Mise en place et évaluation d'un algorithme de répartition de charge pour les plateformes de simulations distribuées basées sur les systèmes multi-agents

Ines Hassoumi, Christophe Lang, Marilleau Nicolas

Mise en place et évaluation d'un algorithme de répartition de charge pour les plateformes de simulations distribuées basées sur les systèmes multi-agents

Inès Hassoumi
LI3 (99/UR/14-01)
Institut Supérieur de Gestion
41, rue de la liberté
2000 Le Bardo, Tunisie
E.mail : ines.hassoumi@gmail.com

Christophe Lang
LIFC (FRE 2661 CNRS)
Université de Franche-Comté
16 route de Gray
25030 Besançon Cedex
E.mail : christophe.lang@univ-fcomte.fr

Nicolas Marilleau
UMI 209 UMMISCO, IRD , UPMC
Centre de recherche d'Ile-de France
32, avenue Henri Varagnat
F-93143 Bondy cedex. France.
E.mail : Nicolas.marilleau@ird.fr

Résumé_ Cet article traite de la problématique de la répartition de charge dans les systèmes multi-agents à travers un algorithme qui assure la distribution de ces agents. Le besoin est né de l'observation de fréquents problèmes de surcharge lors de simulations basées sur ces systèmes multi-agents. Miro, qui est une plateforme de simulation à grande échelle de la mobilité urbaine en est un exemple concret. La difficulté de ces travaux se situe dans la considération des spécificités des plateformes de simulation orientée agent : autonomie des entités à distribuer et forte imprévisibilité du système. Nous adaptons un algorithme de répartition de charge appelé *Comet* aux spécificités des simulations distribuées à base d'agents. Cet algorithme est basé sur l'emploi d'un indicateur appelé « crédit » qui pour chaque agent quantifie son affinité pour chaque machine et détermine les meilleurs agents candidats à la migration. Hormis l'algorithme en lui même, ce document en présente une implémentation et une évaluation sur un simulateur développé avec Netlogo. Le but final est d'identifier les paramètres à prendre en considération pour assurer le bon fonctionnement de l'algorithme lors de son implémentation sur une plateforme réelle de simulation.
Mots clés : répartition de charge dynamique, distribution des simulations orientées agent, migration des agents.

Abstract_ This paper discusses load balancing in Multi-Agent Systems (MAS) through a generic algorithm based on agent's distribution. The need is born from the observation of frequent load-balancing problems over simulations based on multi-agent systems. Miro which is a software system developed for scalable urban mobility simulation is a concrete example of this type of problems. The difficulty of this work lies in the consideration of simulations specificities: autonomy of distributing entities, high unpredictability of the system…We adapt a load balancing algorithm called "Comet" to specificities of distributed simulations based on multi-agent systems. This algorithm proposes a credit-based load balancing policy which evaluates agent affinity to a machine and determines which the best candidates for migration are. Besides, this paper presents an implementation and an evaluation of the algorithm on a simulator developed with NetLogo. The final purpose is to identify which parameters are suitable to the best running of this load-balancing algorithm on simulations based on multi-agent systems.
Key words: dynamic load balancing, distributed simulations based on multi-agent systems, migration of agents

1. Introduction

La problématique de la simulation distribuée des systèmes multi-agents constitue un enjeu majeur, d'actualité. Les besoins en termes de performances des plates-formes de simulation agents sont grandissants. Ceci est le résultat d'une complexité grandissante des systèmes modélisés et d'un besoin de plus en plus important de réalisme dans les simulations.

Ainsi, dans la plupart des domaines de recherche (aussi bien en sciences du sol qu'en sciences sociales) il existe des modèles, tels que Miro [BCL&al, 2005], Sworm [MIC, 2004], Transim [TRA, 2003] ou Min3p [MAY&All, 2002], dont la réalisation d'une simulation nécessite plusieurs heures, plusieurs jours voire plusieurs semaines pour obtenir des résultats intéressants. Pourtant,

plusieurs travaux [Mar 2006] ont montré, qu'avec une répartition adaptée des agents et de l'environnement sur une grille de calculs, le temps d'exécution d'une simulation pouvait être inversement proportionnel au nombre de nœuds (par rapport au temps d'une expérimentation sur un seul nœud). Concrètement, une simulation multi-agents distribuée sur 4 nœuds peut (dans une configuration idéale) se réaliser plus de 4 fois plus vite que son homologue non distribué.

Dans ce contexte, nos travaux s'articulent justement dans la proposition de techniques et d'outils de répartition de charge qui favorisent la conception et le développement de simulateurs distribués à bases d'agents. Ces simulateurs auront la particularité de pouvoir s'exécuter sur une grille de calcul et de bénéficier d'un système de répartition de charge dynamique et à la demande, en fonction de l'évolution des simulations.

La problématique de la répartition de charge n'est pas une science nouvelle si bien que pléthore d'algorithmes et d'architectures existent. Pourtant, l'utilisation de ces techniques s'avère éparse dans le domaine de la simulation agent même s'il existe quelques normes, techniques et outils, dédiés comme MASIF [OMG, 1998], HLA [LG, 2001] Les stratégies de répartition de charge classiques applicables à la gestion de processus et d'objets distribués ont montré leurs limites dans le contexte de la simulation orientée agents [LAN, 1999]. Les spécificités des systèmes multi-agents (autonomie, interaction, émergences, etc) qui font la force de cette approche constituent un frein vers l'adoption, *in extenso*, de techniques de répartition de charge issues du monde des systèmes distribués. La répartition de charge d'une simulation distribuée orientée agents induit les mêmes difficultés que celles des applications distribuées, auxquelles s'ajoute l'imprévisibilité du système (montée en charge soudaines, communications et activités inopinées des agents) résultant de la dynamique même du système complexe étudié. A cela s'ajoute aussi toutes les problématiques de la gestion : (i) d'une horloge distribuée (un temps simulé qui se déroule de manière parfaitement synchronisée sur l'ensemble des nœuds de simulation) ; (ii) de la concurrence aux ressources critiques des agents dans un contexte de simulation distribuée (par exemple la nourriture, dans le cadre d'un modèle de fourmis fourragère).

Cet article n'a pas la prétention de répondre à l'ensemble des questions précitées. La principale contribution de ce travail se situe dans la proposition d'une architecture de simulateurs distribués et d'un algorithme de répartition de charge. L'architecture proposée repose sur les techniques usuelles des agents mobiles. L'algorithme, que nous appelons LBSIM, se base, quant à lui, sur des résultats issus du domaine de la répartition de charge de logiciel, plus particulièrement sur l'algorithme *Comet*, que nous avons adapté au contexte de la simulation distribuée.
Dans cet article, nous commençons par présenter les contraintes spécifiques aux plateformes de simulation orientées agents, ainsi que quelques algorithmes de la littérature et principalement l'algorithme *Comet* sur lequel nous nous sommes basés dans l'élaboration de ce travail. Ensuite, nous abordons l'architecture de notre système de répartition de charge et présentons l'algorithme. En dernier lieu, nous proposons une discussion sur la base de résultats d'expérimentation.

2. Contraintes de migration des agents dans les plateformes de simulation.

L'une des principales difficultés de la répartition de charge dans les simulations orientées agents est d'accélérer et d'améliorer les temps de simulation sans pour autant perturber son déroulement et donc ses résultats. Ainsi, le mécanisme doit fournir une réponse rapide et efficace à des phénomènes constatés de surcharges par la migration (de nœud en nœud de simulation) des agents et des parties de l'environnement. Ces migrations doivent se réaliser sous certaines contraintes dont :

- Conservation du contexte d'exécution [PER, 1997] [KGR, 1999]: Il doit être possible d'arrêter le processus d'exécution de l'agent et de le rependre là où il s'est arrêté. Il faut donc être en mesure d'obtenir le contexte d'exécution de l'agent (ses connaissances, ses groupes, sa position dans l'espace, son activité, etc), de transférer cet état via le réseau, de le recharger et de relancer le

processus de décision de l'agent là où il s'est arrêté.

- Transparence dans le temps : la migration d'un agent doit être réalisée de manière virtuellement instantanée, c'est à dire sans durée pour la simulation. Si un agent migre à la date simulée t0, il reprend son exécution à la date t0.
- Transparence dans l'espace : un agent doit conserver sa situation dans l'environnement avant et après migration. Par exemple, la position de l'agent dans l'environnement ne doit pas évoluer durant le processus de migration.
- Conservation des accointances : un agent en interaction avec d'autres agents doit être en mesure de conserver ces interactions. Ainsi il est important de disposer d'un mécanisme de désignation global qui permet d'identifier et d'adresser un agent de façon unique et distribuée [VIE, 2003]. Un agent doit porter la même identité malgré ses migrations.

Cette présente liste n'est pas exhaustive. Elle énumère néanmoins des préceptes fondamentaux qui doivent être respectées dans le contexte de la simulation multi-agents. Elle pourrait être complétée par des préceptes clefs de l'AOSE (Agent Oriented Software engineering). Une plate-forme de simulation répartie orientée agent pourrait s'appuyer sur la norme MASIF [OMG, 1998], intégrer des primitives de sécurité [MAN 2004].

3. Algorithmes de répartition de charge

3.1 Le modèle AGR, le clonage et le protocole HLA

Il existe plusieurs algorithmes dans la littérature qui traitent de la problématique de répartition de charge dont on pourrait citer : (i) l'algorithme de migration des agents selon le modèle AGR sur la plateforme MADKIT [AUZ, 2006] qui utilise le concept du Agent/Groupe/Rôle où chaque machine est assimilée à un groupe ; (ii) l'algorithme de clonage des agents [SCJ, 1998] qui propose une autre stratégie de répartition de charge basée sur le clonage local ou distant des agents (iii) les différentes stratégies de répartition de charge basées sur le protocole HLA (High Level Architecture) [LG, 2001] où un agent Manager gère la migration des agents fédérés à chaque constatation de surcharge sur le nœud qui les héberge.

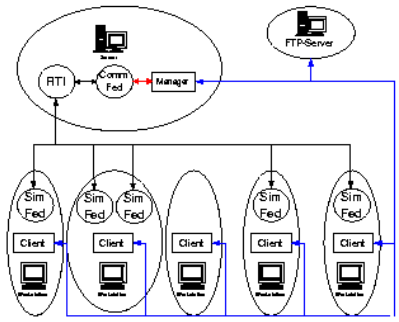

Figure 1 : Echanges entre le manager RSS et les fédérés

L'ensemble de ces algorithmes proposent une approche de répartition de charge très classique et loin d'être optimale. En effet l'inconvénient commun entre l'ensemble de ces algorithmes est le fait qu'ils font déplacer automatiquement l'agent ayant provoqué la surcharge vers une autre machine moins chargée sans pour autant chercher à choisir le meilleur candidat à faire migrer vers la meilleure machine destinatrice capable de l'héberger. Pour cette raison, nous nous sommes intéressés de plus prés à d'autres algorithmes tels que l'algorithme *COMET.*

3.2 L'algorithme *Comet*

Dans ce qui suit, nous présentons avec plus de détails l'algorithme de répartition de charge *Comet* qui nous a fourni une bonne base de travail pour l'élaboration de notre algorithme.

Algorithme *Comet* : [CHK, 2002]
Début
Data : seuil de charge maximale *Th*, Temps de collecte des informations τ (en secondes), Nombre d'agents *n* Caractéristiques des machines hôtes (adresse et autres informations système), Nombre de machines hôtes *p*

1. Initialisation : vérifier la position actuelle des agents
2. While (true) Do {
3. Phase de collection des informations : calculer le crédit de chaque agent et déterminer la charge de chaque hôte
4. Pour Chaque hôte disposant d'une charge sup à *Th* Faire {
5. Faire migrer l'agent possédant le plus petit crédit conformément aux politiques de sélection et de localisation
6. Répéter le processus jusqu'à ce que la charge devienne inférieure à *Th* }
8. Attendre (τ) //attendre le signal du réveil également}

FIN

Analyse de l'algorithme

Il s'agit d'un algorithme de répartition de charge basé sur la technique du calcul de crédit selon la formule suivante :

$Ci = - x1wi + x2hi – x3gi$

Où *wi* : charge de calcul de l'agent *ai*

hi : charge de communication intra-machine de l'agent *ai*

gi : charge de communication inter-machines de l'agent *ai*

et où x1, x2 et x3 sont des réels positifs qui constituent les coefficients de dépendance attribués à chaque agent afin d'estimer son affinité par rapport à la machine.

Pour le calcul de la charge de la machine hôte l'algorithme utilise la formule suivante :

$Lk = \Sigma (wi + ui)$

$M(ai) = k$

Où Lk est la charge de la machine Mk calculée par la sommes des agents *ai* qui se situent sur cette machine, sachant que :

ui : la somme des charges de communication inter et intra machines.

$ui = hi + gi$

$= \Sigma c(ai , aj) + \Sigma c(ai , aj)$

$M(ai) = M(aj)$ $M(ai) \neq M(aj)$

Où c(*ai*, *aj*) : nombre de ticks (unité de mesure du temps) nécessaires pour établir une communication entre deux agents *ai* et *aj*

F : facteur de dégradation de la bande passante dans les communications inter-machines

Selon la politique de sélection : l'agent possédant le plus petit crédit va être sélectionné pour la migration, car cet agent passe une grande partie de son temps d'exécution en communication avec des agents distants.La politique de localisation consiste à identifier l'agent distant qui a généré le plus grand flux de communication avec l'agent à faire migrer. Ainsi la machine dans laquelle réside l'agent choisi va être la nouvelle destination de l'agent.

Limites :

L'algorithme ne traite pas les cas suivants :

- le cas où on se retrouve avec des agents à crédits égaux
- le cas où la machine choisie selon la politique de localisation soit chargée
- le cas où l'agent choisi selon la politique de sélection ne communique avec aucun agent externe (quelle machine choisir ?)
- De plus l'algorithme ne précise pas si les coefficients x1, x2 et x3 d'un agent changeraient suite à sa migration sur une nouvelle machine ou pas.

Il est à noter que l'ensemble des algorithmes trouvés dans la littérature présentent des stratégies de répartition de charge applicables sur les systèmes SMA mais non adaptées aux spécificités des domaines d'application de ces algorithmes à savoir les plateformes de simulation orientées agents.

4. Architecture du système de répartition de charge

Notre mécanisme de répartition de charge est déployé sur une architecture spécifique composée d'agents tels que :

- les agents fixes (représentant les machines de la plateforme),
- les agents mobiles représentent les humains qui se déplacent sur des environnements urbains virtuels. Ils sont composés chacun par une première unité de contrôle constituée par un agent (Brain) qui va diriger les déplacements d'une deuxième unité d'exécution appelée (Body) sur la carte géographique. On va attribuer à chaque agent Brain dès sa création : un identifiant, une adresse (un pointeur sur l'agent), une machine où il va effectuer ses traitements et des coefficients de dépendance à cette machine avec une charge et un crédit initialisés à 0.
- Les agents Managers sont les agents qui contrôlent l'ensemble des machines de la plateforme (par analogie à la notion d'agence dans la norme MASIF) et gèrent le processus de migration des agents mobiles.
- Les agents reporters sont des agents mouchards qui rapportent les activités des agents mobiles dans des rapports destinés à l'agent Manager (un agent reporter par machine par analogie à la notion de place dans la norme MASIF). Dans la variable charge on va juste constater sa charge de traitement (la charge de communication est négligeable).
- Les liens qui relient les agents reporters entre eux, les agents reporters aux agents managers, les agents reporters aux agents brains, les agents brains entre eux et les agents brains avec leurs unités body. Pour chaque lien reliant deux agents Brains ou un agent Brain avec un sa partie Body, on va attribuer une valeur

qui va constituer le poids de l'arc reliant l'agent *ai* à l'agent *aj* c(ai,aj). Ce poids va être exprimé en fonction de sa consommation des ressources : mémoire, CPU, disque dur, interface réseau

Le schéma suivant montre l'architecture du système de répartition de charge nécessaire au déploiement de cet algorithme ainsi que l'enchainement des différentes étapes de l'algorithme.

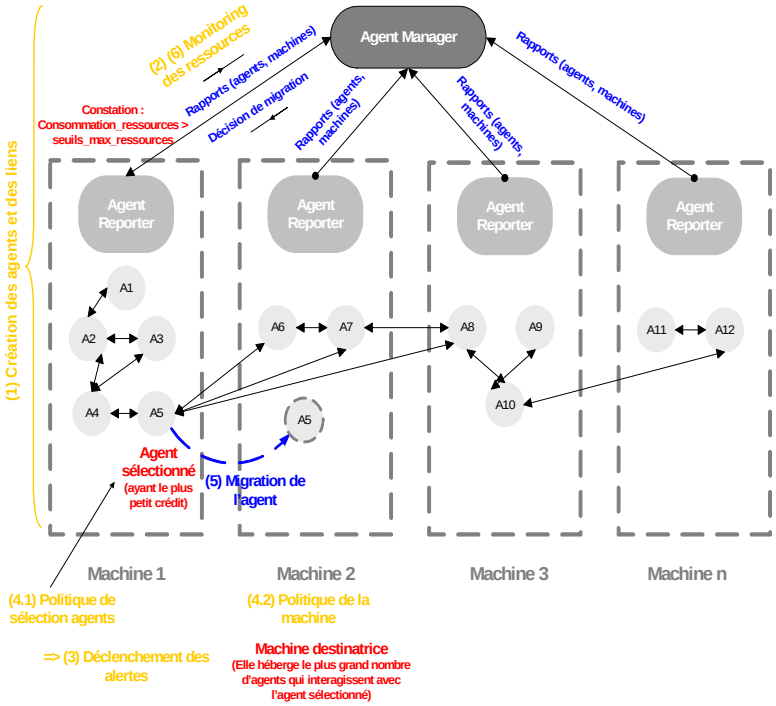

Figure 2 : Architecture du mécanisme de répartition de charge

5. Présentation de l'algorithme LBALG

Nous allons commencer par présenter une version pédagogique de l'algorithme qui illustre le déroulement des six étapes de façon simple.

Algorithme de Répartition de charge

```
Etape 1 Création des agents   Création des agents et des liens
Etape 2 Monitoring des ressources   Monitoring agents
                                    Monitoring ressources
                                    Si dépassement des seuils max alors
Etape 3 Déclenchement des alertes   Déclenchement des alertes
                                    Surcharge ← vrai
                                    //déclenchement du processus de répartition de charge
Etape 4 Exécution du processus de répartition de charge   Tant que surcharge Faire
                                        Agent-migration ← politique de sélection de l'agent
                                        Machine-destinatrice ← politique de sélection de la machine
Etape 5 Migration des agents            Migration-agent
                                        Monitoring ressources
Etape 6 Vérification finale des ressources   Si dépassement des seuils alors
                                                Garder les alertes
                                                Sinon surcharge ←faux
                                            Fin si
                                        Fin Tant que
                                    Fin si
```

Cet algorithme va permettre à l'agent de migrer tout en continuant d'exécuter les tâches qu'il est censé faire. L'agent va pouvoir reprendre son itinéraire au même point d'arrêt et garder une image exacte de son positionnement spatio-temporel même s'il est situé sur un environnement différent et sur une machine distante. Par ailleurs le mécanisme de répartition de charge utilisé va prendre en considération non seulement la charge de travail de l'agent mais aussi la charge de communication de celui-ci (ce qui est généralement négligé par la plupart des algorithmes). Grâce à la notion du crédit, l'algorithme permettra de faire un choix optimal des agents à faire migrer et des nouvelles machines qui vont les accueillir. Par ailleurs, la mobilité physique de l'agent a été déployée tout en respectant les règles de la norme Masif [MBB&alt, 1998]. Nous considérons par ailleurs que cet algorithme permettra de partir d'un état initial où des agents sont dispersés de façon aléatoire sur des machines différentes pour progresser vers un état final où les agents qui communiquent le plus entre eux et qui créent la charge soient regroupés sur la même machine.

Etape 1 : création des agents et des liens

Il s'agit dans cette étape de créer :

- les agents fixes
- les agents mobiles (avec leurs parties Brain et Body).
- Les agents Managers
- Les agents reporters
- Les liens qui relient les différents agents de la plateforme entre eux

Etape 2 : La surveillance des ressources et des agents sur chaque machine

Il s'agit dans cette étape de :

- calculer la charge totale (ai) de l'agent Brain qui va être constituée par sa charge de communication inter-machines (gi), sa charge de communication intra-machines (hi) et sa charge de calcul (wi) (attribuée à l'agent dés sa création) [CHK, 2002]. Notons que la charge de calcul de l'unité d'exécution Body sera négligeable, en fait celui-ci va uniquement se déplacer sur la carte selon les ordres de son agent Brain. On considérera donc que l'unité d'exécution Body va avoir uniquement une charge de

communication, générée suite à ses interactions avec l'agent Brain.

Ainsi la *Charge ai* = *wi* + *hi* + *gi* constitue la charge totale d'un agent *ai*

Où

wi = charge de traitement propre à l'agent

hi = Σ c(*ai* , *aj*)
M(*ai*) = M(*aj*)

gi = Σ c(*ai* , *aj*)
M(*ai*) ≠ M(*aj*)

A chaque création d'un agent on va définir sa charge de calcul et les poids des liaisons qu'il aura avec les différents autres agents en fonction de leur taux de consommation de chaque ressource.

- Calculer le crédit de chaque agent dont la valeur va être calculée de la façon suivante [CHK, 2002]:
  Crédit *ai* = - x1*wi* + x2*hi* – x3*gi*
  Où x1, x2 et x3 sont des réels positifs qui constituent les coefficients de dépendance attribués à chaque agent dés sa création afin d'estimer son affinité par rapport à la machine.
  Ainsi pour dire qu'un agent communique beaucoup avec les autres agents situés sur sa machine on va devoir augmenter la valeur de x2. Par contre, si on désire mettre en phase la communication inter-machines d'un agent on pourra augmenter la valeur de X3.
  A chaque intervalle de temps t, l'agent Reporter rapporte à l'agent Manager la charge, le crédit et la machine de l'agent Brain dont il est responsable. Le tout sera centralisé chez l'agent Manager qui est le seul agent à avoir une vue globale sur le système.
- L'agent Manager va devoir calculer la consommation de chaque ressource sur chaque machine en termes d'unités mémoire, de processeur, de carte réseau et de disque dur (nous nous sommes limités à ces quatre ressources critiques).

Etape 3 : Le déclenchement des alertes

Dés que les seuils max de charge seront atteints pour une de ces ressources sur une machine, l'agent Manager va déclencher une alerte correspondant à la ressource où il y a eu la constatation de la surcharge. Exemple : alerte-mémoire, alerte-CPU, alerte-disque, alerte-int.réseau

Etape 4 : L'exécution du processus de répartition de la charge

L'agent Manager va déclencher le processus de répartition de charge.

Politique de sélection de l'agent :

Il d'agit de sélectionner l'agent ayant le plus petit crédit pour la migration car cet agent passe une grande partie de son temps à communiquer avec des agents distants. En effet l'agent ayant le plus petit crédit sera l'agent qui possède la plus grande charge de communication externe et la plus petite charge de communication interne, étant donné que la formule de calcul du crédit est - x1*wi* + x2*hi* – x3*gi*. Notons que si x2 est petit alors l'agent n'est pas très attaché à la machine sur laquelle il réside. Et si hi est petit alors l'agent n'a pas beaucoup d'interactions avec les agents situés sur sa machine, il pourra donc migrer plus facilement vers une autre machine sans que cela n'augmente sa charge de communication inter-machines.

En cas d'égalité entre les crédits de deux agents ou plus, il faudra ajouter un deuxième critère pour la sélection de l'agent. On prendra l'agent ayant la plus petite valeur x2 * hi. En cas d'égalités encore, ceci voudra dire que la migration des deux agents va donner le même résultat. On pourra donc effectuer un choix aléatoire entre les deux agents sélectionnés.

On ajoutera un compteur pour calculer le nombre de migration effectué par l'agent. Si l'agent effectue plus de quatre migrations successives (étant donné qu'on a quatre machines) alors nous devons constater un effet de cycle. On empêchera l'agent de migrer si celui-ci est sélectionné une nouvelle fois après l'application de la politique de sélection de l'agent. Il faudra cependant ajouter un système de timing pour enlever ce verrou après un certain laps de temps

Politique de sélection de la machine destinatrice :

Le principe étant de choisir la meilleure machine qui soit disponible en termes de ressources et qui ne va pas augmenter la charge de communication inter-machine de l'agent [CHK, 2002], notre politique va changer selon la nature des interactions que développe l'agent avec le reste des

agents. Ainsi, si l'agent possède des liens avec des agents distants, on va sélectionner la machine qui héberge le plus grand nombre d'agents Brains ayant interagi avec l'agent sélectionné durant la simulation [CHK, 2002]. Ainsi si l'agent migre vers cette machine, ses communications inter-machines vont se transformer en des communications intra-machines nettement moins coûteuses.

Avant d'effectuer la migration de l'agent on va vérifier au préalable que cet agent ne va pas provoquer la saturation d'une des ressources de la machine. Pour cela on va recalculer la consommation de chaque ressource sur la machine en ajoutant la charge de l'agent. Si l'ajout de la charge de l'agent provoque une surcharge au niveau d'une ou de plusieurs ressources de la machine, ceci voudra dire que même si cette machine offre une répartition optimale de la charge de l'agent elle ne pourra pas l'héberger étant donné qu'il y a un risque de surcharge. On va donc chercher la machine suivante qui héberge le plus grand nombre d'agents ayant généré le plus grand trafic avec l'agent sélectionné. Ce processus va se répéter jusqu'à trouver une machine disponible ou jusqu'à faire le tour des machines qui hébergent les agents ayant interagi avec l'agent sélectionné.

Si maintenant l'agent n'a aucun lien avec d'autres agents distants ou si on n'a pas trouvé une machine disponible parmi celles qui communiquent avec l'agent, on va sélectionner la machine la moins chargée c'est-à-dire celle ayant la plus petite valeur de Lk avec :

$$Lk = \Sigma (wi + hi + gi)$$
$$M(ai) = k$$

Où Lk est la charge de la machine Mk composée de la somme de la charge des agents *ai* qui résident sur cette machine.

Etape 5 : La migration de l'agent sélectionné vers la nouvelle machine

Une fois que la décision de faire migrer un agent est prise, l'agent Manager contactera respectivement l'agent Reporter de la machine source et de la machine destination. Ces deux agents vont être prévenus de la décision de migration, du numéro de l'agent sélectionné et de la machine destinatrice. Le processus de migration consiste en une initialisation d'un nouvel agent sur la machine de destination accompagnée d'une terminaison de l'activité de l'agent sélectionné sur la machine source [CHK, 2002]. Le schéma suivant illustre ce mécanisme :

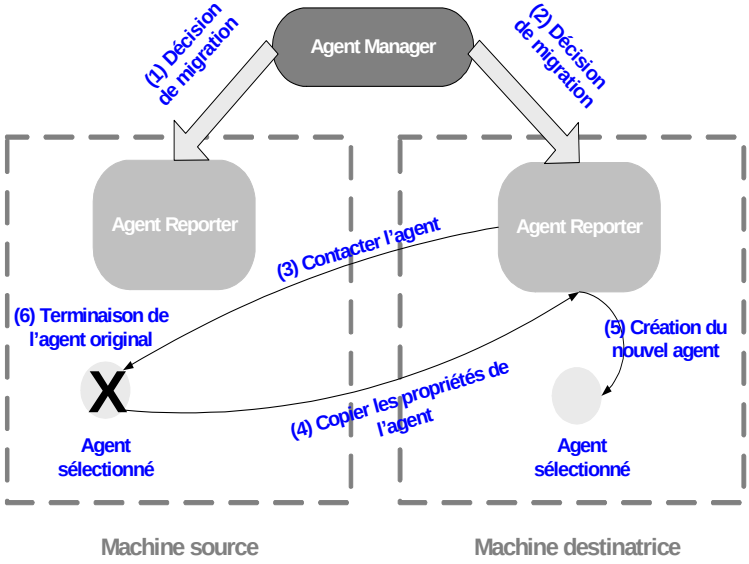

Figure 3 : Processus de migration des agents

Ainsi à ce niveau l'agent Brain va se déplacer vers la nouvelle machine et laisser sa partie Body sur la machine hébergeant l'environnement géographique auquel il appartient. Si le nombre de liens externes qu'entretient désormais l'agent sur sa nouvelle machine avec les autres agents est inférieur au nombre de liens internes, alors son coefficient x2 va augmenter, on va le multiplier par 10. Si par contre le nombre de liens externes augmente (ceci pourrait arriver si l'agent migre vers la machine la moins chargée pour les raisons précitées sans application de la politique de localisation) alors x3 doit augmenter, on lui ajoutera le nombre de liens externes de l'agent. La modification des coefficients x2 et x3 a pour but de renforcer ou de diminuer le degré d'affinité de l'agent par rapport à la nouvelle machine et de pouvoir recalculer la nouvelle valeur du crédit de l'agent en prenant en considération sa nouvelle situation. Ceci va implicitement permettre d'éviter l'effet de cycle dans la migration des agents (il ne sera plus un candidat favorable pour la migration lors de la prochaine surcharge constatée puisque son crédit va augmenter), ce qui va contribuer dans la stabilisation du système.

Etape 6 : La vérification finale de la charge des machines

Pour vérifier si le processus de migration a réglé le problème de saturation des ressources on va recalculer la nouvelle charge de l'agent ai sélectionné et la nouvelle charge de la machine cible. L'agent Manager revérifie les consommations de chaque ressource sur la machine. S'il constate encore un dépassement des seuils de saturation des ressources, il garde les alertes et il continue le processus de répartition de charge. Si la migration de l'agent a permis une répartition optimale de la charge. On annule les alertes.
Si aucune machine ne satisfait les contraintes de la politique de localisation, cela voudra dire que toute la plateforme est surchargée, on signale un état de surcharge globale et on laisse l'agent là où il est jusqu'à ce qu'il y ait une machine disponible.

6. Résultats expérimentaux

Pour assurer une implémentation de l'algorithme de répartition de charge, nous avons mis en place un simulateur basé sur la plateforme de simulation Netlogo [SKL, 2007].
Dans cette section nous allons effectuer une série de tests en enregistrant des mesures spécifiques portant sur les critères suivants :

- **Le quotient (charge totale des machines / somme des seuils max des ressources)** : Ce quotient permet de mesurer la différence entre la charge totale de la plateforme et les ressources disponibles. En effet plus ce quotient est grand plus la charge totale de la plateforme est proche des ressources disponibles et plus le risque de surcharge augmente. Ce quotient constitue par ailleurs un bon indicateur quant à la bonne exploitation des ressources de la plateforme avant et après la répartition de la charge.
- **Constatation d'une stabilisation** : ce critère permet de vérifier si l'algorithme permet à la plateforme d'atteindre la stabilité dans différents cas de figures. On entend par stabilité, le fait qu'il n'y ait plus de surcharge au niveau des machines de la plateforme.
- **Mesure du temps de stabilisation** : Le temps de stabilisation a un grand impact sur l'efficacité du système et de l'algorithme de répartition de charge. Plus ce temps est petit, plus le système va reprendre rapidement son activité principale (à savoir dans notre cas la simulation d'un environnement urbain).
- **Mesure du delta de charge entre les machines** : Pour mesurer cet indicateur on va calculer la différence de charge entre la machine la plus chargée et la machine la moins chargée sur la plateforme. Plus cette variation est grande plus le risque d'avoir des machines surchargées et des machines faiblement chargées sur la plateforme augmente.
- **Calcul du nombre de migration** : Le nombre de migration permet de savoir les facteurs favorables à l'augmentation ou à la diminution du nombre de migrations effectuées sur la plateforme.

Plan d'expérience
Nous avons supposé que les environnements géographiques simulés sur notre plateforme étaient distribués sur quatre machines ayant les caractéristiques suivantes :

Mémoire principale	2 G octets
Processeur	3.33 Ghz équivalent à 2800 Moctets/s
Disque dur	80 G octets
Interface réseau	100 Mbits /s

Tableau 1 : Configuration matérielle des machines simulées

Nous avons procéder par la suite à une série de tests afin de tester l'effet de la variation de certains paramètres dont on pourrait citer:

- nombre d'agents mobiles
- seuils de saturation des machines
- charge des agents
- coefficients des agents

Lors de chacune de ces expérimentations, nous avons testé un seul paramètre à la fois. Dans ce qui suit nous présentons quelques extraits des résultats trouvés :

Nombre d'agents	10	20	30
Quotient (CT/SS)	31,19%	37,61%	45,25%
Stabilisation	Oui	Oui	Non
Temps de stabilisation (secondes)	1	10	N/A
Delta de charge	12000	15000	3000
Nombre de migration	1	3	1 (avec deux demandes de migration non effectuée)

Tableau 2 : Impact de la variation du nombre d'agent sur le processus de répartition de charge

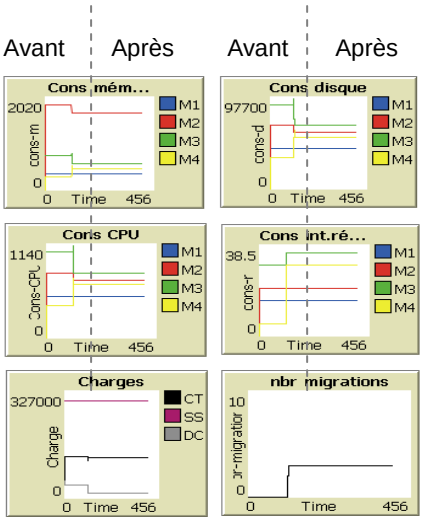

Figure 4 : Résultats de la répartition de charge avec 20 agents

Les résultats présentés ci-dessus montrent que notre mécanisme de répartition de charge a permis de réduire la consommation des quatre machines M1, M2, M3 et M4 en termes d'unités de mémoire, de disque dur, de processeur et que celle-ci a légèrement augmenté la charge réseau suite aux migrations des agents vers d'autres machines externes. On constate également que lorsque le nombre d'agents est égal à 20, la charge totale de la plateforme diminue (figure 5 courbe noire histogramme des charges) et la répartition de la charge sur les machines se fait mieux puisque l'indicateur delta de charge diminue (figure 5 courbe grise histogramme des charges). Ceci peut être expliqué par le fait que plus le nombre d'agents augmente plus le choix des agents pour la migration est large. D'autant plus que l'augmentation du nombre d'agents favorise l'augmentation de la communication avec le reste des machines et donc le choix de la machine destinatrice devient meilleur. Notons cependant, que l'algorithme est inefficace quand il y a une surcharge globale sur la plateforme.

Coefficients des agents	X1 = X2	X2 > X3	X1 = X2 = X3	X3 >x2
Quotient (CT/SS)	51,01%	50%	47,97%	46,59%
Stabilisation	Oui	Oui	Oui	Oui
Temps de stabilisation	3	3	3	8
Delta de charge	31000	19000	28000	19000
Nombre de migration	1	1	1	3

Tableau 3 : Impact de la variation des coefficients des agents sur la répartition de charge

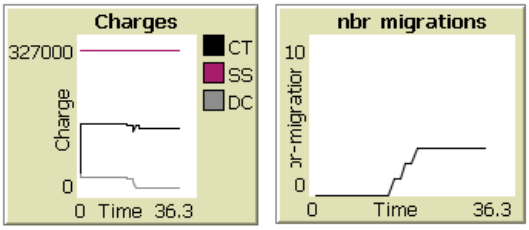

Figure 5 : Résultat de la répartition de charge avec des agents ayant des coefficients x3 supérieurs à x2

On constate d'après les résultats obtenus que le nombre de migration augmente quand le coefficient de charge externe x3 est supérieur à x2, ceci est expliqué par le fait que l'algorithme sélectionne les agents ayant le plus faible crédit et donc ayant le plus grand coefficient x3, ainsi plus x3 augmente plus la probabilité de sélection de l'agent pour la migration augmente.

7. Discussion

A travers ces différentes expérimentations nous avons noté que l'algorithme de répartition de charge offre une grande stabilisation pour la plateforme de simulation avec un temps de réponse remarquablement faible et une réduction considérable de la différence entre les charges des machines. Nous avons également pu conclure que l'algorithme de répartition de charge fonctionne mieux avec un grand nombre d'agents étant donné que son choix de l'agent à faire migrer et de la machine destinatrice devient plus large à condition que le nombre d'agents choisi ne provoque pas une surcharge globale sur la plateforme. Nous avons pu déduire que la fixation des seuils de saturation des machines va avoir un grand impact sur l'augmentation ou la diminution du nombre de migrations effectuées par les agents. Nous avons obtenu notamment de meilleurs résultats par la création d'agents avec des charges différentes, et nous avons noté à la fin que le nombre de migration diminue avec l'augmentation du coefficient x2 et qu'il s'élève avec l'augmentation du coefficient x3. Tous ces paramètres sont à prendre en considération incontestablement lors de l'implémentation de l'algorithme sur la plateforme de simulation Miro.

En se basant sur ces observations, nous proposons certains axes d'amélioration qui vont constituer une extension propice pour notre étude. En effet, dans un souci d'amélioration du temps de réponse de la

plateforme de simulation, nous proposons de mettre en place un système capable de prévoir la surcharge d'une machine. Ainsi, suite à une estimation de la future charge de l'agent (examen de sa liste de tâches), l'agent répartiteur devrait être capable de prévoir que la charge de l'agent va probablement augmenter dans un futur proche. Celui-ci va devoir donc prendre les mesures nécessaires et demander à l'agent Manager de faire migrer l'agent en question avant même la constatation d'une surcharge au niveau de la machine qui l'héberge.

8. Conclusion

Dans ce travail, nous avons présenté un algorithme qui propose une amélioration d'autres algorithmes existants afin d'assurer son adaptation à la plateforme de simulation orientée agent et particulièrement à la plateforme de simulation de la mobilité urbaine Miro. Cet algorithme est basé sur l'emploi des agents pour la répartition de charge et utilise la notion de crédit qui permet d'élaborer une politique de sélection de l'agent à faire migrer et une politique de localisation de la machine destinatrice.
Ainsi contrairement aux stratégies de répartition de charge classiques qui choisissent par défaut l'agent ayant causé la surcharge pour le faire migrer par défaut aussi vers la machine la moins chargée, la première politique que nous utilisons considère que l'agent ayant le plus gros volume de communication externe et le plus faible degré d'affinité avec sa machine (et donc le plus petit crédit) constitue le meilleur agent à faire migrer suite à la constatation d'une surcharge sur sa machine. La deuxième politique de localisation considère que la machine qui héberge le plus grand nombre d'agents distants qui interagissent avec l'agent sélectionné constitue la meilleure machine à sélectionner pour la nouvelle destination de l'agent. Notre approche a donc pour avantage d'assurer un choix optimal de l'agent à faire migrer et de la machine destinatrice qui va l'héberger tout en prenant en considération des contraintes relatives à la simulation de la mobilité urbaine et tout en respectant les normes de mobilité en l'occurrence la norme Masif lors de la migration de l'agent.

Références bibliographiques

[AUZ, 2006] : Arnaud Auzolat, 2006: Modèle organisationnel de répartition de charges pour des agents dans des contextes contraints. Application à la résolution de problèmes et à la simulation multi-agent distribuée sur le GRID, Université Montpellier II

[BCL&al, 2005]: Arnaud Banos, Sonia Chardonnel, Christophe Lang, Nicolas Marilleau, Thomas Thévenin, Une approche multi-agents de la ville en Mouvement 2005

[CHK, 2002] : Ka-Po Chow and Yu-Kwong Kwok, Member, IEEE : On Load Balancing for Distributed Multiagent Computing, 2002

[KGR, 1999]: David Kotz and Robert S. Gray, Department of Computer Science / Thayer School of Engineering Dartmouth College.Mobile Agents and the Future of the Internet, 1999

[LAN, 1999] Christophe LANG Répartition de charge dynamique à l'initiative des processus : étude, algorithmes et implémentations Décembre 1999

[LG, 2001]: Johannes Luthi, Steffen Grobmann The ressourec sharing system dynamic federate mapping for HLA- bases distributed simulation, 2001

[MAN, 2004] : Mansour Saber, Mobilité sur la plateforme Madkit, rapport DEA 2003-2004

[Mayer, 2002] : Mayer K.U, Frind E.O and Blowes D W, 2002. Multicomponent reactive transport modeling in variably saturated porous media using a generalized formulation for kinetically controlled reactions. Water Resources Research

[MAR, 2006] : Nicolas Marilleau, Méthodologie, formalismes et outils de modélisation-simulation pour l'étude des systèmes complexes : application à la mobilité géographique, L'UFR des Sciences et Techniques de l'Université de Franche-Comté, 2006

[MIC, 2004] : Formalisme, outils et ´el´ements m´ethodologiques pour la mod´elisation et la simulation multi-agents par Fabien Michel, 2004

[OMG, 1998] MASIF: The OMG Mobile Agent System Interoperability Facility.1998

[PER, 1997]: Stéphane PERRET, THESE de doctorat soutenue le 19 novembre 1997 : Agents mobiles pour l'accès nomade à l'information répartie dans les réseaux de grande envergure.

[SCJ, 1998] : O Shehory, K Sycara, P Chalasani, S Jha - IEEE Communications Magazine, 1998 - comsoc.org Agent cloning: an approach to agent mobility and resource allocation

[SKL, 2007] : Elisabeth Sklar : Netlogo, a multi-agent simulation environment. Artificial Life, 13(3):303–311, June 2007

[TRA, 2003] : Transims, Transportation Analysis Traffic and Simulaton Systems, Los Alamos National Laboratory, 2003,

[VIE, 2003] : H. Tran Viet, Gestion De La Mobilité Dans L'ORB Flexible Jonathan 2003