COMPARAISON D'APPROCHES DE SIMULATIONS DISTRIBUEES A EVENEMENTS DISCRETS D'ENTITES SPATIALISEES

Gauthier Quesnel,
Doctorant en informatique
quesnel@lil.univ-littoral.fr , + 33 3 21 46 56 79

Raphael Duboz,
Doctorant en informatique
duboz@lil.univ-littoral.fr + 33 3 21 46 56 79

Eric Ramat,
Maître de conférences en informatique
ramat@univ-littoral.fr + 33 3 21 46 56 73

Adresse professionnelle
Laboratoire d'Informatique du Littoral Mission de la Recherche Blaise Pascal
50, rue Ferdinand Buisson - BP 719 62228 CALAIS cedex FRANCE

Résumé : Dans cet article, nous voulons répondre à la question suivante : quelle technique choisir pour la simulation à événements discrets de particules intéragissant localement dans un espace continu à trois dimensions, et ceci dans un cadre distribué ? Pour cela, nous comparons les performances de trois méthodes classiques de simulation distribuée : une approche optimiste et deux approches pessimistes (faible et forte). Cette étude nous a amené à choisir la technique pessimiste faible comme une réponse possible.

Summary : In this article, we adress a particular issue: which technique to choose for a distributed discret event simulation of interacting particles in a three-dimensional coutinuous space? To answer, we compare three techniques of distributed simulation: strong synchronous pessimistic, weak asynchronous and optimistic. This study lead us to choose the weak asynchronous one.

Mots clés : Modélisation centrée individus, Simulation distribuée, Événements discrets, MPI.

Comparaison d'approches de simulations distribuées à événements discrets d'entités spatialisées

1 INTRODUCTION

Ce travail se situe dans une perspective d'étude de la dynamique des écosystèmes marins à petite échelle. On s'intéresse en effet plus particulièrement à l'interaction entre des cellules de phytoplancton (algues microscopique) et des animaux zooplanctonique (Ramat, 2003). Les interactions identifiées sont principalement les captures du phytoplancton par les prédateurs. Nous avons développé et simulé en monoprocesseur des modèles (Duboz 2001) de ce système pour des tailles de population de l'ordre de 10^5 phytoplanctons et 100 animaux. Ces quantités nous ont permis d'étudier la dynamique du système sur des échelles de temps et d'espace réduites.

La deuxième étape est donc de développer des modèles de simulation capable d'atteindre des tailles de population bien supérieures aux tailles actuelles (de l'ordre de 10^6 phytoplanctons et 10^3 prédateurs. Ce facteur 10 est impossible à atteindre en monoprocesseur faute de mémoire et de puissance.

Le travail présenté dans cet article est une première étape indispensable permettant de valider un modèle de simulation distribué.

Afin de réduire la complexité du problème, nous avons éliminé les interactions entre proies et prédateurs, ce qui réduit le système à un ensemble de particules effectuant une marche aléatoire simulant ainsi un mouvement brownien. Nous nous sommes focalisés sur deux éléments : la gestion du déplacement des prédateurs (que l'on nommera par la suite particules) et la synchronisation des événements liées à ces déplacements.

Ce dernier point est fondamental. En effet, dans le système que l'on cherche à développer, les particules interagissent et ces interactions nécessitent obligatoirement une synchronisation temporelle. A t=0, chaque particule est positionnée dans l'espace. Elles se déplacent et se situent donc à chaque instant à un endroit précis dans cet espace. Deux particules peuvent alors interagir si elles se situent au même instant dans la même zone. Cette contrainte n'est pas toujours vérifiée dans les modèles de la littérature (Miguet 1995).

En simplifiant le modèle, nous avons fait disparaître, les calculs complexes qui sont réalisés au sein des particules (calculs liés au comportement et à la dynamique interne des prédateurs). Nous pallierons à ce biais en ajoutant fictivement une charge de calcul au sein des particules. Cette charge de calcul fera l'objet d'une étude afin de déterminer son impact sur les stratégies adoptées.

Le système simplifié possède un précieux avantage : on connaît parfaitement son comportement ce qui nous permettra de valider nos algorithmes.

Dans la première partie, nous abordons la description de notre modèle de mouvement brownien. En seconde partie, nous étudions trois techniques de distribution de calcul, deux pessimistes et une optimiste. Dans la troisième partie nous montrons les résultats des simulations. Enfin, dans la dernière partie, nous engagerons une discussion sur les stratégies possibles.

2 LE MODÈLE

Le système que l'on étudie est un ensemble de particules situés dans un espace à trois dimensions et pourvu d'un mouvement brownien. Nous avons modélisé ce mouvement de la manière suivante :

- à un instant t, la particule se situe à une position représentée par un triplet (x, y, z) dans un espace à trois dimensions.
- la particule possède une direction (x, y, z) dont chacune des composantes est initialisées aléatoirement entre -1 et +1.
- la particule possède une vitesse de déplacement générée aléatoirement selon une distribution gaussienne bornée de moyenne 1, d'écart type 0.2 et de limites 0 et 2. Ces valeurs sont inspirées de la vitesse observée des prédateurs.
- la particule se déplace pendant un temps Δt généré aléatoirement selon la même distribution que la vitesse.

À l'issue de Δt, les particules possèdent une nouvelle position et les composantes du mouvement sont régénérées aléatoirement.

Ce modèle nous permet d'identifier les événements du système qui sont tout simplement les instants de changement de direction. Cette approche s'inscrit dans le cadre de la simulation à événement discret. Entre deux changements, la particule se déplace sur une droite. Si on replace le discours dans le cadre de la simulation à événements discrets, l'avancement du temps est dirigé par les événements.

Le temps progresse donc dans la simulation au fur et à mesure que les événements sont traités. Pour gérer la chronologie des événements, ceux-ci sont triés par ordre croissant dans une liste appelée *échéancier*.

Au départ de la simulation, toutes les particules sont positionnées aléatoirement dans l'espace et possèdent un vecteur de direction. Le couple « *particule - temps de déplacement* » est ensuite inséré dans l'échéancier. L'échéancier est ensuite dépilé, afin de sélectionner la première particule devant se déplacer.

Suite au déplacement, deux cas de figure sont à envisager. En effet, nos particules sont plongées dans un espace borné et la stratégie de distribution que l'on va adopter, comme dans la majorité des modèles de ce type (Miguet 1995), implique un découpage de l'espace. Il y a donc deux types de frontières : celles liées à la limite de l'espace modélisé et celles liées au changement de sous espaces. Il y a donc deux politiques de déplacement lorsqu'une particule atteint une frontière. Dans le premier cas, nous itérons le processus de génération des composantes de déplacement jusqu'à ce qu'ils soient orientés dans l'espace valide. Dans le deuxième cas, la particule devra se situer dans un nouveau sous-espace après avoir franchi la frontière. Dans ce cas nous ne changeons pas les composantes du déplacement, seul change le temps de déplacement qui est diminué du temps de déplacement déjà effectué pour arriver sur la frontière.

3 LES TECHNIQUES

La stratégie de distribution adoptée concerne tout d'abord le découpage de l'espace en sous-espaces en y associant les particules appartenant à ces sous-espaces. Cette stratégie a tendance à repartir à la fois la charge de calcul et le besoin en terme de stockage d'information.

La question qui se pose alors est la suivante : quel est l'algorithme de simulation distribuée optimal dans notre cas ? Nous rappelons que l'aspect synchronisation globale des évènements, connu sous l'appellation causalité que nous développerons dans le cadre général, est vital dans l'hypothèse où les particules doivent interagir.

Nous allons dans la suite développer trois algorithmes de simulation distribuée (Fujimoto 1999), puis nous étudierons leurs conséquences sur notre modèle.

3.1 La causalité

Les systèmes physiques obéissent toujours au *principe de causalité*. Il peut être énoncé de la façon suivante : le futur n'influence jamais le passé. La causalité impose donc un ordre des transitions d'états dans les systèmes physiques. Lamport (Lamport 1978) a défini une méthode basée sur l'utilisation d'estampilles pour permettre aux processus de respecter la causalité. L'ordre dans lequel le simulateur produit les événements doit être cohérent pour que le simulateur soit un reflet juste du système physique simulé. Par exemple, si la transition A du système physique intervient à la date 3 et à une influence sur la transition B intervenant à la date 5, le simulateur doit générer l'événement modélisant la transition A avant celui concernant la transition B.

La distribution d'une simulation sur plusieurs machines pose un problème pour le respect du principe de causalité. En effet, les calculs étant répartis, l'avancement du temps simulé peut être différent sur chaque machine. Le problème principal est alors de définir des techniques de synchronisation des calculs. Dans ce qui suit, nous présentons trois techniques possibles. Nous les évaluons en terme de temps de calcul par rapport à une approche monoprocesseur afin de choisir la plus adaptée à notre problème.

3.2 Approche pessimiste : synchrone forte

La première approche, la plus simple, repose sur le principe suivant : chaque simulateur prend la décision de faire avancer la simulation de son sous-espace uniquement quand il est sûr de pouvoir le faire. Cette technique est dite pessimiste et fait partie des premiers algorithmes de synchronisation

(Chandy, 1978). Elle sera reprise dans l'approche synchrone faible.

Dans le cas de l'approche synchrone, un simulateur peut traiter tout événement qui n'a pas de conséquence sur les simulateurs jusqu'à une certaine date appelée barrière de synchronisation. Dans notre cas, nous sommes obligés de considérer que tout événement peut avoir une conséquence d'où l'algorithme adopté. Un processus central gère une horloge globale. Ce processus est appelé coordinateur et donne la main au simulateur qui possède l'évènement dont la date d'occurrence est la plus petite globalement. Cette technique s'inspire directement des travaux de Zeigler et des simulateurs abstraits associés (Zeigler 2000). Dans cette approche, il n'existe aucun parallélisme. Elle offre la possibilité de quantifier le coût en temps engendré par les communications entre les simulateurs, puisque globalement le simulateur synchrone possède un comportement identique au simulateur mono-processeur.

3.3 Approche pessimiste : asynchrone faible

La technique asynchrone pessimiste a pour principe de faire avancer chaque simulateur d'une manière asynchrone tant qu'aucun simulateur ne viole le principe de causalité.

Le principe est le suivant : chaque simulateur possède un échéancier qui fourni la date minimale (date jusqu'à laquelle un simulateur peut traiter les événements sans conséquence sur ces voisins). Celle-ci est envoyée à tous les simulateurs. Si toutes les dates reçues sont supérieures à sa date minimale, le simulateur traite son prochain événement et envoie à tous ses voisins sa date minimale. Le simulateur ne pourra recommencer à dépiler son échéancier seulement quand il aura reçu des dates supérieures à son temps courant des autres processus.

La technique est légèrement différente dans notre implémentation afin de mieux s'adapter à nos besoins (par exemple, utilisation d'un coordinateur pour éviter les DeadLock). Chaque simulateur possède une horloge et calcule une date jusqu'à laquelle aucun message ne peut être envoyé à un autre simulateur. Ces dates sont envoyées au coordinateur, celui-ci sélectionne la plus petite et demande aux simulateurs d'avancer jusqu'à cette date. Si le simulateur détenant la date maximum de déplacement autorisé, modifie sa prochaine date de sortie de particules, il l'envoi au coordinateur qui demandera à tous les nœuds d'envoyer leurs dates maximums afin de sélectionner le simulateur qui détiendra la prochaine date de déplacement de particules.

Le calcul de la date minimale avec sortie d'une particules de l'espace, s'effectue comme suit :

Début

Pour toute particule contenue dans l'espace du simulateur **faire**

- Calcul pour chaque axe (abscisse, ordonné, profondeur), la date à laquelle la particule pourrait sortir en utilisant des vecteurs direction parallèle aux axes
- Sélection de la date minimale de sortie des trois dates.

Fin pour

Sélection de la date minimale de toutes les particules

Fin

Cette technique assure une intégrité du temps (l'horloge ne peut recevoir une date inférieure à celle courante) même si son fonctionnement reste asynchrone.

3.4 Approche optimiste : le Time Warp

Dans une approche optimiste, chaque processus décide de faire avancer la simulation, sans se préoccuper de l'état des autres. Dans ce cas, le principe de la causalité peut être corrompu. Pour rétablir ce principe, Jefferson (Jefferson 1985) propose une solution : le *Time Warp*.

Lorsqu'un processus s'aperçoit que le principe de la causalité n'est plus vérifié, c'est à dire que la date courante de la simulation est postérieure à la date du dernier événement reçu, il annule tous les messages déjà expédiés. Pour cela, il faut gérer un historique des états du système, mais aussi le contenu de tous les messages reçus et expédiés afin de pouvoir avertir les autres processus de faire des annulations. Ce retour en arrière dans le temps est appelé “BackTracking”. Pour cela, nous utilisons des *anti-messages*, qui possèdent exactement les mêmes caractéristiques que les messages d'origine à l'exception d'un flag.

Ces messages sont générés dès qu'un problème de causalité survient. Ils sont

envoyés aux simulateurs pour annuler certains messages. Par exemple, un simulateur A envoi un message M1 au simulateur B qui arrive dans le futur de celui-ci. Si B envoi un message M2 dans le passé de A, celui-ci repasse à l'état sauvegardé où il se trouvait à la date d'arrivée du message M2. Le processus A doit annuler tous ses envois de messages qu'il a fait dans le futur, comme le message M1 en utilisant les *anti-messages*.

Après la description des trois méthodes de synchronisation, nous allons valider leurs modèles à l'aide d'un modèle déterministe.

3.5 Validation

La modélisation du mouvement brownien nous permet de valider notre modèle distribué par comparaison entre les résultats de simulation de la diffusion particulaires discrète avec un modèle mathématique déterministe (Duboz 2003). Nous plaçons toutes les particules au centre de l'espace au point (0, 0, 0). Nous simulons un certain temps *t,* ce qui permet, à l'aide d'une mesure de la distance moyenne parcourue par les particules, de calculer le coefficient de diffusion (Duboz 2003). Pour visualiser les résultats de simulation, nous découpons l'espace en sous espaces suivant l'axe des abscisses x et comptons le nombre de particules dans chaque sous espace afin d'obtenir une courbe de répartition des particules selon x. Nous cherchons à montrer que ces mesures respectent le modèle mathématique suivant (Fick 1855) :

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

où c est la concentration de particules, D le coefficient de diffusion et x l'axe sur lequel a lieu la diffusion.

À partir de cette valeur, nous pouvons calculer analytiquement la courbe de Fick donnée par l'équation suivante :

$$c(x,t) = \frac{c_0 e^{-x^2/4Dt}}{2\sqrt{\pi D t}}$$

où c_0 représente la concentration initiale, $c(x,t)$ la concentration en un point à un instant donné et D le coefficient de diffusion.

Le résultat se présente sous la forme d'une gaussienne comparable à celle obtenue par simulation (figure 1).

Le résultat que nous obtenons montre effectivement que notre modèle distribué dans les cas synchrone fort et faible suit bien la courbe de Fick, validant ainsi notre modèle distribué par rapport au modèle monoprocesseur.

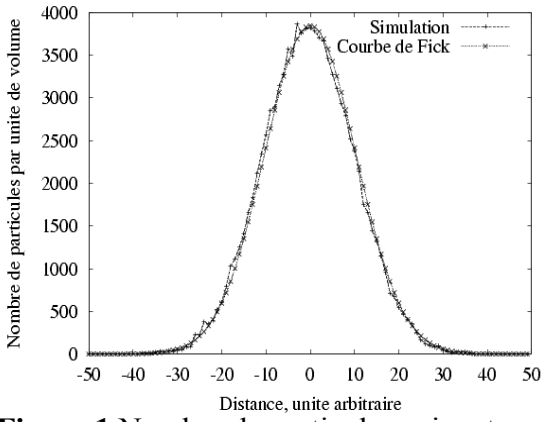

Figure 1 Nombre de particules suivant l'axe des x après une simulation de 30 secondes. La courbe en pointillés correspond au modèle analytique, elle se confond presque parfaitement avec la courbe en trait plein correspondant à la simulation particulaire.

4 LES RESULTATS

Après avoir vérifié empiriquement la validité de nos implémentations, nous allons décrire l'architecture matérielle et logicielle utilisée avant d'en analyser les performances.

4.1 Matériels et logiciels

La réalisation de ce simulateur est basée sur l'utilisation d'un cluster de PC et d'une bibliothèque de programmation réseau.

Nous avons utilisé la bibliothèque de communication réseau MPI (Message Passing Interface) qui nous permet d'assurer une correcte implémentation des approches synchrone et asynchrone :

- Les messages sont envoyés et reçus dans un file d'attente FIFO qui nous assure que si le simulateur A envoi deux messages M1 au temps tA1 et M2 au temps tA2 à B, le simulateur B les réceptionnera dans l'ordre M1 au temps tB1 et M2 en tB2.
- Les messages peuvent être envoyés de manière bloquante ou non bloquante.

- La sûreté, MPI vérifie en effet l'intégrité des paquets qu'il transfert sur le réseau.
- La rapidité de l'envoi et de la réception de messages et un point très important pour notre simulateur (l'ordre de grandeur du nombre de messages avoisine couramment le million lors d'une simulation), MPI est une bibliothèque de haut niveau très proche de la communication par socket.

Le deuxième point fondamental dans le type de travaux que l'on mène concerne les algorithmes de génération aléatoire. Étant au cœur de la validité de nos résultats, ce point n'est pas à négliger. De plus il faut prendre des précautions dès que l'on est dans un cadre distribué (L'Ecuyer 1998).

Le générateur de nombres aléatoires utilisé dans le simulateur est celui fourni par la bibliothèque Glib (bibliothèque de fonctions C utilisée dans le cadre du projet GNU[1]). Ce générateur est basé sur le *Mersene Twister* développé par M. Matsumoto et T. Nishimura (Matsumoto 1998). Son principal avantage est d'avoir une très longue période (2^{19937}-1 tirages indépendants). Dans notre système, un générateur est placé sur chaque nœud fournissant ainsi les nombres aléatoires pour le déplacement des particules. Cependant, son utilisation dans un environnement distribué n'est pas une bonne solution du fait que chaque nœud doit fournir une suite de nombres aléatoires non corrélés entre eux. Une des solutions est d'utiliser le générateur de nombres de P. L'Ecuyer (L'Ecuyer 2002) qui permet d'utiliser la même graine pour chaque génération et de démarrer la génération après avoir effectué un saut dans la séquence. Il faut néanmoins garantir que la séquence utilisée soient suffisamment longue pour éviter les chevauchement (choix du saut réalisé au démarrage).

Le cluster de PC utilisé est un cluster équipé de 8 nœuds, incluant chacun un biprocesseur AthlonXP 1800+, associé à 1 Go de mémoire vive. Les cartes réseaux sont des cartes Gigabits fournissant une bande passante 1Gb/s. Le système d'exploitation utilisé est un RedHat 8.0.

[1] API disponible sur http://www.gnu.org

4.2 Analyse des performances

La comparaison des performances est réalisée entre la version mono-processeur, qui sert de référentiel et les stratégies synchrone forte et synchrone faible. La stratégie asynchrone ne fait pas partie de ce comparatif. En effet, la charge due au backtracking s'avère trop coûteuse dans notre application et élimine d'entrée cette technique des choix possible d'implémentation.

La figure 2 nous montre deux choses. La stratégie synchrone faible est plus coûteuse en temps total de simulation que la version mono-processeur ce qui était prévisible. Le surcoût observé est tout simplement l'œuvre des communications entre le coordinateur et les simulateurs. Il faut noter tout de même que ce coût est relativement faible ce qui nous autorise à dire que cette stratégie est intéressante dans le cas où les données attachées aux particules et aux sous-espaces manipulés par le simulateur ont une taille importante.

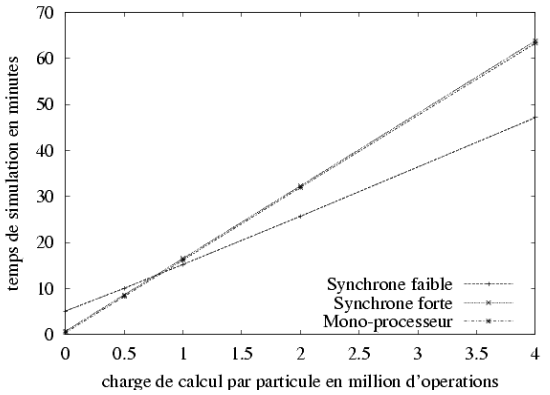

Figure 2 Variation du temps de simulation en fonction de la charge de calcul par particules (30.10^3 particules simulées). L'approche synchrone faible est plus efficace à partir d'une charge d'environs 0,7 millions d'opérations par particules.

Le deuxième constant concerne l'évolution du temps de simulation dans le cas synchrone faible. Si la charge de calcul est négligeable voire nulle, il existe néanmoins un coût lié à la communication et à la synchronisation qui est incompressible. Ce qui implique que cette stratégie est moins bonne que notre stratégie de référence (mono-processeur) pour des charges de calculs inférieurs à 0.7 million d'opérations. Au delà de cette valeur seuil, la stratégie synchrone faible est meilleure. Le constat reste

vrai quelque soit le nombre de particule (voir figure 3). On observe cependant une variation de ce seuil : plus le nombre de particules est élevé, plus la valeur de ce seuil est élevé.

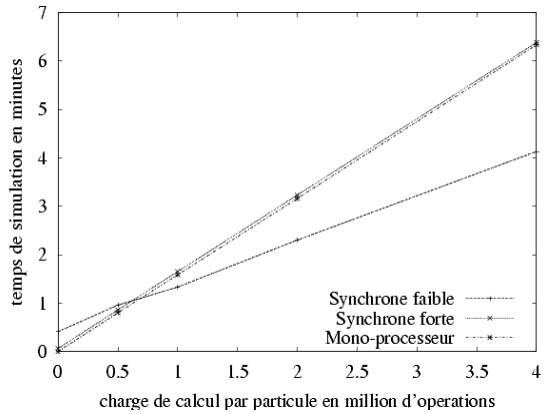

Figure 3 Variation du temps de simulation en fonction de la charge de calcul par particules (3.10^3 particules simulées). L'approche synchrone faible est plus efficace à partir d'une charge d'environs 0,7 millions d'opérations par particules.

La figure 4 représente la variation du temps total de simulation en fonction de la charge de calcul par particule dans le cas synchrone faible en faisant varier le nombre de simulateur. Faire varier le nombre de simulateur consiste tout simplement à diviser l'espace en un nombre plus élevé de sous-espace. Le passage de 3 à 6 diminue sensiblement le temps de simulation. Mais on observe cette tendance qu'à partir d'un certain seuil de charge. Si on augmente le nombre de simulateurs (dans notre exemple à 9), on s'aperçoit que ce seuil s'éloigne rapidement. L'explication est simple. Si on mesure le nombre total de messages échangés entre les simulateurs et entre coordinateur et simulateurs, on observe une augmentation linéaire. En effet, plus le nombre de sous-espaces est élevé plus le nombre de messages est élevé.

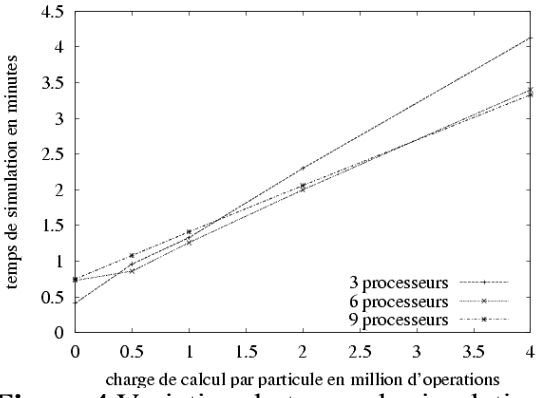

Figure 4 Variation du temps de simulation en fonction de la charge de calcul suivant l'approche asynchrone faible en utilisant 3,6 et 9 processeurs.

5 CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons comparé trois techniques de synchronisation pour des simulations à évènements discrets avec pour objectif de choisir la mieux adaptée à notre problème.

Nous avons réalisé un simulateur de mouvement brownien distribué selon différentes stratégies (synchrone forte et faible, et asynchrone).

Les résultats que nous avons obtenus nous permettent de penser que l'utilisation de la technique synchrone faible semble être la meilleure stratégie. En effet, cette stratégie est la seule de nos deux approches permettant de gagner en temps de simulation par rapport à la technique mono-processeur. Cependant, il faut prendre en compte, que la technique synchrone faible ne devient intéressante qu'à partir d'une certaine charge de calcul par particule. L'approche synchrone forte n'a comme principal intérêt que l'augmentation du nombre de particules qu'il est possible de simuler sans perdre de temps par rapport à la technique mono-processeur et la simplicité de son algorithme.

Dans notre futur système proie-prédateur, les prédateurs réalisent un grand nombre de calculs en interne pour leur comportement (perception, digestion ...), mais aussi pour calculer leur mouvement dans un espace 3D (calculs de trajectoires, intersections ...). Ces coûts nous font penser que la technique pessimiste est la meilleure pour notre modèle proie-prédateur.

Lors du passage du modèle de particules à un modèle d'agent situés (Ferber, 1995), nous serons confrontés à des problèmes liés au mécanisme de perception des prédateurs. En effet, la perception correspond à une interaction distante pouvant se produire entre deux individus localisés sur des simulateurs différents. Dans l'exemple du mouvement brownien, nous n'avons pas été confronté au problème d'équilibrage de charge, car la distribution des particules a tendance à rester uniforme et on sait que la charge de calcul est due au nombre de particule par simulateur. En revanche, dans le cas d'agent situés dotés d'un comportement, la répartition des individus dans l'espace peut ne plus être homogène. Dans ce cas, le modèle de distribution devra être capable de gérer le répartition de charge. Il existe de nombreux travaux traitant de ce problème (Miguet 1995) (Fonlupt 1998). Il reste néanmoins à l'adapter à notre système.

BIBLIOGRAPHIE

(Chandy, 1978) Chandy K. M., Misra J., "Distributed Simulation : A Case Study in Design and Verification of Distributed Programs.", IEEE Transaction on Sofware Engineering, SE-5, No. 5, p. 440-452 (1978).

(Duboz, 2003) Duboz R., Ramat É., Preux P., Amblard F. et Deffuant G. "Utiliser les modèles individus-centrés comme laboratoires virtuels pour identifier les paramètres d'un modèle agrégé.", MOSIM'03, p. 353-357 (2003).

(Duboz, 2001) Duboz R., Ramat É. et Preux P., "Towards a coupling of continuous and discrete formalisms in ecological modelling - Influences of the choice of algorithms on results", ESS01, 13th European Simulation Symposium, p. 481-487 (2001).

(Ferber, 1995) Ferber J. "Les systèmes multi-agents : Vers une intelligence collective", Editions InterEditions (1995).

(L'Ecuyer, 1998) L'Ecuyer P. "Random Number Generation", Chapter 4 of the Handbook on Simulation, Jerry Banks Ed., Wiley, p. 93-137 (1998).

(L'Ecuyer, 2002) L'Ecuyer P., Simard R., Chen E. J. et Kelton W. D., "An Objected-Oriented Random-Number Package with Many Long Streams and Substreams", Operations Research, Vol. 50, No. 6, p. 1073-1075 (2002).

(Fick, 1855) Fick A. "Über diffusion", Annalen der Physik und Chemie, Vol. 94, p. 59-86 (1885).

(Fonlupt, 1998) Fonlupt C., Marquet P., et Dekeyser J., "Data-parallel load balancing strategies", Computing 24, p. 1665-1684 (1998).

(Fujimoto 1999) Fujimoto R., "Parallel and distributed simulation", Winter Simulation Conference, 122-131 (1999).

(Jefferson, 1985) Jefferson D. R. "Virtual time", ACM Transactions on Programming Languages and Sytems, Vol. 7, No. 3, p. 404-425 (1985).

(Lamport, 1978) Lamport L., "Time, clocks and the ordering of events in a distributed system.", Communications of the Association of the Computing Machinery, Vol. 21, No. 7, p.558-565 (1978).

(Leroudier, 1980) Leroudier J., "La simulation à événements discrets", Editions Hommes et Techniques (1980).

(Miguet, 1995) Miguet S. et Pierson J. M. "Dynamic load balancing in a parallel particle simulation", In High Performance Computing Symposium, pages 420-431, Montreal, Canada (1995).

(Matsumoto, 1998) Matsumoto T. et Nishimura T., "Mersene Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, p.3-30 (1998).

(Misra, 1986) Misra J., "Distributed discrete-event simulation.", Computing surveys, Vol. 18, No. 1, p.39-65 (1986).

(Ramat, 2003) Ramat E. et Preux P. "Virtual laboratory environment (VLE): a software environment oriented agent and object for modeling and simulation of complex systems", Simulation Modelling Practice and Theory, 11, p.45-55 (2003).

Rakotoarisoa, (1991) Rakotoarisoa H. et Mussi P., "Parseval : Parallélisation sur réseaux

de transputers de simulations pour l'évaluation de performances.", Technical Report 131, INRIA (1991).

Zeigler (2000) Zeigler B., Kim D. et Praehofer H. "Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic system", academic press (2000).