XXIII Seminário de Iniciação Científica
XX Jornada de Pesquisa
XVI Jornada de Extensão
V Mostra de Iniciação Científica Júnior
V Seminário de Inovação e Tecnologia

Modalidade do trabalho: Relatório técnico-científico
Evento: XX Jornada de Pesquisa

BALANCEAMENTO DE CARGA EM SISTEMAS MULTIPROCESSADORES UTILIZANDO O MODELO DE PROGRAMAÇÃO CHARM++[1]

Guilherme Henrique Schiefelbein Arruda[2], Edson Luiz Padoin[3].

[1] Trabalho desenvolvido no contexto do projeto "Utilização de Balanceamento de Carga e DVFS na Paralelização de Aplicações Almejando Aumento da Eficiência Energética em Sistema Paralelos e Heterogêneos"
[2] Aluno do curso de Ciência da Computação da Unijuí, guilherme.arruda@unijui.edu.br
[3] Professor Orientador do Departamento de Ciências Exatas e Engenharias, padoin@unijui.edu.br

Introdução

À medida que a produção de software avança, ocorre um crescimento da demanda por processamento computacional. A necessidade de alto desempenho fez com que sistemas computacionais fossem desenvolvidos para suprir esta necessidade. Um dos componentes de hardware que apresentou uma grande evolução foi o processador. Este deixou de ser single core e passou a contar com vários núcleos, proporcionando a execução simultânea de diversas aplicações. Com isso, foi possível construir computadores pessoais, sistemas embarcados e até supercomputadores. Isso abriu as portas para a programação paralela, pois possibilitou que as aplicações fossem divididas em partes e inseridas nos núcleos das unidades de processamento (Pilla & Meneses, 2015).

Desde então, diversos programas paralelos são construídos para que seja possível utilizar todo o potencial dos sistemas computacionais atuais. Infelizmente, a programação paralela ainda é difícil de aplicar em plataformas de desenvolvimento. É possível utilizar algumas interfaces de programação de aplicativos (APIs) específicos para desenvolver programas paralelos, facilitando esse trabalho para aplicações regulares e ambientes homogêneos. Como exemplo destas ferramentas pode-se citar o OpenMP (Chandra, 2001) e o MPI (Dongarra, 1996). Porém, utilizá-las em aplicações dinâmicas ou ambientes variáveis pode acarretar em perda de desempenho (Pilla & Meneses, 2015).

Será utilizado o modelo de programação Charm++ pelo fato de ser a ferramenta mais indicada para trabalhar com ambientes multiprocessadores. Com este modelo, será criado um novo balanceador de carga (BC) que faz uso de uma estratégia, baseada em médias aritméticas, a qual permite equilibrar as cargas e o tempo de execução entre os processadores, impedindo que um deles fique ocioso ou sobrecarregado. Assim, o BC proposto foi denominado AverageLB.

Metodologia

Modalidade do trabalho: Relatório técnico-científico
Evento: XX Jornada de Pesquisa

Alguns ambientes de programação adotam uma metodologia baseada na medição das cargas dos objetos que executam em cada processador. Para isso, o BC coleta automaticamente estatísticas da carga computacional e da comunicação destes objetos e armazena estas informações em um banco de dados. Este banco de dados vai ajudar o BC a decedir quando e onde migrar os objetos (Jyothi et al, 2004). Um exemplo deste tipo de ambiente é o Charm++ (Kale & Krishnan, 1993), que é uma extensão da linguagem C++, no qual proporciona um ambiente para programação paralela orientada a objetos. Seu desenvolvimento foi feito pelo Laboratório de Programação Paralela da Universidade de Illinois, em 1993. Oferece suporte a diversas plataformas e permite que os programas desenvolvidos neste modelo executem tanto em ambientes com memória compartilhada quanto com distribuída. Esta ferramenta utiliza uma técnica que consiste em dividir um problema em diversos componentes migráveis, que são executados em vários processadores. Estes componentes migráveis são chamados de chares.

Chares são uma das entidades de Charm++ e referem-se aos objetos do C++ com seus atributos e dados privados, sendo instanciados através de uma classe chare. Isso significa que um chare não pode acessar as informações de outro diretamente. Para isso, a comunicação entre eles é feita através da troca de mensagens. O BC proposto foi desenvolvido utilizando o framework de balanceamento de cargas disponibilizado pelo Charm++. Possui uma abordagem centralizada, o que significa que a estrutura de cargas e de comunicação da máquina, além de um processo de tomada de decisão, ficam armazenadas em um único ponto. A escolha de uma abordagem centralizada se deu pelo fato desta realizar um balanceamento mais preciso em relação às outras abordagens. A ideia deste BC é utilizar as informações coletadas em tempo de execução para criar duas variáveis que vão conter médias aritméticas. Estas serão usadas para controlar quais chares devem ser migrados e qual valor cada processador deve ter para estar em equilíbrio. Devido à esta técnica de equilíbrio de cargas através de médias, o BC proposto foi chamado de AverageLB.

Foi criado para se adaptar melhor com a tarefa de equilibrar cargas de processadores sem precisar utilizar dois ou mais BCs diferentes para isso. Para que o tempo de execução dos processadores fossem equilibrados, foi utilizada uma estratégia baseada na média aritmética de suas cargas. Primeiramente, o algoritmo extrai a quantidade de objetos e o total de carga de cada processador. Neste ponto, já é possível perceber que estas duas variáveis são desproporcionais devido ao modo como os objetos são distribuídos pelas aplicações. O próximo passo é calcular a carga média por processador (CMP), que é feito dividindo o somatório das cargas de cada objeto pelo número total de objetos de cada processador. Esta variável é usada para fazer o controle de quais objetos devem ser migrados de um processador para outro, onde os objetos que possuem uma carga maior que a média são migrados, enquanto que os demais permanecem executando em seu processador de origem. Em seguida, é calculada a média aritmética geral (MAG), que servirá como limitante para equilibrar as cargas dos processadores.

Para efetuar este cálculo, é feito a soma da carga de cada processador e o resultado é dividido pelo número de processadores. Com estas duas variáveis definidas o BC começa o processo de equilíbrio

das cargas. A imagem apresenta dois processadores, X e Y, que foram selecionados pelo BC através de um método. Este método é responsável por mapear as unidades de processamento e selecionar a primeira que se encontra abaixo da MAG, que neste caso é representado por Y. Já o X representa o primeiro processador que está acima da MAG.

Após esta seleção, o BC mapeia os chares do processador X a fim de identificar aqueles que se encontram acima do seu CMP. Uma vez identificados, inicia-se um processo de migração que moverá os chares do processador X para o Y até que a carga de Y possua um valor próximo à MAG. Quando isso acontecer, o BC busca o próximo processador Y abaixo da média geral. O mesmo vale para o processador X.
O processo de balanceamento é encerrado quando não existe mais processadores a serem mapeados e as cargas de todas as unidades de processamento possuem um valor próximo à MAG, como mostra a Figura 2. Mesmo que o número de chares entre X e Y seja diferente, o que realmente importa é o valor total da carga destes objetos.
Resultados e Discussão
A plataforma em que o balanceador proposto foi executado consiste de um notebook que contém um processador octa-core Intel Core i7 3830QM e um total de 8 GB de memória RAM. Nesta máquina, foram instalados o sistema operacional Fedora 21 Linux cuja versão do kernel é 4.0.4-202.fc21.x86_64, a versão 6.5.1 do modelo de programação paralela Charm++, a versão 4.9.2 do compilador gcc para a linguagem C++ e um editor de código para realizar a implementação do algorimto.

Nesta plataforma, foi utilizado o benchmark lb_test por se tratar de uma ferramenta bastante utilizada para avaliar balanceadores de carga. Com ela, foi realizado um teste com o algoritmo do AverageLB e o resultado disso foi comparado com dois BCs disponibilizados pelo Charm++, que são o GreedyLB e o RefineLB. A Figura 1 mostra o gráfico resultante da avaliação do benchmark que compara os tempos de execução dos três balanceadores.

Modalidade do trabalho: Relatório técnico-científico
Evento: XX Jornada de Pesquisa

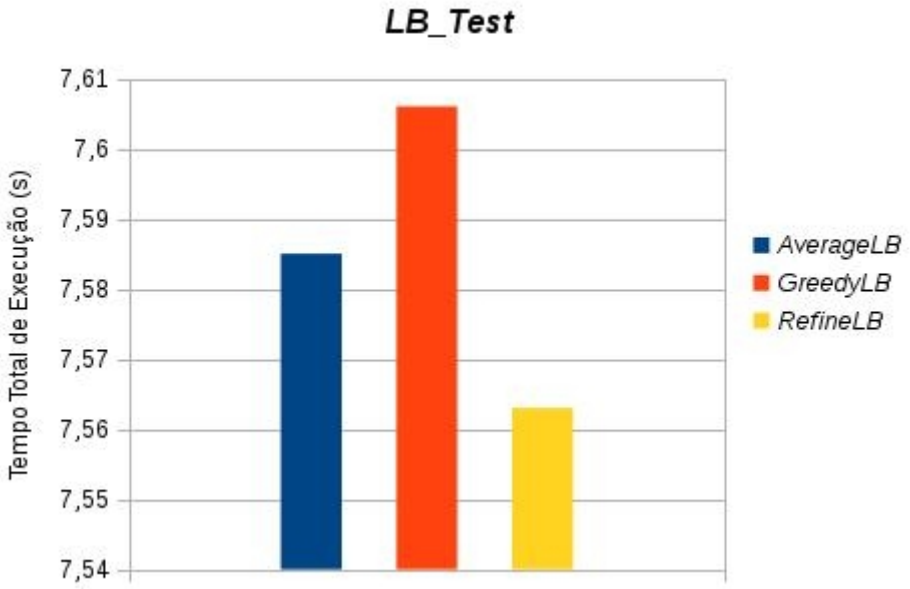

Figura 1. Comparação de diferentes balanceadores de carga para o benchmark lb_test

Conclusões

A maioria das aplicações paralelas envolve comportamentos dinâmicos ou cálculos baseados em diversas fórmulas complexas. Por conta disso, empresas e instituições buscam adquirir uma infraestrutura suficiente para suportar tais aplicações. A presente situação do ambiente paralelo mostra uma grande preocupação com os sistemas computacionais de alto desempenho (CAD). Este fator é consequência do objetivo destes sistemas de atingir a escala do hexaflop. Por conta disso, existe um grande investimento em máquinas que possuem centenas de milhares de processadores, pois necessitam obter resultados precisos no menor tempo possível.

O grande problema por trás disso é que muitas vezes não há uma preocupação com o desbalanceamento de carga gerado por estas aplicações, ou até mesmo pelas próprias máquinas, que tendem a sofrer com problemas de temperatura e consumo de energia. O desbalanceamento das cargas computacionais é o grande responsável por impedir que as máquinas paralelas aproveitem todo o seu desempenho. Diante deste grande problema, foi construído um balanceador de carga baseado em uma estratégia que utiliza médias aritméticas para equilibrar as cargas em cada processador.

Por conta disso, este balanceador recebe o nome de AverangeLB. As médias são limitantes para controlar a taxa de migração de objetos, pois é uma tarefa muito custosa e que muitas vezes é ignorada por algoritmos de balanceamento de carga. Testes com este balanceador mostra que seu tempo de execução é adequado comparado aos outros balanceadores com os quais este foi

comparado. Diante disso, os próximos passsos deste trabalho estão voltados para a comunicação entre os objetos, onde pretende-se considerar esta variável. Também objetiva-se realizar testes nas grandes máquinas paralelas a fim de resolver problemas reais.

Palavras-chave

Balanceamento de Carga; Multiprocessadores; Modelo de Programação

Referências Bibliográficas

Pilla, L. L. and Meneses, E. (2015). Programação paralela em Charm++.

Jyothi, R., Lawlor, O. S., and Kale, L. V. (2004). Debugging support for Charm++. In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, page 264. IEEE.

Chandra, R. (2001). Parallel programming in OpenMP. Morgan Kaufmann.

Dongarra, J. J., Otto, S. W., Snir, M., and Walker, D. (1996). A message passing standard for mpp and workstations. Communications of the ACM, 39(7):84–90.

Kale, L. V. and Krishnan, S. (1993). Charm++: a portable concurrent object oriented system based on C++, volume 28. ACM.