Using Interest Management to Improve Load Balancing in Distributed Simulations

Felipe C. Bacelar, Carlos J. P. de Lucena
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, Brasil
{fbacelar, lucena}@inf.puc-rio.br

Pierre Bommel
CIRAD
Unidade GREEN
Montpellier, France
bommel@cirad.fr

***Abstract*— This paper presents an approach to distribute an agent-based simulation over a network of computers. The developed work aims at improving the load balancing of the simulation distribution. In order to reach such objective, we propose to use an Interest Management technique presented by Brian Logan and Georgios Theodoropoulos who proposed to distribute a simulation by dynamically partitioning the environment according to the Interest of the agents. In order to assess its efficiency, we have re-implemented this model using the distribution mechanisms provided by some of the main multi-agent system platforms.**

Keywords—Load Balancing; Interest Management; Spheres of Influence; Distributed Simulation; Multi-agent Systems

I. INTRODUÇÃO

Agentes podem ser entendidos como entidades autônomas localizadas em um ambiente. Eles são capazes de se comunicar entre si e interagir com seu ambiente [13]. Sistemas multiagentes podem ser compostos por conjuntos de agentes que interagem com o propósito de resolver um problema ou alcançar um objetivo. Em certos casos, os agentes colaboram entre si para atingir uma meta em comum. Em outros, os objetivos dos agentes podem ser opostos [11].

O comportamento autônomo dos agentes e sua capacidade de tomada de decisão são fatores que tornam a utilização de sistemas multiagentes muito atrativa para o desenvolvimento de simulações.

Em nosso contexto é possível definir simulação como uma representação do comportamento de um sistema, real ou imaginário, através do tempo [5]. O emprego de simulações tem sido uma forma muito eficaz de estudar e analisar problemas reais. A possibilidade de modelar diferentes cenários para o mesmo problema e reproduzi-los diversas vezes permite a identificação de situações incomuns e comportamentos inesperados.

Como os agentes costumam ser unidades complexas de software, manter um número elevado deles em uma simulação pode ser muito custoso. Simulações de larga escala, com centenas ou milhares de agentes, tendem a ter seu desempenho comprometido.

Uma solução para este problema é distribuir a simulação entre diversos computadores. Contudo, simplesmente dividir a simulação em partes e executar cada uma delas em uma máquina diferente pode não ser uma boa solução. Entre outros problemas, é preciso considerar o balanceamento de carga. Balanceamento de carga é uma técnica que visa dividir coerentemente a carga de trabalho entre computadores de uma rede, maximizando o desempenho e evitando sobrecarga.

O presente trabalho, referente a uma pesquisa em andamento, propõe uma abordagem para melhorar o balanceamento de carga de uma simulação distribuída explorando o conceito de esferas de influência como estratégia de gerenciamento de interesse dinâmico.

II. GERENCIAMENTO DE INTERESSE

Em simulações distribuídas de larga escala a comunicação do tipo *broadcast* deve ser evitada para reduzir o custo de comunicação entre as máquinas da rede. Uma proposta para resolver este problema é conhecida como Gerenciamento de Interesse [14]. O princípio deste modelo é baseado na idéia de que as entidades raramente usam todas as informações disponíveis, mas elas podem manifestar interesse em apenas um subconjunto de informações que são relevante para elas. Um agente deve ser provido de acesso apenas ao conjunto de elementos com os quais poderá interagir (ler/atualizar). Contudo, esse conjunto pode mudar com o tempo. Ao se mover pelo ambiente, um agente pode desviar de um obstáculo e passar a ter um campo de visão maior, aumentando o número de variáveis que poderá ler, por exemplo. O mecanismo de Gerenciamento de Interesse deve ser capaz de se adaptar a estas mudanças [6], [7], [9], [12].

A. *Esferas de Influência*

Com o intuito de solucionar o problema do gerenciamento de interesse dinâmico, Logan e Theodoropoulos criaram o conceito de Esferas de Influência [6], [7], [9], [12]. O modelo da simulação é divido em conjuntos de Processos Lógicos (LP na sigla em inglês) concorrentes, cada um mantendo uma parte disjunta do espaço do sistema. Os LPs podem ser processos lógicos de agentes ou processos lógicos de ambiente [6], [7], [9]. Todos os LPs de uma simulação geram um número limitado de tipos de eventos. Diferentes tipos de eventos geram diferentes efeitos sobre o estado da simulação.

A esfera de influência de um evento pode ser entendida como “o conjunto de variáveis lidas ou atualizadas em decorrência do evento”. A esfera de influência de um LP em

um dado intervalo de tempo é a união das esferas de influência dos eventos gerados por esse LP no intervalo. A interseção das esferas de influência dos LPs gera uma ordenação parcial das variáveis de estado em que os primeiros elementos são os acessados pela maior quantidade de LPs e os últimos são acessados pela menor quantidade [9]. Esta ordenação parcial pode ser utilizada para decompor o estado da simulação de forma a manter um determinado agente no mesmo nó da plataforma distribuída em que se encontram os elementos com os quais ele mais interage na simulação. Isto garante uma redução no custo de comunicação entre as máquinas.

Em [6], [7], [9] e [12] o estado da simulação pode ser decomposto usando um novo conjunto de processos lógicos chamados *Communication Logical Processes* (CLP). Cada CLP armazenará um subconjunto do estado da simulação. Inicialmente, toda a simulação é de responsabilidade de um único CLP. Com o progresso da simulação, o CLP realiza o cálculo aproximado das esferas de influência dos agentes. Se o CLP ficar sobrecarregado (atingir um determinado limite de tráfego de rede, por exemplo), um novo CLP é criado e um subconjunto disjunto do estado da simulação é atribuído a ele, normalmente os últimos elementos da ordenação parcial gerada pela interseção das esferas de influência dos LPs. O processo é então repetido para o novo CLP, monitorando a carga e criando novos CLPs caso necessário.

Segundo Logan e Theodoropoulos, o custo computacional de calcular as esferas de influência é muito alto. Qualquer implementação eficiente pode apenas aproximar o resultado ideal [9]. A abordagem apresentada pelos autores foi implementada em um framework chamado PDES-MAS [10].

III. Abordagem Proposta

No framework PDES-MAS os CLPs são organizados naturalmente em uma árvore e cada CLP possui informação de roteamento que indica quais tipos de eventos são relevantes para seu CLP pai e para seus filhos (LPs e outros CLPs) [9], [10].

Para simplificar a implementação e adaptação a qualquer plataforma distribuída de sistemas multiagentes, o trabalho sugere evitar o uso de CLPs.

Muitas plataformas de sistemas multiagentes podem ser distribuídas entre máquinas de uma rede local ou até mesmo pela Internet. Em geral, a plataforma mantém informações sobre todos os nós conectados e é capaz de gerenciar a comunicação entre eles.

O presente trabalho propõe utilizar o conceito de esferas de influência para decompor o estado de uma simulação e distribuí-la entre os nós de uma plataforma que forneça recursos de distribuição. Cada nó assumirá o papel de um CLP passando a conter uma parcela do estado da simulação. No entanto, o nó não é responsável por guardar informações de roteamento, deixando para a plataforma o gerenciamento da comunicação.

Com a finalidade de evitar gargalos na simulação, são estabelecidos limites que, uma vez atingidos, classificam um nó da plataforma como sobrecarregado. Alguns critérios úteis são: uso de CPU, uso de memória e tráfego de rede.

A simulação é iniciada no hospedeiro principal da plataforma e quando um dos limites estabelecidos é alcançado, o processo de distribuição é disparado. Utilizando a ordenação obtida através das esferas de influência dos LPs, uma parte do estado da simulação é migrada do hospedeiro principal para um nó disponível na plataforma. Se o nó em questão ficar sobrecarregado, a migração continua para o próximo nó disponível. O processo é interrompido quando não há mais nós sobrecarregados, incluindo o hospedeiro principal, ou quando não há mais máquinas disponíveis na plataforma.

Durante a execução da simulação, se qualquer nó da plataforma ficar sobrecarregado, o processo de migração é disparado novamente.

IV. O Modelo da Simulação

Para ilustrar o método proposto, está em desenvolvimento uma simulação simples representando um cenário de derramamento de petróleo. Este tipo de desastre é comum e pode causar diversos tipos de impactos ao meio-ambiente. Entre os mais frenquentes é possível listar: envenenamento de peixes, bloqueio da luz solar impedindo a fotossíntese das algas e morte por afogamento de pássaros que tiveram suas penas cobertas de petróleo (ficando incapazes de voar).

O modelo da simulação é composto por um pequeno conjunto de classes de agentes. Há uma classe para representar os barcos recolhedores, uma classe para manchas de petróleo e uma classe representando células de mar. O ambiente é apresentado em um *grid* composto por células de mar.

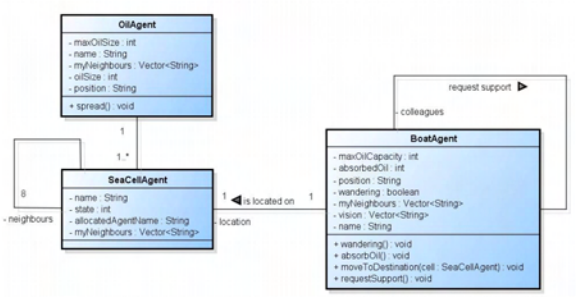

Fig. 1. Diagrama de classes do modelo de derramamento de petróleo

Como pode ser visto no diagrama apresentado na figura 1, o agente de mancha possui apenas o comportamento de se espalhar. O agente barco tem o comportamento de vaguear pelo ambiente até que uma mancha de petróleo entre em seu campo de visão. Ao avistar uma mancha, o agente barco percorre o caminho até ela e a absorve assim que a alcança. Se a capacidade do barco não for suficiente para absorver todo o petróleo, outro barco recolhedor é solicitado.

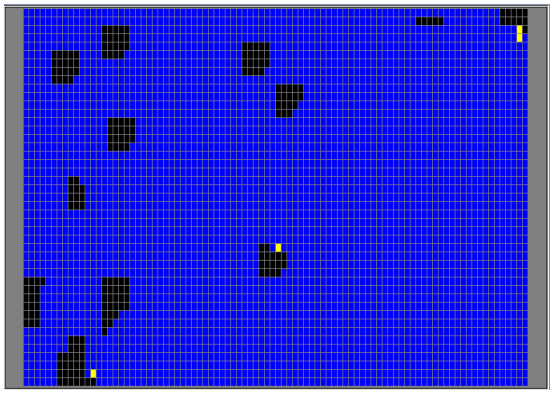

Fig. 2. Execução da simulação em desenvolvimento

A figura 2 apresenta a simulação em desenvolvimento. As células azuis do *grid* representam as células do mar. As células em preto, por sua vez, são representações das manchas de petróleo. Por último, as células em amarelo são os barcos recolhedores.

Para exibir graficamente a interação entre os agentes foi utilizado o *framework* "Agent. GUI". O *framework* provê uma interface gráfica pré-definida que pode ser adaptada às necessidades do desenvolvedor. É completamente desenvolvido em Java e baseado no *framework* JADE [4].

JADE é uma sigla para Java Agent DEvelopment Framework. Trata-se de um *framework* desenvolvido em Java que fornece recursos para implementação de sistemas multiagentes [1]. Sua plataforma de agentes é usada no projeto desenvolvido no presente trabalho, pois possui todos os recursos de distribuição necessários [1], [2], [3].

V. Considerações Finais

O presente trabalho é referente a um projeto de pesquisa em andamento que visa melhorar o balanceamento de carga ao distribuir simulações. Para isto, propõe a utilização de uma técnica de gerenciamento de interesse que busca manter cada agente no mesmo nó da rede que se encontram as partes da simulação com as quais ele mais interage.

Uma implementação de uma simulação em uma plataforma distribuída utilizando o método proposto está em desenvolvimento. Outras técnicas de gerenciamento de interesse estão sendo analisadas para eventualmente atuarem em conjunto com o conceito de esferas de influência na solução final do projeto. Além disto, uma estratégia para selecionar o melhor nó disponível da plataforma (máquina mais potente ou com melhor tempo de resposta) deve ser discutida.

References

[1] Bellifemine, F.; Bergenti, F.; Caire, G.; Poggi, A., JADE - A Java Agent Development Framework. In Proceedings of Multi-Agent Programming, p. 125-147, 2005.

[2] Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G. JADE Programmer's Guide. Available at: http://jade.tilab.com/doc/programmersguide.pdf Access: 10 dec. 2012

[3] Caire, G.; Rimassa, G.; Bellifemine, F. JADE: a versatile run-time for distributed applications on mobile terminals and networks. In Proceedings of SMC (2), p. 1882-1888, 2004.

[4] Derksen, C.; Branki, C.; Unland, R. Agent.GUI: A Multi-agent Based Simulation Framework. In Proceedings of FedCSIS, p. 623-630, 2011.

[5] Fujimoto, Richard M. Parallel and Distributed Simulation Systems. New York: Wiley Interscience, 2000. 300 p.

[6] Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Distributed Simulation of MAS. In Proceedings of the Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation (MAMABS'04), p. 21-30, 2004.

[7] Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Modelling Environments for Distributed Simulation. In 1st International Workshop on Environments for Multi-Agent Systems (E4MAS), in conjunction with the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04), p. 150-167, 2004.

[8] Lees, M.; Logan, B.; Theodoropoulos, G. 2007. Distributed Simulation of Agent-based Systems in HLA, ACM Transactions on Modelling and Computer Simulation, Vol. 17, 3, Available at: http://doi.acm.org/10.1145/1243991.1243992) ISSN: 1049-3301.

[9] Logan, B.; Theodoropoulos, G. The distributed simulation of multi-agent systems. In Proceedings of the IEEE 89(2), p. 174-186, 2001.

[10] Oguara, T.; Chen, D.; Theodoropoulos, G.; Logan, B., and Less, M. An Adaptive Load Management Mechanism for Distributed Simula-tion of Multi-agent Systems. Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applica-tions, pp. 179–186. October 2005.

[11] Parunak, H. V. D.; Brueckner, S.; Fleischer, M. et Odell, J., 2003. A Design Taxonomy of Multi-Agent Interactions. Pages 123-137 of : Paolo Giorgini, Jörg P. Müller, James Odell (ed), Agent-Oriented Software Engineering IV : 4th International Workshop, AOSE 2003, Melbourne, Australia, July 15, 2003, Revised Papers. Lecture notes in computer science LNCS, vol. 2935. Springer.

[12] Theodoropoulos, G; Logan, B. An Approach to Interest Management and Dynamic Load Balancing in Distributed Simulation. In Proceedings of the 2001 European Simulation Interoperability Workshop, p. 565-571, 2001.

[13] Wooldridge, M.; Jennings, N.R., Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2), p. 115-152, 199

[14] Morse, K. L.; Interest Management in Large-Scale Distributed Simulation; Volume 96, Issue 27 of Technical report (University of California, Irvine. Dept. of Information and Computer Science), 1996