer Science*, 334(1-3):3–33.

[Kondo *et al.*, 2007] Kondo, D., Anderson, D. P. et Vii, J. M. (2007). Performance Evaluation of Scheduling Policies for Volunteer Computing. *In Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, pages 415–422, Bangalore (84),India.

[Kumar, 1986] Kumar, D. (1986). Simulating Feedforward Systems Using a Network of Processors. *In Proceedings of the 19th Annual Symposium on Simulation*, ANSS '86, pages 127–144, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Lamport, 1978] Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM*, 21(7):558–565.

[Lee *et al.*, 2010] Lee, Y. C., Zomaya, A. Y. et Siegel, H. J. (2010). Robust task scheduling for volunteer computing systems. *Journal of Supercomputing*, 53(1):163–181.

[Lendermann *et al.*, 2007] Lendermann, P., Heinicke, M. U., McGinnis, L. F., McLean, C., Strassburger, S. et Taylor, S. J. E. (2007). Panel : distributed simulation in industry - a real-world necessity or ivory tower fancy ? *In 2007 Winter Simulation Conference*, pages 1053–1062, Washington, DC (335), USA.

[Li et Franzinelli, 2016] Li, W. et Franzinelli, E. (2016). Decentralizing Volunteer Computing Coordination. *In Social Computing*, pages 299–313, Singapore. Springer Singapore.

[Luczak et Cohen, 1991] Luczak, T. et Cohen, J. E. (1991). Stability of vertices in random boolean cellular automata. *Random Structures & Algorithms*, 2(3):327–334.

[MCMAHON et MILENKOVIC, 2011] MCMAHON, A. et MILENKOVIC, V. (2011). Social Volunteer Computing. *Systemics, Cybernetics and Informatics*, 9(4):34–38.

[Mersenne Research, 1996] Mersenne Research, I. (1996). GIMPS History - PrimeNet. `https://www.mersenne.org/various/history.php`, accédé le 05/01/2019.

[Misra, 1986] Misra, J. (1986). Distributed discrete-event simulation. *ACM Computing Surveys*, 18(1):39–65.

[Moreira *et al.*, 2005] Moreira, E. M., Helena, R., Santana, C. et Santana, M. J. (2005). Using consistent global checkpoints to synchronize processes in distributed simulation. *In Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 43–50, Montreal, Que. (45), Canada.

[Morrison *et al.*, 2001] Morrison, J. P., Kennedy, J. J. et Power, D. A. (2001). WebCom : a Web based volunteer computer. *Journal of Supercomputing*, 18(1):47–61.

[Mustafee *et al.*, 2012] Mustafee, N., Taylor, S., Katsaliaki, K., Dwivedi, Y. et Williams, M. (2012). Motivations and barriers in using distributed supply chain simulation. *International Transactions in Operational Research*, 19(5):733–751.

[Newcomb, 1961] Newcomb, T. M. (1961). *The acquaintance process.* Holt, Rinehart and Winston, New York.

[Ngo *et al.*, 2008] Ngo, S. H., Fukushi, M., Jiang, X. et Horiguchi, S. (2008). Efficient scheduling schemes for sabotage-tolerance in volunteer computing systems. *In 22nd International Conference on Advanced Information Networking and Applications (AINA 2008)*, pages 652–658, GinoWan,Okinawa,Japan.

[Nouman Durrani et Shamsi, 2014] Nouman Durrani, M. et Shamsi, J. A. (2014). Volunteer computing : requirements, challenges, and solutions. *Journal of Network and Computer Applications*, 39(1):369–380.

[Park et Fujimoto, 2009] Park, A. et Fujimoto, R. M. (2009). Efficient Master/Worker Parallel Discrete Event Simulation. *In Proceedings - Workshop on Principles of Advanced and Distributed Simulation, PADS*, pages 145–152, Lake Placid, NY (29),USA.

[Ray, 2003] Ray, C. (2003). *ATLAS, une plate-forme pour la modélisation et la simulation de systèmes désagrégés.* Thèse de doctorat, Université de Resnnes 1.

[Ries, 2013] Ries, C. B. (2013). *UML for BOINC : A Modelling Language Approach for the Development of Distributed Applications based on the Berkeley Open Infrastructure for Network Computing.* Thèse de doctorat, Glyndwr University of Wales.

[Righter et Walrand, 1989] Righter, R. et Walrand, J. C. (1989). Distributed Simulation of Discrete Event Systems. *Proceedings of the IEEE*, 77(1):99–113.

[Robinson *et al.*, 2004] Robinson, S., Nance, R. E., Paul, R. J., Pidd, M. et Taylor, S. J. E. (2004). Simulation model reuse : definitions, benefits and obstacles. *Simulation Modelling Practice and Theory*, 12(7):479–494.

[Rönngren et Ayani, 1994] Rönngren, R. et Ayani, R. (1994). Adaptive Checkpointing in Time Warp. *SIGSIM Simul. Dig.*, 24(1):110–117.

[Sarmenta, 2001] Sarmenta, L. F. G. (2001). *Volunteer Computing.* Phd thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

[Sarmenta *et al.*, 1998] Sarmenta, L. F. G., Hirano, S. et Ward, S. A. (1998). Towards Bayanihan : Building an extensible framework for volunteer computing using Java. *Concurrency - Practice and Experience*, 10(11-13):1015–1019.

[Tabassum *et al.*, 2018] Tabassum, S., Pereira, F. S., Fernandes, S. et Gama, J. (2018). Social network analysis : An overview. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 8(5):1–21.

[Tanenbaum et van Steen, 2006] Tanenbaum, A. S. et van Steen, M. (2006). *Distributed Systems : Principles and Paradigms (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Taylor, 2019] Taylor, S. J. (2019). Distributed simulation : state-of-the-art and potential for operational research. *European Journal of Operational Research*, 273(1):1–19.

[Taylor *et al.*, 2012] Taylor, S. J. E., Turner, S. J., Strassburger, S. et Mustafee, N. (2012). Bridging the Gap : A Standards-based Approach to OR/MS Distributed Simulation. *ACM Trans. Model. Comput. Simul.*, 22(4):18 :1–18 :23.

[Toth et Finkel, 2009] Toth, D. et Finkel, D. (2009). Improving the Productivity of Volunteer Computing by Using the Most Effective Task Retrieval Policies. *Journal of Grid Computing*, 7(4):519.

[Venu et Joe, 2014] Venu, M. et Joe, I. (2014). Improving performance of optimistic simulation for distributed simulation system using speculative computation. *In 2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 428–432, Busan (297),South Korea.

[Wolfram, 1994] Wolfram, S. (1994). *Cellular automata and complexity : collected papers*. Addison-Wesley Pub. Co., Reading, Mass.

[Zacharewicz *et al.*, 2005] Zacharewicz, G., Giambiasi, N. et Frydman, C. (2005). GDEVS/HLA Environment : A Time Management Improvement. *In The 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, pages T4–I–116–0844, Paris, France.

[Zhu *et al.*, 2014] Zhu, Y., Li, D., Xu, W., Wu, W., Fan, L. et Willson, J. (2014). Mutual-Relationship-Based Community Partitioning for Social Networks. *IEEE Transactions on Emerging Topics in Computing*, 2(4):436–447.