SimSPD: Um simulador de sistema de processamento distribuído com foco em distribuição de carga

Régis Feitosa Brilhante[1] Guilherme Esmeraldo[2], Elaine Nagai[1]

[1]Faculdade Paraíso do Ceará (FAP)
Rua da Conceição, 1228 – Juazeiro do Norte – CE - Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - *campus* Crato
Rod. CE 292, Km 15 – Crato – CE – Brasil

regis.brilhante@fapce.com.br, guilhermealvaro@ifce.edu.br,
elaine.nagai@gmail.com

Resumo. *No desenvolvimento Sistemas de Processamento Distribuido (SPDs) vários aspectos devem ser considerados, como, por exemplo, suporte de hardware, paralelização e comunicação entre tarefas. Este trabalho propõe um simulador de SPDs para suportar o desenvolvimento de aplicações paralelas e avaliar seu desempenho sob diferentes condições de hardware e interconexão de redes.*

1. Introdução

Com a demanda de processamento de maior volume de informações em intervalos de tempo menores, as aplicações estão se tornando cada vez mais complexas. Antes a demanda por recursos computacionais era atendida apenas por supercomputadores, conhecidos como *mainframes*. Porém, com a redução dos preços e o aumento do desempenho dos computadores pessoais, surgiram os Sistemas de Processamento Distribuídos (SPDs). Um SPD pode ser definido como um conjunto de programas e dados distribuídos entre computadores independentes conectados por uma rede de comunicação [Tanenbaum 2007].

No desenvolvimento desses sistemas, existem muitas características que devem ser consideradas, como configuração do hardware, distribuição de carga, paralelização, concorrência, entre outros.

Nesse contexto, um software capaz de simular um ambiente distribuído permite estabelecer um ambiente controlado para exploração de características, estimar resultados, mitigar erros e riscos, visando o aumento da qualidade do produto final. O projeto Neko [Urbán et al. 2002], é um *framework* para a simulação de algoritmos distribuídos e utiliza troca de mensagens para comunicação entre processos. Nele os algoritmos podem ser simulados ou executados em uma rede real. Já o SimGrid [Casanova et al. 2008] é um kit de ferramentas que fornece mecanismos para a simulação de ambientes P2P, *Grids*, HPC e simulações reais de código escrito em MPI [MPI 2008]. A ferramenta GridSIm [Sulistio et al. 2008] permite a modelagem e simulação, tanto de computação distribuída, quanto de usuários, aplicações e recursos. No entanto, essas soluções focam no desenvolvimento e optimização de algoritmos

paralelos em *Grid*, não abordando temas como computação de alto desempenho e configuração de hardware nos hosts.

Este artigo apresenta um simulador de sistemas de processamento de aplicações paralelas, chamado de SimSPD, com foco em computação de alto desempenho. O simulador permite o desenvolvimento de novas aplicações paralelas, bem como a avaliação de seu desempenho de computação e comunicação sob diferentes configurações de hardware nos *hosts* e de rede de interconexão.

A próxima seção apresenta, com maiores detalhes, o simulador proposto. Na Seção 3, é apresentado um estudo de caso juntamente com alguns resultados parciais. Por fim, a Seção 4 apresenta as considerações finais.

2. Descrição do simulador SimSPD

O SimSPD é um simulador de sistemas de processamento distribuído para suportar o desenvolvimento e avaliação de desempenho de aplicações paralelas. O simulador permite a seleção do número e do tipo dos nós computacionais (*hosts*), bem como do tipo de infraestrutura de rede para a comunicação entre processos paralelos.

Para o desenvolvimento de uma nova aplicação, o SimSPD oferece uma linguagem de programação, que é um subconjunto da linguagem C, para especificar o comportamento do processo paralelo. Após a especificação, o simulador gera, com o uso de um *parser*, uma Máquina de Estados Finitos (MEF) [Wang 2013], para cada processo. As MEFs são utilizadas para simular a execução da aplicação paralela nos *hosts*. Na Figura 1 (a), pode-se ver um exemplo de código paralelo desenvolvido no simulador e, na Figura 1 (b), a respectiva MEF, gerada pelo *parser*.

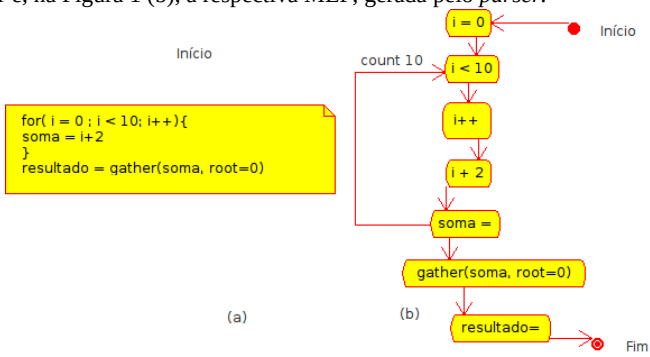

Figura 1 - Exemplo de uma aplicação paralela. Em (a) é representado o código-fonte e, em (b), a respectiva MEF.

O código-fonte na Figura 1 (a) mostra um bloco *for*, onde é atualizado o valor da variável *soma* a cada iteração. Após as iterações, o valor em *soma* é enviado, através da instrução de comunicação *gather* para o processo de *rank* 0. As funções de comunicação utilizadas no simulador têm especificação semelhante às das contidas no padrão MPI.

Para coordenar a transição de estados nas MEFs, o simulador conta com um relógio global. Cada *host* do SPD captura os eventos desse relógio, que serão

multiplicados de forma a representar um relógio local, correspondendo à frequência do respectivo processador. Cada MEF efetuará uma transição de estado a partir de cada evento de relógio local. O relógio global também é utilizado para coordenar as tarefas de comunicação. Seus eventos também serão multiplicados de acordo com a interconexão de rede selecionada, compondo assim o relógio de comunicação. Este será utilizado para sincronizar a comunicação entre MEFs.

O processo de avaliação de desempenho de uma aplicação paralela em desenvolvimento é dado a partir da simulação, onde, ao fim, são apresentadas métricas de desempenho, como: o tempo de execução do algoritmo em cada nó, o tempo total de comunicação e o tempo total para concluir a execução da aplicação paralela.

O simulador foi codificado com a linguagem de programação Java e o processo de simulação é completamente suportado pelo uso de *threads*, as quais permitem a simulação dos processos paralelos, através das MEFs, bem como do relógio de sistema.

A próxima seção apresenta um estudo de caso utilizado para avaliar o processo de desenvolvimento de uma aplicação paralela, utilizando o SimSPD. Serão mostrados, ainda na próxima seção, alguns resultados parciais.

3. Estudo de Caso e Resultados Parciais

Para avaliar o suporte ao desenvolvimento de novas aplicações, foi proposto um estudo de caso que consiste no desenvolvimento e avaliação de uma aplicação paralela de multiplicação de matrizes. Essa aplicação consiste de dois tipos de processo: o primeiro é responsável por gerar aleatoriamente as matrizes que serão multiplicadas e distribuir uma linha e uma coluna para cada um dos demais processos; já o segundo tipo, que recebe a linha e a coluna, multiplica-as e envia o resultado para o processo inicial.

No simulador, o *parser* irá gerar dois tipos de MEF, uma para cada tipo de processo. A MEF para geração das matrizes será mapeada para o *host* mestre e a MEF de multiplicação de linha e coluna será mapeada para os *hosts* escravos.

Para a execução da aplicação, definiu-se matrizes 20x20 de números inteiros de 32 bits e utilizou-se duas configurações distintas de SPDs: a primeira consiste de uma rede de interconexão sem fio, com taxa de transferência de 11 Mbps, e 21 hosts com processadores de 1,6 GHz; já a segunda inclui uma rede cabeada *Fast Ethernet*, com taxa de transferência de 100Mbps, e 21 *hosts* com processadores de 2,2 GHz.

A Tabela 1 mostra um comparativo entre os desempenhos de computação para geração e multiplicação das matrizes, considerando os dois cenários descritos anteriormente. Além disso, também pode ser visto os tempos necessários para envio das linhas e colunas das matrizes a todos os *hosts* escravos, bem como o tempo para os mesmos enviarem suas respectivas respostas ao *host* mestre. Os resultados mostraram que ao se utilizar processadores com maior frequência de operação é possível reduzir os tempos de computação, bem como ao se utilizar interconexões com taxas de transferência maiores é possível reduzir as latências de comunicação. Ainda na Tabela 1 percebe-se que ao se utilizar a combinação de processadores e interconexões de redes

mais rápidas, foi possível realizar um *speedup* de mais de 10x no desempenho da aplicação (ver última coluna).

Tabela 1 - Resultados das simulação da aplicação de multiplicação de matrizes para duas configurações de SPDs distintas.

Configuração	Tempo de Computação em cada *Host* (em segundos)	Tempo Total de Comunicação (em segundos)	Tempo Total de Execução da Aplicação (em segundos)
Rede sem Fio e *Hosts* com Processador de 1,6 GHz.	Geração de Matrizes: $2{,}56 \times 10^{-6}$ Multiplicação de Matrizes: $4{,}6 \times 10^{-5}$	Envio de Matrizes: $2{,}22 \times 10^{-2}$ Envio de Resposta: $5{,}55 \times 10^{-5}$	$2{,}23 \times 10^{-2}$
Rede Cabeada e *Hosts* com Processador de 2,2 GHz.	Geração de Matrizes: $1{,}86 \times 10^{-6}$ Multiplicação de Matrizes: $3{,}35 \times 10^{-5}$	Envio de Matrizes: $2{,}44 \times 10^{-3}$ Envio de Resposta: $6{,}10 \times 10^{-6}$	$2{,}48 \times 10^{-3}$

4. Conclusões

No projeto de aplicações paralelas, diversas variáveis devem ser consideradas, como, por exemplo, estrutura do SPD, configuração de hardware, interconexão e distribuição de carga.

Diante disto, este trabalho apresentou um simulador de SPDs onde é possível desenvolver um aplicação paralela bem como avaliar seu desempenho sob diferentes condições de hardware e interconexões de rede.

Para avaliação do simulador foi proposto um estudo de caso que consistiu no desenvolvimento e avaliação de desempenhode uma aplicação paralela sob diferentes condições de configuração de hardware e interconexão de redes. Os resultados mostraram que foi possível aumentar o desempenho em mais de 10x.

Referências Bibliográficas

Tanenbaum, A. S. (2007) Sistemas Distribuídos: princípios e paradigmas. 2ª ed. São Paulo: Pearson Prentice Hall.

Urbán, P., Défago, X., Schiper, A. (2002) Neko: A single environment to simulate and prototype distributed algorithms. In: Journal of Information Science and Engineering, 18(6):981–997, November.

Sulistio, A., Cibej, U., Venugopal, S. Robic, B., Buyya, R, (2008) A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim, In: Concurrency and Computation: Practice and Experience, Volume 20, Number 13, Pages: 1591 - 1609, ISSN: 1532-0626, Wiley Press.

Casanova, H., Legrand, A., Quinson, M. (2008) SimGrid: a Generic Framework for Large-Scale Distributed Experimentations, In: Proceedings of the 10th IEEE International Conference on Computer Modelling and Simulation.

MPI Forum (2008) “Message passing interface forum”, http://www.mpi-forum.org/.

Wang, J. (2013) Handbook of Finite State Based Models and Applications, CRC Press.