Étude comparative des plateformes parallèles pour systèmes multi-agents

Alban Rousset, Bénédicte Herrmann, Christophe Lang

ComPAS'2014 : Parallélisme / Architecture / Système
Neuchâtel, Suisse, du 23 au 25 avril 2014

Étude comparative des plateformes parallèles pour systèmes multi-agents

Bénédicte HERRMANN, Christophe LANG, Alban ROUSSET

Institue Femto-ST - 16 Route de Gray,
Université de Franche-Comté,
25030 Besançon cedex - France
alban.rousset@femto-st.fr

Résumé
La simulation est devenue un outil indispensable à la recherche pour explorer les systèmes sans avoir recours à l'expérience. En fonction des caractéristiques du système la méthode de modélisation utilisée pour représenter le système varie. Les systèmes multi-agents sont ainsi souvent utilisés pour modéliser et simuler les systèmes complexes. Quel que soit le type de modélisation utilisée, l'augmentation de la taille et de la précision du modèle fait croître le nombre des calculs, rendant nécessaire l'utilisation de systèmes parallèles. Dans cet article, nous nous intéressons aux plateformes de simulation multi-agent parallèles. Notre contribution est une étude comparative de ces différentes plateformes, dans un contexte de calcul intensif. Nous présentons une analyse qualitative, à partir de critères que nous avons définis, puis un comparatif de performance, sur la base d'un modèle agent que nous avons implémenté sur chaque plateforme.

Mots-clés : parallélisme, simulation multi-agent, plateforme multi-agent, comparaison de performances

---

1. Introduction

Dans le domaine de la simulation, nous cherchons souvent à repousser les limites, c'est-à-dire à analyser des modèles plus grands et plus précis pour se rapprocher de la réalité d'un problème. De ce fait, la taille croissante des modèles a un impact direct sur la quantité de calcul et les ressources des systèmes centralisés ne sont pas ou plus suffisantes pour simuler ces modèles. L'utilisation de ressources parallèles permet de s'abstraire des limites de ressources des systèmes centralisés et ainsi d'augmenter la taille des modèles simulés. Dans le monde de la simulation, il y a plusieurs manières de modéliser un système. Par exemple, la modélisation du comportement temporel d'un grand nombre de systèmes physiques repose sur des équations différentielles. La discrétisation du modèle permet de le représenter sous la forme d'un système linéaire et d'utiliser les bibliothèques parallèles existantes pour tirer parti d'un grand nombre de nœuds de calcul. A l'inverse, "un «système complexe» est un système composé d'un grand nombre d'entités hétérogènes, où les interactions entre les entités créent de multiples niveaux de structure et d'organisation qui empêchent un observateur de prévoir un comporte-

ment ou sa rétroaction" [1]. Les systèmes multi-agents sont souvent utilisés pour modéliser des systèmes complexes car ils reposent sur une description algorithmique d'agents qui interagissent et représentent ainsi bien le comportement attendu. Du point de vue de l'augmentation de la taille du système, les systèmes multi-agents sont soumis aux mêmes règles que les autres techniques de modélisation.
Dans cet article, nous nous intéressons aux plateformes multi-agents parallèles qui sont les environnements de programmation et d'exécution des systèmes multi-agents. Ces dernières années, l'intérêt pour les plateformes multi-agents parallèles est allé croissant. Cet engouement provient du fait que les plateformes parallèles offrent davantage de ressources pour exécuter des simulations agents à plus large échelle. Ceci permet d'obtenir des résultats ou des comportements qu'il n'était pas possible de mettre en avant à cause d'un trop petit nombre d'agents (par exemple la simulation d'une ville en mouvement, l'interaction entre des cellules).
Notre contribution est la comparaison de différentes plateformes de simulation multi-agent parallèles d'un point de vue fonctionnel (facilité de développement, prise en charge de la distribution...) et d'un point de vue technique (mesure de performance). Pour réaliser cette comparaison, nous avons effectué un travail de bibliographie puis implémenté et testé notre propre modèle sur toutes les plateformes.
L'article se compose de la manière suivante. Nous situons nos recherches en présentant le contexte des systèmes multi-agents en 2 puis un état de l'art sur les plateformes dans la section 3. Dans la section 4 nous expliquons la démarche suivie pour réaliser cette étude comparative et nous détaillons les critères de classification, d'implémentation, le concept de simulation agents. Cette section est suivie par un descriptif sur toutes les plateformes utilisées dans notre comparatif en section 5 et d'une présentation du modèle utilisé pour l'évaluation de performances en 6. En section 7 nous présentons nos résultats de comparaison, quantitatifs et qualitatifs avant de conclure.

2. Contexte

Le concept d'agent a été l'objet de nombreuses études depuis plusieurs décennies et dans différents domaines. Il est non seulement utilisé dans les systèmes à base de connaissances, la robotique et d'autres domaines de l'intelligence artificielle, mais aussi dans les domaines comme la psychologie [7, 25]. L'une des premières définitions de l'agent est due à Ferber [16] :

> "Un agent est une entité réelle ou virtuelle, évoluant dans un environnement, capable de le percevoir et d'agir dessus, qui peut communiquer avec d'autres agents, qui exhibe un comportement autonome, lequel peut être vu comme la conséquence de ses connaissances, de ses interactions avec d'autres agents et des buts qu'il poursuit".

2.1. Les différents types d'agents

De cette définition, nous pouvons mettre en avant différentes propriétés des agents telles que l'action, la perception et la communication. Parmi les agents, différentes classes d'agents sont généralement identifiées. Par exemple les auteurs Chaib-draa et al. dans [9] distinguent les agents réactifs et cognitifs. Ces deux classes d'agents diffèrent par leurs capacités d'action et de raisonnement. Un agent réactif possède un mécanisme de réaction aux événements, ne prenant en compte ni l'explicitation des buts, ni celle des mécanismes de planification. Les agents cognitifs disposent d'une base de connaissances et sont généralement couplés à un moteur d'in-

1. Définition d'un système complexe dans Complex System Society

férence qui leur permet de prendre des décisions. La séparation entre ces deux classes d'agents n'est pas toujours nette et il existe des classes d'agents, qui mélangent en fonction des besoins le réactif et le cognitif.
Pour permettre à un agent d'atteindre un but, il est parfois nécessaire qu'il se déplace dans son environnement. Un agent ayant la capacité de se déplacer s'appelle un agent mobile, contrairement à un agent fixe qui, quant à lui, se cantonne à une position donnée.

2.2. Les systèmes multi-agents

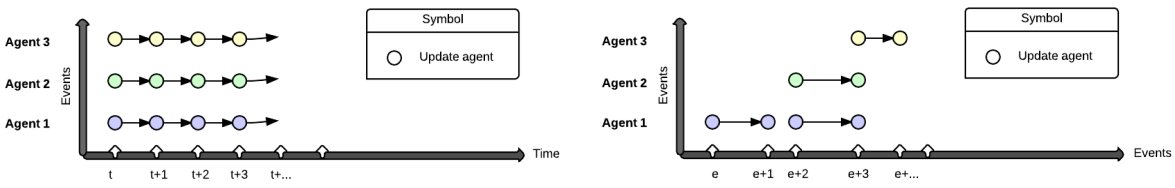
Les agents sont des entités individuelles définies par un comportement plus ou moins limité, souvent un algorithme. Si nous regroupons un ensemble d'agents et que ceux-ci mettent en commun leurs compétences et leurs connaissances en interagissant avec leur environnement ou avec les autres agents, nous obtenons un système multi-agents. Chaib et al. définit dans [9] un système multi-agents comme :

> "Un système distribué qui se compose d'un ensemble d'agents qui interagissent dans et au travers d'un environnement. L'environnement est constitué par les entités en interaction."

Les systèmes multi-agents permettent d'observer des phénomènes ou des comportements au cours du temps par l'intermédiaire de simulations.
A notre connaissance, il n'y a pas de système multi-agents parallèle utilisant des agents cognitifs. Notre étude se concentre donc sur des systèmes multi-agents avec un grand nombre d'agents proches du réactif dont l'objectif est de faire émerger un comportement, par exemple celui d'une ville en mouvement.

2.3. Les simulations multi-agents

Dans la littérature [6, 17], deux types de simulation sont distingués : les simulations dites continues et les simulations dites discrètes (DES). Les simulations multi-agents sont des simulations discrètes. Elles se décomposent en deux grandes classes [17] : les simulations discrètes par pas de temps (time-driven) et les simulations discrètes par événements (event-driven).
Les pas de temps dans une simulation discrète (time-driven) se définissent comme des intervalles de temps réguliers qui sont parcourus au cours de la simulation. Le pas de temps peut prendre différentes unités (secondes, minutes, jours...). Lors de l'incrémentation d'un pas de temps, tous les agents de la simulation sont mis à jour et/ou exécutés. En d'autres termes, nous incrémentons toute la simulation d'une unité de temps comme présenté dans la Figure 1-a. Pour leur part les simulations discrètes par événements (event-driven) utilisent des listes ordonnées d'événements, et traitent donc le premier événement qui est chronologiquement dans la liste. Ceci permet de dérouler la simulation comme présenté dans la Figure 1-b.



(a) pas de temps (b) évènements

FIGURE 1 – Les 2 types de simulations discrètes pour les simulations agents

Le type de simulation est généralement déterminé par le système que nous souhaitons modéliser. Par exemple si nous souhaitons modéliser le comportement de personnes dans une ville, il convient d'utiliser la simulation par pas de temps afin d'observer l'évolution au cours du temps pour chaque agent et pour l'ensemble de la simulation.

3. Les plateformes multi-agents

Il existe des plateformes qui permettent la prise en charge des fonctions de base d'un simulateur multi-agents comme la communication, le cycle de vie des agents, la perception et l'environnement. Parmi les plateformes les plus connus il y a Madkit [19], Mason [22], NetLogo [28] et Gama [1]. Cependant, ces plateformes n'intègrent pas de mécanismes de parallélisme, il est nécessaire de développer une sur-couche à la main, pour distribuer ou paralléliser une simulation. Un autre type de plateformes multi-agents prennent en compte nativement la parallélisation comme RepastHPC [13], D-Mason [14], Pandora [2], FLAME [10], JADE [3].
Il existe quelques articles qui établissent la comparaison de plateformes multi-agents [29, 5, 4, 20]. En revanche durant l'analyse de la littérature, nous n'avons eu connaissance de l'existence d'autres comparatifs des différentes plateformes multi-agents parallèles hormis celui présenté dans l'article de Coakley et al. [10]. Ce comparatif repose uniquement sur des critères qualitatifs tels que le langage d'implémentation mais ne présente pas de comparaison de performances. D'où la problématique qui est la nôtre et que nous allons exposer dans la section suivante.

4. Méthodologie

Nous présentons dans cette partie la méthodologie utilisée pour cette comparaison. Une recherche bibliographique classique (mots clefs dans les moteurs de recherche sur Internet et références d'articles) nous a permis d'établir une liste, que nous espérons complète, des plateformes de simulations multi-agents parallèles existantes. Nous avons ensuite établi des critères de sélection pour ces différentes plateformes et analysé les plateformes au regard de ces critères. Pour finir, nous avons évalué les plateformes en procédant à l'implémentation d'un modèle significatif et à son exécution afin de comparer les performances de chacune des plateformes.
Pour réaliser ce comparatif nous nous sommes intéressés aux aspects qui nous ont parus significatifs dans la mise en œuvre d'une simulation multi-agents parallèle : le développement du modèle et l'exécution efficace de ce dernier. Nous avons donc défini plusieurs critères qui se décomposent en deux grandes catégories : les critères en lien avec le calcul intensif et les critères en lien avec le développement d'un simulateur et son exécution. Nous expliquons brièvement à quoi correspond chaque critère.
Pour le développement d'un modèle, les plateformes imposent des contraintes. Ces contraintes auront un impact sur la programmation (facilité de développement, par exemple) et sur l'implémentation elle-même (modèles d'agent, type). Les critères retenus sont les suivants :

1. Langage de programmation : dans quel langage les simulations sont-elles développées ?
2. Représentation des agents, canevas générique/modèles d'agents
3. Type de simulation : pas de temps (time-driven) ou évènements (event-driven)
4. Reproductibilité des simulations : plusieurs exécutions d'une même simulation donnent elle le même résultat ?

Le choix d'une plateforme passe également par l'évaluation de la qualité de son environnement de développement, de déploiement et sa pérennité. Pour mesurer ces éléments nous avons établi une notation sur les critères suivants :

1. Documentation utilisateur ou développeur
2. Exemples, leur nombre et la couverture
3. Tutoriaux pour comprendre le fonctionnement de la plateforme et le développement d'une simulation
4. Développement en terme de facilité
5. Prise en main, temps passé pour comprendre la plateforme et effectuer le développement
6. Communauté et sa réactivité
7. Compilation, principalement la facilité
8. Exécution dans un environnement de calcul intensif

Puisque nous nous intéressons aux plateformes parallèles, les propriétés classiques suivantes, en lien avec le parallélisme, sont analysées :

1. Extensibilité de la plateforme : en termes d'agents et de nœuds de calcul
2. Répartition de charge
3. Exécution multi-threadée
4. Bibliothèque de communication utilisée

5. Les Plateformes évaluées

Parmi les travaux collectés au cours de la recherche bibliographique, nous avons identifié 10 projets de plateformes multi-agents parallèles. Pour chacune nous avons essayé de télécharger les codes sources ou exécutables, de compiler et de tester les exemples et modèles fournis. Certaines ont été abandonnées soit parce qu'il n'existe pas de code source ou exécutable téléchargeable (MACE3J [18], JAMES[], SWAGES [24]), soit parce que seule une démonstration est fournie et il n'est pas possible de réaliser des développements (PDES-MAS [23, 27]), soit car la plateforme manque de documentation et il n'a pas été possible de compiler le modèle développé (Ecolab [26]). Ces plateformes ont fait l'objet d'une analyse qualitative qui ne fait pas partie de cet article.
Pour les plateformes restantes, au nombre de 5, il a été possible d'implémenter un modèle, c'est-à-dire qu'elles offrent toutes une plateforme multi-agent parallèle fonctionnelle.

5.1. RepastHPC

RepastHPC [13] est développé par le laboratoire national d'Argonne (USA) (`http://repast.sourceforge.net/repast_hpc.html`). Il fait partie de la série de plateformes de simulation multi-agents RepastJ, RepastSimphony. Il est spécialement conçu pour les environnements haute performance. RepastHPC reprend le core de RepastSimphony, c'est-à-dire l'utilisation des projections (grid, network) et des contextes mais en les adaptant à l'environnement parallèle et distribué. Le langage utilisé pour implémenter une simulation agent est le C++ ou le ReLogo, un dérivé du langage Netlogo. Pour la communication, la plateforme RepastHPC utilise MPI par l'intermédiaire de la librairie Boost [12].

5.2. D-Mason

D-Mason (Distributed Mason) [14] est développé par l'université de Salerne (`http://isis.dia.unisa.it/projects/dmason/`). D-Mason est la version distribuée de la plateforme de simulation multi-agent MASON. Les auteurs ont souhaité élaborer la version distribuée de Mason afin de fournir une solution qui n'oblige pas les utilisateurs à réécrire les simulations

qu'ils ont déjà développées pour repousser le nombre maximal d'agents. D-Mason utilise JMS ActiveMQ pour la communication, malgré le fait qu'elle ne soit pas la plus extensible des solutions dans un environnement HPC. D-Mason utilise le langage Java.

5.3. Pandora

Pandora [2] est développé par le groupe de recherche de simulation sociale du Centre Supercomputing Barcelone (`https://github.com/xrubio/pandora`). Il a été explicitement programmé pour permettre l'exécution de simulations multi-agents à grande échelle et il est capable, selon la littérature, de traiter des milliers d'agents avec des actions complexes. Pandora possède un support pour un système d'information géographique, pour faire face à des simulations dans lesquelles les coordonnées spatiales sont pertinentes, à la fois en termes d'interactions de l'agent et de l'environnement. Pandora utilise le langage C + + pour définir et implémenter un modèle de simulation agent. En revanche pour la communication, Pandora génère automatiquement du code MPI.

5.4. Flame

FLAME [10] est développé principalement à l'Université de Sheffield (`http://www.flame.ac.uk`). FLAME a été conçu pour permettre une large gamme de modèles d'agents. FLAME fournit des spécifications sous la forme d'un cadre formel qui peut être utilisé par les développeurs pour créer des modèles et des outils. FLAME permet la parallélisation à l'aide de MPI. L'implémentation d'une simulation FLAME est basée sur la définition de X-machines [11] qui sont définies comme des automates d'états finis avec de la mémoire. Ils peuvent, en plus, recevoir et envoyer des messages à l'entrée et à la sortie de chaque état.

5.5. Jade

JADE [3] est développé par le laboratoire Télécom Italia (`http://jade.tilab.com/`) et vise à simplifier la mise en oeuvre de systèmes multi-agents distribués à travers un middleware qui se conforme aux spécifications de FIPA [3] et à travers un ensemble d'outils qui prennent en charge les phases de débogage et de déploiement. La plateforme peut être distribuée sur plusieurs ordinateurs et la configuration peut être contrôlée à partir d'une interface graphique à distance. La configuration peut même être changée au moment de l'exécution par des agents mobile d'une machine à une autre en cas de besoin. L'implémentation d'un agent se fait par l'intermédiaire du langage Java, tandis que la communication s'effectue à l'aide de RMI.
Ces descriptions succinctes permettent de voir que certaines plateformes ont déjà été évaluées, voire conçu, pour des systèmes parallèles de type cluster alors que d'autres sont plus orientées sur les systèmes distribués, moins fortement couplés, de type réseaux de stations de travail.

6. Définition du modèle

Pour avoir une base de comparaison quantitative et pour exhiber les performances des plateformes, nous avons défini un modèle agent qui a été implémenté sur chaque plateforme. Ce modèle agent est qualifié de "stupid model" car il est simple. Il met cependant en œuvre les différentes propriétés d'un agent : la mobilité, la communication avec son environnement et avec d'autres agents et la perception.
La figure 2 donne une représentation en AML [8] (Agent Modeling Language) de notre modèle. Un modèle AML se compose de différentes entités qui sont un environnement, un agent, des comportements ainsi qu'une ressource. Dans notre modèle, l'environnement (environment) est représenté par une grille ayant une longueur et une largeur. Les agents (Person), sont mo-

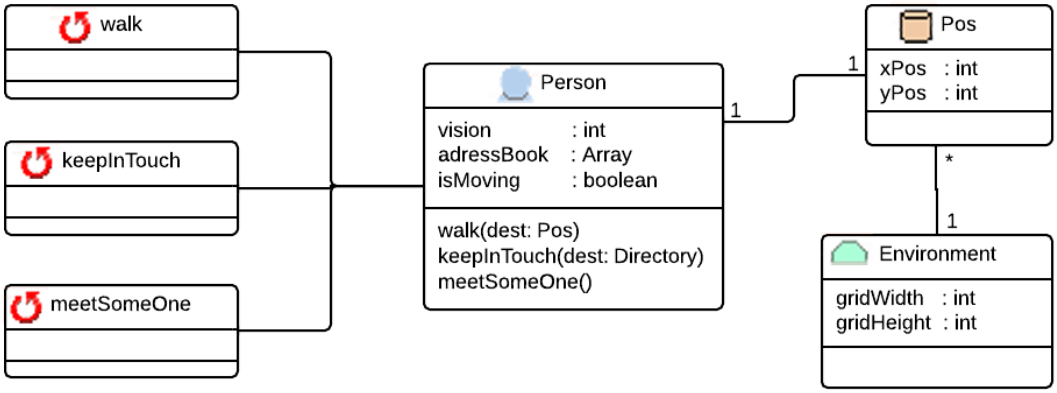


FIGURE 2 – Modèle agent en AML

biles, c'est-à-dire qu'ils ont la capacité de se déplacer aléatoirement sur la grille et possèdent un champ de vision qui leur octroie une perception, limitée, de leur environnement. Chaque agent possède trois comportements *a*) Le comportement Walk permet le déplacement aléatoire d'un agent dans l'environnement. Il y a un seul agent par position dans l'environnement. *b*) Le comportement MeetSomeOne permet à un agent de faire connaissance avec un agent qu'il rencontre dans son champ de vision et de l'ajouter à son carnet d'adresse. *c*) Le comportement KeepInTouch permet de garder le contact avec d'autres agents. Un agent peut aléatoirement, à certains pas de temps, envoyer un message à tous les agents qu'il a rencontrés auparavant. Le comportement d'un agent consiste donc à exécuter chacun de ces trois comportements à chaque pas de temps. Le modèle nous permet, en faisant varier la probabilité de communication, d'avoir soit un modèle communicant, soit un modèle non-communicant.

7. Comparaison des plateformes

Dans cette section nous présentons une comparaison des plateformes au regard des critères présentés dans la section 4, liés au développement des modèles et leur exécution.
Le modèle présenté précédemment a été développé pour toutes les plateformes. La majorité des plateformes utilise des langages classiques tels que C/C++ ou Java pour définir les agents, sauf pour la plateforme Flame qui utilise le langage XMML, langage étendu du XML qui permet de définir une X-Machine. La plateforme RepastHPC, quant à elle, implémente en plus un langage connu dans le monde agent qui est Logo. Le Repast-Logo ou R-Logo est une implémentation du Logo pour C++ qui permet de simplifier l'écriture des simulations mais nous perdons du pouvoir d'expression par rapport au langage C++.
La représentation des agents se ressemble dans la majorité des plateformes où agent sont définis comme des objets comportant des méthodes qui représentent les comportements. Un conteneur d'agents va ensuite regrouper tous les agents. Ce conteneur est découpé et réparti pour l'exécution parallèle. L'implémentation est différente pour la plateforme Flame qui n'utilise pas le concept d'objet pour définir un agent mais utilise des automates (X-machine). Les comportements représentent les états et l'ordre d'exécution des comportements est représenté par les transitions. Cette différence change la logique de programmation mais n'induit pas de limitation par rapport aux autres car les agents sont, au final, codés en C.
Pour le type de simulation (cf. Table 1), toutes les plateformes utilisent le concept de simulation par pas de temps à l'exception de la plateforme RepastHPC. Cette plateforme fonctionne par

événements (event-driven), en revanche l'ordonnanceur de la plateforme permet de donner une périodicité à chaque événement et ainsi d'implanter des simulations par pas de temps.
Pour finir la totalité des plateformes permet aux agents de communiquer que cela soit en interne, avec les agents qui sont sur le même nœud, ou en externe, avec les agents qui sont sur des nœuds différents. Certaines plateformes (D-MASON, Pandora) utilisent le concept de la programmation objet (intéraction entre objets) pour communiquer avec les autres agents, c'est-à-dire que nous pouvons appeler une méthode sur un autre agent. Les autres plateformes utilisent des messages pour communiquer entre les agents.

	RepastHPC	D-Mason	FLAME	Pandora	Jade
Lang. progm	C++/R-Logo	Java	XMML/C	C/C++	Java
Représentation agents	Object	Object [15, 21]	X-Machine[10]	Object	Object
Type simulation	event-driven	time-driven	time-driven	time-driven	time-driven
Reproductibilité	Oui	Oui	Non	Oui [2]	Non

TABLE 1 – Comparatif du développement et de l'exécution de la simulation

Nous donnons dans le tableau 2 une notation basée sur notre expérience acquise par la prise en main et l'implémentation du modèle. Nous y avons regroupé les critères sur la qualité de l'environnement de développement, du déploiement et la pérennité, critères donnés en section 4. La notation est basée sur une suite de (+) pouvant aller de 1(+) pour le moins bon et 4(+) pour le meilleur.

	D-Mason	RepastHPC	Pandora	Flame	Jade
Documentation	++	+++	+++	+++	+++
Exemples	+++	+++	+++	+++	++
Tutoriaux	++	+++	+++	++	++
Développement	+++	+++	+++	+++	+++
Prise en main	+++	++	+++	+++	++
Communauté	+++	+++	+	++	+
Compilation	+++	+	++	+++	++
Exécution	+	+++	+++	+++	+

TABLE 2 – Comparatif développement/déploiement/exécution

En synthèse nous pouvons dire que l'ensemble des cinq plateformes étudiées dans cette section répond bien aux attentes de développement de simulation parallèles. Chacune fournit les outils adaptés pour permettre une exécution sur plusieurs cœurs ou nœuds.

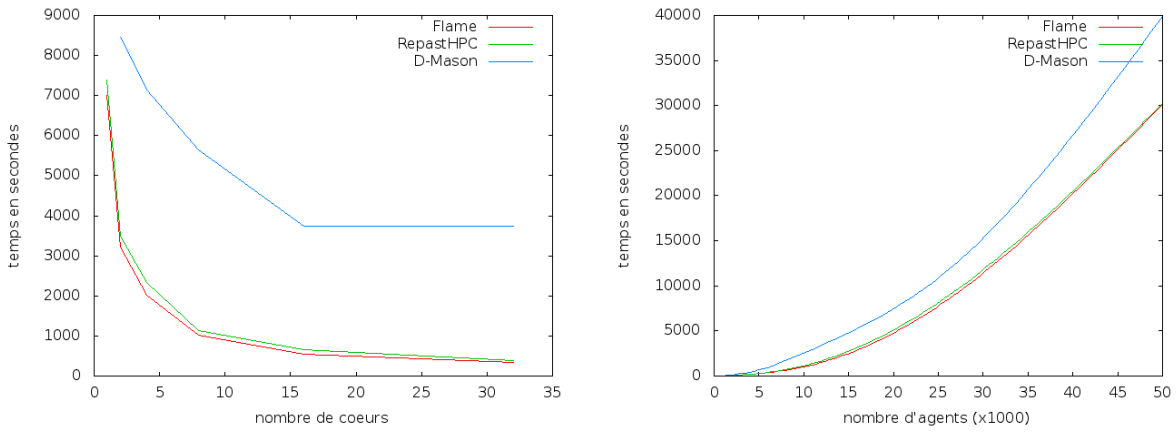
8. Tests d'exécution et analyse de performances

Les tests d'exécution nous ont permis de montrer certaines limites des plateformes. Le critère de l'interface graphique n'a pas été cité dans la Table 2, même si certaines plateformes offrent des interfaces graphiques de qualité. Elles ne sont pas nécessaires dans le contexte du calcul intensif où l'exécution est réalisée sur un cluster en mode batch. Cette interface peut même

	RepastHPC	D-Mason	FLAME	Pandora	Jade
Extensibilité	1028 proc. [21]	36 nodes [10]	432 proc. [10]	NA	NA
Répartition de charges	Dynamique	Dynamique	Statique [10]	Dynamique	Statique [3]
Exécution multi-threadée	Oui [10]	Oui [14, 10]	Non [10]	Oui	Oui
Communication	MPI [12, 13]	JMS [14]	MPI [21]	MPI [2]	RMI

TABLE 3 – Comparatif des critères concernant le HPC

devenir bloquante lorsqu'il n'est pas possible de s'en abstraire pour lancer une simulation et qu'elle bloque l'exécution, c'est le cas de la plateforme D-Mason. La plateforme Jade semble, elle, plutôt conçue pour une surveillance d'équipements distribués et elle ne permet pas de stopper la simulation à un certains pas de temps, ce qui la rend inexploitable ici. Pour finir la plateforme Pandora n'arrive pas à s'exécuter sur notre cluster pour des raisons d'interblocages sur lesquelles nous discutons avec l'équipe. Ainsi seules les plateformes Flame et RepastHPC s'exécutent correctement sur un cluster en mode batch.
Pour l'analyse des performances des plateformes nous avons utilisé les implémentations de notre modèle. Nous avons réalisé plusieurs exécutions pour mettre en avant, d'une part, le comportement des plateformes face à la montée en charge, et d'autre part, face à leur extensibilité : nous faisons varier le nombre de nœuds sur lesquels la plateforme s'exécute en fixant le nombre d'agents (10000 agents) puis nous faisons varier le nombre d'agents pour un nombre de nœuds fixe (8 nœuds) dans la simulation. Chaque exécution pour chaque simulation a été réalisée 10 fois. La variance est faible entre chaque résultat, la moyenne de ces résultats est donc significative.

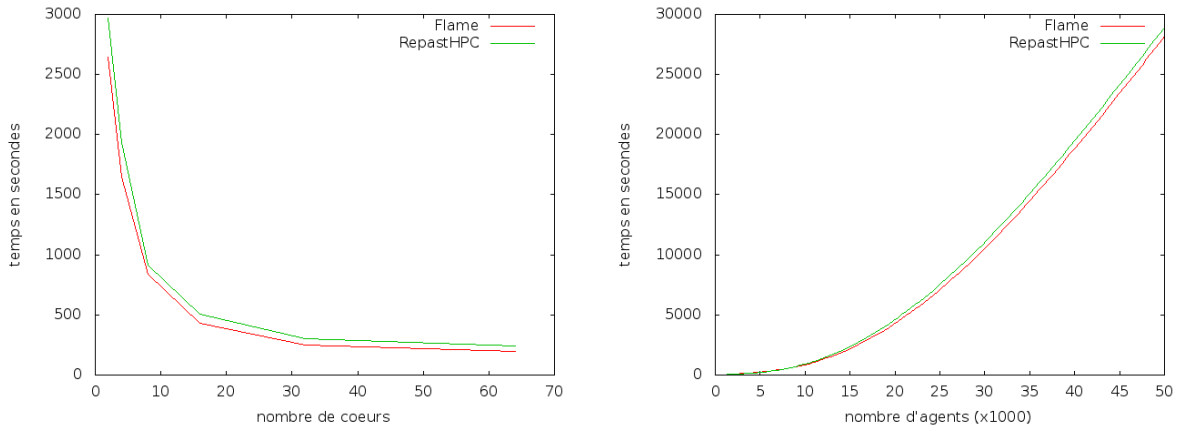


(a) extensibilité (b) monté en charge

FIGURE 3 – Performance des plateformes, machine à mémoire partagée 32 cœurs

Dans une première expérience, nous avons exécuté des simulations sur une machine à mémoire partagée. Cette machine est composée de quatre processeurs Xeon (8*4=32 coeurs) cadencés à

2Ghz avec 64Go de mémoire vive. Son utilisation est interactive, ce qui nous permet d'utiliser la plateforme D-MASON en plus des plateformes Flame et RepastHPC. La Figure 3 présente les résultats de l'expérience pour l'extensibilité (Figure 3(a)) et pour la montée en charge (Figure 3(b)). Les résultats obtenus montrent que la plateforme Flame est la plus extensible ainsi que la moins sensible à la montée en charge. Pour la plateforme RepastHPC, le comportement est proche de Flame avec des temps d'exécution légèrement plus élevés. En revanche pour la plateforme D-Mason, les temps d'exécution sont significativement plus élevés et ne suivent pas la même tendance. Cela s'explique par le fait que cette plateforme est plutôt conçue pour un support distribué (réseau de stations) plutôt que parallèle. Pour finir, cette plateforme supporte moins bien l'extensibilité. Une des causes, mais qui n'est pas encore vérifiée, peut être l'utilisation de JMS pour les communications.



(a) extensibilité (b) monté en charge

FIGURE 4 – Performance des plateformes, cluster en mode batch

Dans une seconde expérience nous avons exécuté la simulation de notre modèle avec les plateformes RepastHPC et Flame sur un cluster en mode batch. Le cluster est constitué de nœuds biprocesseurs, avec des processeurs Xeon E5 (8*2 coeurs) cadencés à 2.2Ghz et 32Go de mémoire vive. Le cluster possède un total de 764 coeurs gérés par le système de batch SGE. Les résultats des exécutions sont donnés par la Figure 4 ; Figure 4(a) pour l'extensibilité des plateformes et Figure 4(b) pour la montée en charge. Nous constatons, malgré une tendance identique, que Flame est plus extensible et supporte davantage la montée en charge que RepastHPC. La différence de performance, entre les deux plateformes varie de 15,6% pour 4 cœurs à 19% pour 64 cœurs. Le speed-up atteint est de 13 pour Flame avec 64 cœurs mais l'amélioration entre 32 et 64 cœurs est peu significative. Les simulations réalisées n'utilisent pas plus de 64 cœurs.
Il faut noter que, d'après ses concepteurs RepastHpc est conçu pour être exécuté sur un très grand nombre de nœuds, des tests de simulation ont ainsi été réalisés jusqu'à 1024 cœurs alors que le plus test de Flame a été exécuté avec 432 cœurs, ce qui peut laisser supposer une meilleure performance sur des plus gros modèles. De même les résultats obtenus dépendent de notre modèle et il est possible que les plateformes se comportent différemment avec d'autres modèles.

9. Conclusion

Dans cet article nous avons présenté un comparatif de différentes plateformes de simulation multi-agents parallèles. Ce comparatif est réalisé à un niveau qualitatif, à l'aide de critères que nous avons choisis, et à un niveau quantitatif, via l'implémentation et l'exécution d'un modèle agent simple. La comparaison qualitative nous permet montrer les propriétés de l'ensemble des plateformes que nous avons étudiées. Pour la partie quantitative, c'est à dire l'exécution des simulations, la plateforme FLAME est plus performante que les autres que cela soit en mode batch ou interactif ou non. Nous pouvons nous poser la question de savoir si cela est du à l'utilisation non pas du concept objet mais des X-Machines, à un nombre de nœuds trop faible pour percevoir le potentiel de RepastHPC ou aux mécanismes de synchronisation sur lesquels repose le parallélisme.
Ainsi, dans nos travaux futurs, nous souhaitons d'une part que toutes les plateformes soient fonctionnelles sur un cluster en mode batch et d'autre part nous souhaitons s'interroger sur l'efficacité des mécanismes de synchronisation dans les plateformes parallèles.

Remerciements : L'ensemble des calculs ont été effectués sur le mésocentre de Calcul de l'université de Franche-Comte.

Bibliographie

1. Amouroux (E.), Chu (T.-Q.), Boucher (A.) et Drogoul (A.). – Gama : an environment for implementing and running spatially explicit multi-agent simulations. *In : Agent computing and multi-agent systems*, pp. 359–371. – Springer, 2009.
2. Angelotti (E. S.), Scalabrin (E. E.) et Ávila (B. C.). – Pandora : a multi-agent system using paraconsistent logic. – In *Computational Intelligence and Multimedia Applications, 2001. ICCIMA 2001. Proceedings. Fourth International Conference on*, pp. 352–356. IEEE, 2001.
3. Bellifemine (F.), Poggi (A.) et Rimassa (G.). – Jade–a fipa-compliant agent framework. – In *Proceedings of PAAM*volume 99, p. 33. London, 1999.
4. Berryman (M.). – *Review of software platforms for agent based models*. – Rapport technique, DTIC Document, 2008.
5. Bordini (R. H.), Braubach (L.), Dastani (M.), El Fallah-Seghrouchni (A.), Gomez-Sanz (J. J.), Leite (J.), O'Hare (G. M.), Pokahr (A.) et Ricci (A.). – A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)*, vol. 30, n1, 2006, pp. 33–44.
6. Braubach (L.), Pokahr (A.), Lamersdorf (W.), Krempels (K.-H.) et Woelk (P.-O.). – A generic time management service for distributed multi-agent systems. *Applied Artificial Intelligence*, vol. 20, n2-4, 2006, pp. 229–249.
7. Carslaw (G.). – *Agent based modelling in social psychology*. – Thèse de PhD, University of Birmingham, 2013.
8. Červenka (R.), Trenčanskỳ (I.), Calisti (M.) et Greenwood (D.). – Aml : Agent modeling language toward industry-grade agent-based modeling. *In : Agent-Oriented Software Engineering V*, pp. 31–46. – Springer, 2005.
9. Chaib-Draa (B.), Jarras (I.) et Moulin (B.). – Systèmes multi-agents : principes généraux et applications. *Edition Hermès*, 2001.
10. Coakley (S.), Gheorghe (M.), Holcombe (M.), Chin (S.), Worth (D.) et Greenough (C.). – Exploitation of high performance computing in the flame agent-based simulation framework. – In *Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and*

Systems, *HPCC '12*, HPCC '12, pp. 538–545, Washington, DC, USA, 2012. IEEE Computer Society.

11. Coakley (S.), Smallwood (R.) et Holcombe (M.). – Using x-machines as a formal basis for describing agents in agent-based modelling. *SIMULATION SERIES*, vol. 38, n2, 2006, p. 33.
12. Collier (N.). – Repast hpc manual, 2010.
13. Collier (N.) et North (M.). – *Repast HPC : A platform for large-scale agentbased modeling*. – Wiley, 2011.
14. Cordasco (G.), Chiara (R.), Mancuso (A.), Mazzeo (D.), Scarano (V.) et Spagnuolo (C.). – A Framework for Distributing Agent-Based Simulations. – In *Euro-Par 2011 : Parallel Processing Workshops*, *Lecture Notes in Computer Science*, volume 7155, pp. 460–470, 2011.
15. Cowling (A. J.), Georgescu (H.), Gheorghe (M.), Holcombe (M.) et Vertan (C.). – Communicating stream x-machines systems are no more than x-machines. *Journal of Universal Computer Science*, vol. 5, n9, 1999, pp. 494–507.
16. Ferber (J.) et Perrot (J.-F.). – *Les systèmes multi-agents : vers une intelligence collective*. – InterEditions Paris, 1995.
17. Fujimoto (R. M.). – Parallel simulation : distributed simulation systems. – In *Proceedings of the 35th conference on Winter simulation : driving innovation*, pp. 124–134. Winter Simulation Conference, 2003.
18. Gasser (L.) et Kakugawa (K.). – Mace3j : fast flexible distributed simulation of large, large-grain multi-agent systems. – In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 2*, pp. 745–752. ACM, 2002.
19. Gutknecht (O.) et Ferber (J.). – Madkit : A generic multi-agent platform. – In *Proceedings of the fourth international conference on Autonomous agents*, pp. 78–79. ACM, 2000.
20. Heath (B.), Hill (R.) et Ciarallo (F.). – A survey of agent-based modeling practices (january 1998 to july 2008). *Journal of Artificial Societies and Social Simulation*, vol. 12, n4, 2009, p. 9.
21. Holcombe (M.), Coakley (S.) et Smallwood (R.). – A general framework for agent-based modelling of complex systems. – In *Proceedings of the 2006 European Conference on Complex Systems*, 2006.
22. Luke (S.), Cioffi-Revilla (C.), Panait (L.) et Sullivan (K.). – MASON : A New Multi-Agent Simulation Toolkit. *Simulation*, vol. 81, n7, juillet 2005, pp. 517–527.
23. Oguara (T.), Theodoropoulos (G.), Logan (B.), Lees (M.) et Dan (C.). – Pdes-mas : A unifying framework for the distributed simulation of multi-agent systems. *SCHOOL OF COMPUTER SCIENCE RESEARCH REPORTS-UNIVERSITY OF BIRMINGHAM CSR*, vol. 6, 2007.
24. Scheutz (M.), Schermerhorn (P.), Connaughton (R.) et Dingler (A.). – Swages-an extendable distributed experimentation system for large-scale agent-based alife simulations. *Proceedings of Artificial Life X*, 2006, pp. 412–419.
25. Smith (E. R.) et Conrey (F. R.). – Agent-based modeling : A new approach for theory building in social psychology. *Personality and social psychology review*, vol. 11, n1, 2007, pp. 87–104.
26. Standish (R. K.) et Leow (R.). – Ecolab : Agent based modeling for c++ programmers. *arXiv preprint cs/0401026*, 2004.
27. Suryanarayanan (V.), Theodoropoulos (G.) et Lees (M.). – Pdes-mas : Distributed simulation of multi-agent systems. *Procedia Computer Science*, vol. 18, 2013, pp. 671–681.
28. Tisue (S.) et Wilensky (U.). – Netlogo : Design and implementation of a multi-agent modeling environment. – In *Proceedings of Agent*volume 2004, pp. 7–9, 2004.
29. Tobias (R.) et Hofmann (C.). – Evaluation of free java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, vol. 7, n1, 2004.