IMPLEMENTAÇÃO E AVALIAÇÃO DE UM MODELO DE MERCADO IMPERFEITO EM COMPUTAÇÃO PARALELA

Ana Luísa de A. Santos, Diego Carvalho, Felipe G. França
DEL/UFRJ, COPPE/UFRJ
E-mail: analuisa@lps.ufrj.br

RESUMO

Este trabalho apresenta a implementação de um algoritmo paralelo de um modelo econofísico de um mercado imperfeito formado por agentes que negociam diferentes produtos entre si, por meio de trocas monetárias baseadas em informações locais. A flutuação do valor dos produtos individualmente, associado à condição de lucro máximo nas transações, leva à flutuações na riqueza individual dos agentes, que são análogas às encontradas nos mercados financeiros.

A aplicação de processamento paralelo neste problema implicou na necessidade de mecanismos de controle que tornaram o algoritmo mais complexo. Contudo, a paralelização diminuiu o tempo total de execução do algoritmo, por meio da distribuição do processamento entre processadores interconectados, ou seja, a implementação proposta apresentou *Speedup* significativo com relação à implementação sequencial.

1. INTRODUÇÃO

Nos últimos anos, a demanda por capacidade de processamento para a solução de problemas complexos computacionalmente vêm impulsionando o aumento do poder de processamento dos computadores. Contudo, o alto custo de desenvolvimento e aquisição de *hardware* mais poderosos tem estimulado a utilização de soluções computacionais alternativas, tais como o processamento paralelo. O processamento paralelo objetiva a obtenção de ganhos de performance na execução de um algoritmo por meio da distribuição do seu processamento para diferentes processadores executarem simultaneamente. Estes processadores podem estar presentes em diferentes computadores interligados por redes, ou em um mesmo computador com múltiplos processadores.

Contudo, a implementação de algoritmos paralelos também apresenta desvantagens, que devem ser equacionadas de acordo com o problema em questão. De forma geral, enquanto algoritmos paralelos proporcionam aumento do poder computacional, por outro lado apresentam maior complexidade de implementação e custos de comunicação mais altos.

Atualmente, os clusters de computadores, conjunto de estações de trabalho interligadas por redes de comunicação de alta velocidade, são cada vez mais lugar comum. A ploriferação de clusters, aliada ao desenvolvimento de ferramentas de apoio à implementação de algoritmos paralelos, sejam bibliotecas ou linguagens de programação específicas, têm estimulado enormemente a paralelização de algoritmos.

Recentemente, a utilização da computação paralela na solução de problemas científicos e de engenharia vem sendo cada vez mais difundida. De fato, a arquitetura paralela está presente nos computadores com maior capacidade de processamento existentes atualmente, denominados supercomputadores [1], que chegam a processar 40TFLOPS. Suas principais aplicações ocorrem nas áreas militar, meteorológica, banco de dados, telecomunicações, automobilística, aeroespacial, farmacêutica, eletrônica, dentre outras. No entanto, apesar das aplicações existentes, a computação paralela apresenta potencial de agregar bastante valor a diversas áreas do conhecimento em que sua possibilidade de aplicação é praticamente desconhecida.

2. DEFINIÇÃO DO PROBLEMA

A Econofísica [2] consiste em uma área do conhecimento ainda incipiente, em que são aplicados métodos, modelos e idéias provenientes da Física Estatística e de Sistemas Complexos, na análise de fenômenos econômicos. O modelo de um mercado imperfeito abordado neste trabalho se trata de um modelo econofísico.

No modelo econofísico de um mercado imperfeito proposto por Donangelo et al [3], o mercado é formado por agentes que negociam diferentes tipos de produtos entre si por meio de trocas monetárias, baseando-se em informações locais. A flutuação do valor dos produtos individualmente, associado à condição de lucro máximo nas transações, leva à flutuações na riqueza individual dos agentes, que são análogas às encontradas nos mercados financeiros.

Neste modelo de mercado, a riqueza ou a falência dos agentes ocorre em função da demanda individual destes agentes por maior lucro nas transações. Isto significa que, dado 3 agentes distintos pertencentes a um mercado: A, B e C; o agente A visa encontrar o agente B disposto a vender-lhe um produto à custo baixo, a fim de que posteriormente o agente A o

revenda ao agente C por um preço mais elevado, auferindo lucro nestas negociações. Donangelo et al. [3], denominam Fat Cat (FC) este modelo de um mercado não-equilibrado, já que se baseia na necessidade dos agentes por maximizar o lucro, gerando grandes flutuações no mercado (*fat tails*).

O modelo em questão [3] se assemelha a um mercado de antiguidades, já que se trata de um mercado fechado, onde os agentes se encontram, consultam o portfolio de produtos uns dos outros, e decidem comprar ou vender entre si a mercadoria disponível que lhes proporcione lucro máximo segundo suas percepções individuais do mercado.

O funcionamento do algoritmo original do modelo de mercado imperfeito [3] consiste em um Laço de iterações representando instantes de tempo. Em cada iteração ocorre um encontro aleatório de um par de agentes, seguido da negociação entre estes agentes. A negociação pode ser bem sucedida, gerando uma transação de compra e venda; ou não, fazendo os agentes alterarem suas percepções de preços do mercado (o agente vendedor diminui seu preço do produto negociado, e o comprador passa a se dispor a pagar um preço mais alto pelo produto).

O modelo de mercado imperfeito [3] tende a equilibrar grandes diferenças de preço, mas induz a diferenças de preços quando compradores e vendedores entram em acordo com relação ao preço do produto mais negociado. Este desequilíbrio é essencial para induzir dinâmicas em um modelo como este, onde todos os agentes agem baseados numa estratégia comum. Sem este fator de desequilíbrio, o mercado congelaria em um estado onde todos os agentes concordariam sobre o preço de todos os produtos.

Dado aos ajustes na percepção de preço dos produtos resultantes de negociações mal sucedidas, o preço dos produtos nunca alcança o equilíbrio, e diferentes agentes podem ter diferentes percepções de preço de um mesmo produto. Assim, o gráfico com a amostragem da percepção de preço de um produto por 2 agentes no tempo deve apresentar flutuações persistentes, o que demonstra a possibilidade de um agente auferir lucro.

O funcionamento do modelo proposto por Donangelo et al [3] demonstra que a distribuição da riqueza se auto-organiza segundo um padrão dinamicamente estável. Assim, são esperadas no modelo flutuações consideráveis da Riqueza de um agente.

3. MODELAGEM DO ALGORITMO PARALELO

Segundo Donangelo et al [3], o sorteio de encontros entre pares de agentes ocorre de forma ordenada no tempo. O estado de um agente, isto é, sua quantidade em estoque de cada produto e suas percepções de preço, depende do resultado de todos os encontros anteriores. Contudo, existe uma dependência parcial na ordem do processamento de diferente encontros, ou seja, encontros que envolvam agentes distintos podem ser computados fora de ordem. Em outras palavras, dado o sorteio de um par de agentes **x** e **y**, todos os pares de agentes sorteados em seguida que não contenham **x** ou **y** podem ser processados de forma independente, ou seja, mesmo que o processamento do encontro entre **x** e **y** ainda não tenha sido concluído. Assim, se, enquanto o processamento do encontro entre **x** e **y** não houver terminado algum novo encontro sorteado contiver **x** e/ou **y**, este encontro não poderá ser processado, devendo permanecer em estado de espera para ser processado.

Este comportamento, a que iremos nos referir como dependência parcial, pode ser explicado da seguinte forma: o sorteio dos agentes corresponde a um encontro aleatório de dois indivíduos, agentes do mercado, que pretendem negociar produtos entre si, baseados em suas percepções de preço dos produtos no mercado, com o intuito de auferir lucros. Estes indivíduos decidem por comprar ou vender produtos com base em suas percepções de preço dos produtos no mercado e de estoques individuais, no momento em que se encontram. Desta forma, para que vários encontros possam ser processados de forma paralela, a consistência das informações dos agentes no momento em que se processam os encontros deve ser mantida.

O modelo de mercado imperfeito assume que os encontros entre pares de agentes ocorrem seqüencialmente no tempo. Assim, a implementação do algoritmo distribuído deste modelo deve manter a consistência da evolução do mercado com relação à evolução seqüencial.

O algoritmo distribuído proposto é adequado para análise de um mercado com grande número de agentes. Neste mercado, a probabilidade de pares de agentes sorteados em seqüência possuirem agentes iguais é bastante pequena. Este comportamento é, então, utilizado para a distribuição do processamento de encontros de pares de agentes distintos que tenham sido sorteados sequencialmente. Para manter a consistência temporal dos encontros entre agentes do mercado, o processador Mestre deve manter o controle de seqüenciamento temporal dos resultados do processamento de encontros pelos Escravos, e levar em consideração a dependência parcial do processamento das informações dos agentes.

Neste trabalho é proposta uma implementação paralela do algoritmo de mercado imperfeito baseada em uma arquitetura do tipo Mestre-Escravo para a divisão do processamento entre diferentes computadores. Ao ser executado, o algoritmo paralelo atribui a função de Mestre para um dos processadores, e de Escravos para os demais. Esta arquitetura foi escolhida devido a necessidade de se manter consistentes as informações dos agentes no momento em que são processados, recém denominada dependência parcial. As funções de controle da dependência parcial do sorteio dos agentes,

bem como a geração de sorteios e distribuição de forma balanceada do processamento é centralizada, e atribuída ao Mestre. Os Escravos, por sua vez, são responsáveis por processar as tarefas recebidas do Mestre, e retornar seus resultados. A comunicação entre o Mestre e os Escravos se ocorreu através de troca de mensagens. As mensagens contém blocos (conjuntos) de pacotes, onde cada pacote representa uma instrução independente.

3.1. Funções do Mestre

O processador Mestre tem como função principal controlar a evolução temporal do algoritmo distribuído, através das seguintes atribuições:

- Sortear encontros entre pares de agentes;
- Decidir qual Escravo deve processar negociações entre cada par de agentes;
- Distribuir Tarefas (pares de agentes, e eventuais informações adicionais) para os Escravos apropriados processarem suas negociações, de forma balanceada;
- Receber resultados das negociações processadas pelos Escravos;
- Garantir consistência da dependência parcial do processamento dos encontros;
- Manter o controle dos Escravos que possuem informações autalizadas sobre cada agente;
- Manter o controle e armazenar informações atualizadas de todos os agentes do mercado.

3.2. Funções dos Escravos

Os Escravos são responsáveis por processar as negociações entre pares de agentes recebidas do Mestre, e posteriormente retornar os resultados obtidos para o próprio Mestre.

4. IMPLEMENTAÇÕES

Na implementação do algoritmo sequencial foi utilizada a linguagem de programação C, sob o Sistema Operacional Linux (Red Hat versão 9.0). O algoritmo foi executado em uma estação de trabalho com 2 processadores (Intel Xeon Dual de 2.4GHz), e 4GB de memória.

Na implementação do algoritmo paralelo foi utilizada a linguagem de programação C com o suporte do MPI-2 [4][5][6], no ambiente LAM/MPI [7], usando o Sistema Operacional Linux (Red Hat versão 9.0). O algoritmo foi executado em um cluster de 4 processadores distribuídos em 2 estações de trabalho ligadas via rede local (Ethernet 100Mbps), onde cada estação de trabalho possuía 2 processadores (Intel Xeon Dual de 2.4GHz), e 4GB de memória.

Cabe ressaltar que o cluster de computadores interconectados por rede local, e a natureza do algoritmo paralelo proposto pode ser classificado, segundo a nomeclatura [8] de computação paralela, como MIMD, medium-grained e com memória distribuída.

4. METODOLOGIA DE MEDIDA DE EFICIÊNCIA DE ALGORITMOS PARALELOS

Os fatores mais importantes na determinação da eficiência da aplicação de computação paralela a um determinado problema são: em primeiro lugar, o limite de implementação de paralelismo em programas sequenciais; em segundo lugar, o custo de comunicação entre os nós de paralelismo. Tais limitações dificultam a obtenção de ganhos de velocidade de processamento das aplicações de programação paralela.

A medição do ganho de velocidade de processamento de aplicações paralelas pode ser feita através da Lei de Amdahl [9], proposta da seguinte maneira:

Dadas as variáveis:

- *Speedup*: Ganho de velocidade com o processamento paralelo;
- $Fraction_{enhanced}$: porção do processamento do programa sequencial implementada de forma paralela;
- $Speedup_{enhanced}$: número de processadores utilizados no processamento paralelo;

A Lei de Amdahl é enunciada como:

$$Speedup = \frac{1}{\frac{Fraction_{enhanced}}{Speedup_{enhanced}} + (1 - Fracion_{enhanced})}$$

5. RESULTADOS

Os resultados gerados pela implementação sequencial foram calculados para os mesmos parâmetros do mercado utilizado por Donangelo et al [3], com o objetivo de validar a implementação sequencial.

Parâmetros do mercado simulado:

- Num. de agentes = 100;
- Num. inicial de produtos por agente = 100;
- Tipos de produtos no mercado = 100;
- Quantidade inicial de dinheiro por agente = 500;
- Número de iterações: até 3.000.000;

Na Figura 1 são observadas as flutuações da percepção do preço de um mesmo produto por 2 agentes, ao longo do tempo. A Figura 2, por sua vez, apresenta a flutuação da Riqueza de um agente. Estas flutuações são persistentes, e se mostram de acordo com os resultados esperados pelo modelo proposto por Donangelo et al [3]

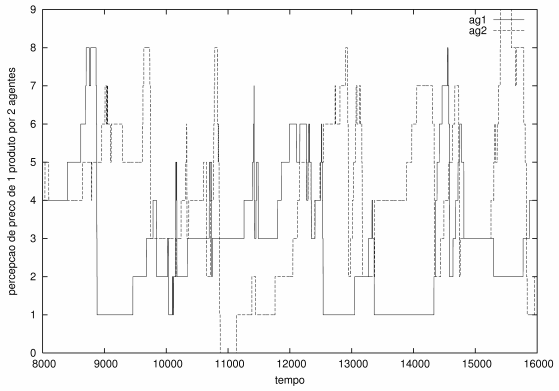

Fig. 1: Evolução da Percepção de Preço de 1 produto para 2 agentes (implementação seqüencial)

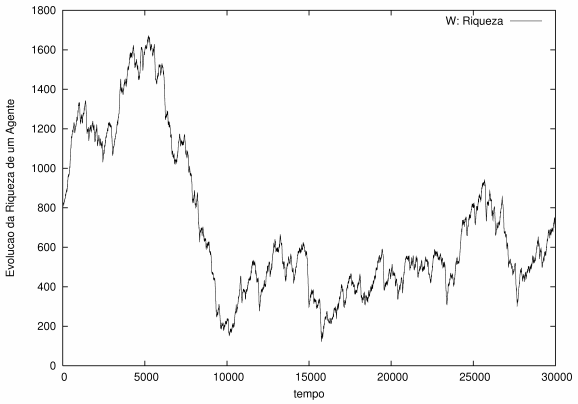

Fig. 2: Flutuação da Riqueza de um agente (implementação seqüencial).

A validação da implementação paralela se deu através da verificação passo a passo da atualização das variáveis do mercado, de acordo com a ordem temporal de eventuais dependências de encontros entre os agentes gerados.

Os resultados gerados pela implementação paralela foram calculados para um mercado imperfeito com variáveis de maior ordem de grandeza do que as das simulações geradas pelo algoritmo original. Isso se fez possível devido a que, segundo Donangelo et al [3], o comportamento do mercado reproduz uma lei de potência, ou seja, seu comportamento se mantém, independentemente da ordem de grandeza das variáveis do mercado.

Como resultado desta simulação, foi obtido o gráfico da Figura 3, onde são observadas as flutuações da Riqueza de um agente. Estas flutuações são persistentes, e se mostram de acordo com os resultados esperados pelo modelo proposto por Donangelo et al [3], o que demonstra consistência desta implementação do algoritmo paralelo do modelo.

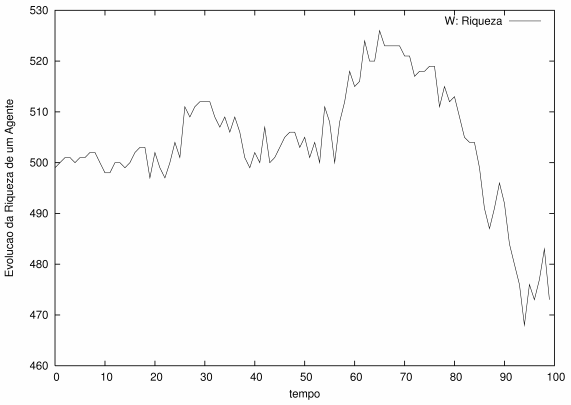

Fig. 3: Flutuação da Riqueza de um agente (implementação paralela).

Após a validação das implementações sequencial e paralela, se fez necessária a comparação do tempo de processamento de ambas através do cálculo do Speedup. Para isso, ambas as implementações são executadas para um mercado com as mesmas variáveis, com ordem de grandeza maior do que das variáveis originais do modelo. Esta comparação é possível, já que, segundo Donangelo et al [3], o comportamento do modelo proposto segue uma lei de potência, ou seja, suas características se mantém mesmo com a avaliação de um mercado com parâmetros de maior ordem de grandeza.

A avaliação do tempo de processamento das implementações sequencial e paralela, foi executada para um mercado com as seguintes variáveis:

- Num. de agentes = 10.000;
- Num. inicial de produtos por agente = 100;
- Tipos de produtos no mercado = 100;
- Quantidade inicial de dinheiro por agente = 500;
- Número de iterações: 1.000.000;
- Intervalo de tempo entre coleta de dados: 1000.

Na implementação paralela, outra variável crítica para a performance do mercado também foi determinada: tamanho da mensagem de transferência de trabalho do Mestre para os Escravos. Após alguns testes empíricos, foi escolhido um tamanho capaz de armazenar até no máximo 3 pacotes de informação.

A partir da execução de ambas as implementações, foram extraídos os 5 menores tempos de processamento para cada implementação, e os resultados mínimos e médios alcançados estão discriminados na tabela abaixo.

Tempo de Processamento (segundos)	Implementação Seqüencial	Implementação Paralela
Tempo Médio	42,348	26.575
Tempo Mínimo	41,244	20,363

O cálculo do *Speedup* foi realizado a partir do tempo mínimo de processamento, por este representar o melhor resultado obtido. O *Speedup* foi calculado pela relação entre o Tempo mínimo de processamento da implementação seqüencial e paralela, obtido como *Speedup* = 41,244 / 20,363 ˜ 2,025.

A partir da Lei de Amdahl [9] enunciada na sessão 4, para os 4 processadores que executaram a implementação paralela (*Speedupenhanced*), foi obtido que 67,5% do processamento do algoritmo original foi implementado de forma paralela (*fractionenhanced*).

6. CONCLUSÕES

Com relação às questões de ordem práticas do trabalho, os objetivos foram alcançados, a priori, já que os resultados das implementações sequencial e paralela reproduziram os resultados esperados do modelo de mercado imperfeito proposto por Donangelo et al [3]. Em primeiro lugar, a implementação sequencial reproduziu os mesmos resultados do modelo proposto: flutuações persistentes da evolução da riqueza de um agente, e da percepção de preço de um produto por 2 agentes. Em segundo lugar, a metodologia proposta na implementação paralela foi bem sucedida, já que manteve a consistência temporal do algoritmo sequencial original, apresentando também resultados de flutuações persistentes da riqueza de um agente.

Assim, com a validação das implementações sequencial e paralela, foram realizadas várias simulações de mercados com grande número de agentes para ambas as implementações, em que o algoritmo paralelo se mostrou consideravelmente mais eficiente. A implementação paralela apresentou *Speedup* de pouco mais de 2, que significa que o algoritmo paralelo foi executado praticamente na metade do tempo do sequencial, para as condições das simulações.

Como propostas de trabalhos futuros, podem ser realizadas mudanças mais profundas no próprio modelo; ou simplesmente análises estatísticas do comportamento do modelo paralelo, de forma a determinar o tamanho adequado de mensagens para o processamento de mercados com variáveis de maior ordem de grandeza. Certamente, o fator tamanho de mensagens constitui um gargalo intrínseco de qualquer modelo de processamento paralelo, principalmente em se tratando de clusters, onde o tráfego de rede pode influenciar o *Speedup* do algoritmo paralelo.

Cabe ressaltar também a possibilidade de aumentar o percentual de processamento paralelo no código, principalmente no que concerne as operações de coleta de resultados, que foram identificadas nas simulações como gargalos de processamento. De fato, foi observado que o modelo de implementação paralela proposto apresenta margem de aumento do nível de paralelismo, já que para o *Speedup* obtido nas condições das simulações, segundo a Lei de Amdahl [9], 67,5% do processamento do programa foi implementado de forma paralela.

7. REFERÊNCIAS

[1] http://www.top500.org/. Agosto 2004.

[2] YOUGUI, W., JINSHAN, W., ZENGRU, D., "Physics of Econophysics" http://www.unifr.ch/econophysics/. Março 2004.

[3] DONANGELO, R., HANSEN, A., SNEPPEN, K., SOUZA, S.R., "Modelling an Imperfekt Market", Physica A, n.283, pp. 469-478, 2000.

[4] GROPP, W., LUSK, E., SKJELLUM, A., Using MPI: Portable Parallel Programming with the Message Passing-Interface. MIT Press, 1994.

[5] http://www.mpi-forum.org/. Março 2004.

[6] "Message Passing Interface Forum. MPI: A Message Passing Interface". In Proc. of Supercomputing'93, p878–883. IEEE Computer Society Press, Novembro 1993.

[7] http://www.lam-mpi.org/. Março 2004.

[8] FLYNN, M.J., Very High-Speed Computing Systems, Proceeding of the IEEE, 54(12), December 1966, p1901-1909

[9] PATERSON, D., HENNESSY, J., Computer Architecture a Quantitative Approach. 2a Edição. Morgan Kaufmann Publishers, 1996.