SALÃO DO CONHECIMENTO UNIJUÍ 2018
CIÊNCIA PARA A REDUÇÃO DAS DESIGUALDADES
XXVI Seminário de Iniciação Científica
XXIII Jornada de Pesquisa
XIX Jornada de Extensão
VIII Mostra de Iniciação Científica Júnior
VIII Seminário de Inovação e Tecnologia
01 a 04 de outubro de 2018

Evento: Bolsistas de Iniciação Científica e Iniciação Tecnológica da Unijuí

ALGORITMO DE BALANCEAMENTO DE CARGA PARA REDUÇÃO DO TEMPO DE EXECUÇÃO DE APLICAÇÕES PARALELAS[1]
PARALLEL APPLICATIONS' RUNTIME REDUCTION FOCUSED LOAD BALANCEMENT ALGORITHM

Vinícius Mânica Mastella[2], Edson Luiz Padoin[3], Giovane Da Rosa Lizot[4], Vinícius Ribas Samuel Dos Santos[5], Cleber Cristiano Sartorio[6]

[1] Trabalho parcialmente apoiado por UNIJUI e CNPq. Pesquisa realizada no contexto do Laboratório Internacional Associado LICIA e tem recebido recursos do edital da VRPGPE de bolsa e PIBIC/UNIJUI
[2] Aluno do curso de Ciência da Computação da Unijuí e Bolsista PIBIC/UNIJUÍ,
[3] Professor Orientador do Departamento de Ciências Exatas e Engenharias
[4] Aluno do curso de Ciência da Computação da Unijuí e Bolsista PIBIC/UNIJUÍ
[5] Aluno do Curso de Ciência da Computação UNIJUÍ
[6] Egresso do curso de Ciência da Computação UNIJUÍ

INTRODUÇÃO

Atualmente as simulações computacionais mostram-se cada vez mais complexas, continuamente demandando mais poder de processamento dos sistemas de High Performance Computing (HPC). Quando paralelizadas, apresentam excessiva comunicação entre tarefas e desbalanceamento de cargas, impedindo o uso eficiente do seu potencial. Sendo assim, algumas unidades de processamento podem receber tarefas com menor carga ou permanecer ociosas enquanto outras executam as aplicações, causando, como consequência, ineficiência na utilização dos sistemas e aumento do tempo de execução.
Para resolver tais impasses, balanceadores de carga têm sido propostos, almejando aumentar a utilização do potencial dos sistemas computacionais em nível de processamento e tempo de execução. Nesse contexto, muitas aplicações paralelas que envolvem simulação com comportamentos dinâmicos ou cálculos baseados em fórmulas complexas têm utilizado tais recursos devido ao desbalanceamento de carga e a quantidade de comunicações [PILLA 2015].

A computação paralela cresceu advinda dos sistemas com múltiplas unidades de processamento. Pode-se ser analisado como a paralelização de tarefas ajuda no desempenho dos processadores com migrações de tarefas, do mesmo modo que a paralelização das aplicações explorou a concorrência com o objetivo de melhorar a eficiência no processamento paralelo.

METODOLOGIA

Primeiramente foram selecionados os balanceadores de carga AverageLB [ARRUDA 2015] e SmartLB [VINÍCIUS 2017] como base para o desenvolvimento de novo balanceador, adotando-se a

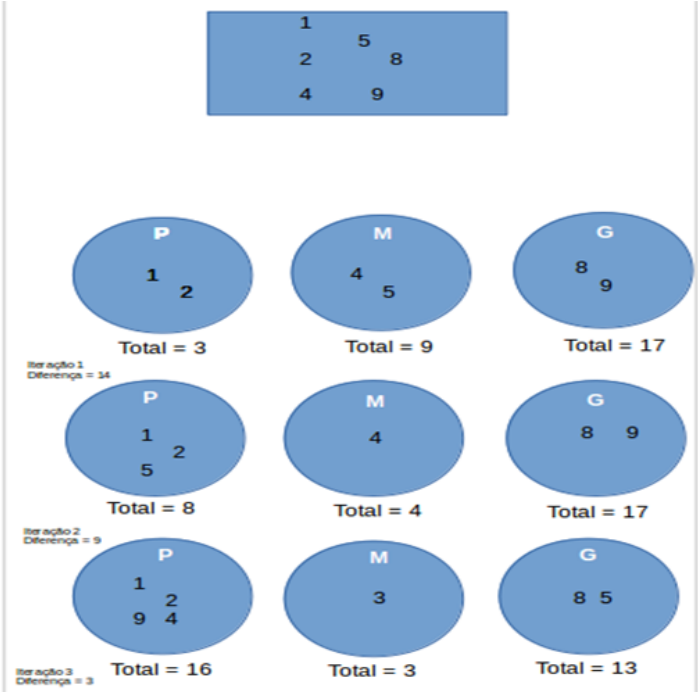
plataforma de programação Charm++ para sua implementação. Este ambiente oferece suporte a diversas plataformas, permitindo aos programas desenvolvidos neste modelo executarem tanto em ambientes com memória compartilhada como distribuída.
O Balanceador de Cargas AverageLB [ARRUDA 2015] tem como estratégia de balanceamento uma abordagem do tipo centralizada, o qual toma decisões em um único processo. A estratégia do algoritmo leva em consideração a média aritmética de cada processador, calculando suas cargas, com o intuito de reduzir o número de migrações, buscando um equilíbrio entre as cargas [FREYTAG 2015].
Ambos os algoritmos coletam informações do sistema e da aplicação em tempo de execução para utilizá-las dinamicamente nas tomadas de decisões de balanceamento de carga, visando reduzir o tempo total de execução. A partir destes, um novo balanceador de carga foi proposto, denominado GroupLB. Este adota uma estratégia que divide os processos de acordo com as cargas computacionais em três grupos, denominados Pequeno (P), Médio (M) e Grande (G), os quais, respectivamente, armazenam os processos de cargas consideradas relativamente pequenas, médias e grandes.
Desta forma, com a aplicação do balanceador, serão verificadas as cargas dos cores, e, a partir dessa informação, será analisada a diferença de cargas entre os cores. Posteriormente, são divididos os pesos das cargas nos respectivos grupos, ou seja, uma lista de todas as cargas analisadas é passada aos arrays. No Algoritmo haverão as variáveis de controle, onde a variável “less”, representa a soma das cargas no array P, a variável “bigger” a soma das cargas no array G, e a variável “delta” a diferença entre as variáveis G e P.

Na Figura 1 é exibido a estratégia de iteração adotada. Nela pode ser vista a divisão das cargas nos grupos Pequeno, Médio e Grande, como também analisar como a diferença dos grupos Pequeno e Grande, a variável “delta”, diminui a cada iteração. Ao final as cargas estão equilibradas de acordo com a estratégia adotada, diminuindo o desbalanceamento, as migrações desnecessárias e o tempo de execução da aplicação.

Para validar a proposta, utilizou-se um equipamento: processador modelo Intel Core i5-3230M com 4 cores físicos. Para os testes, utilizou-se o sistema operacional Linux Ubuntu 16.04 com kernel versão 4.4.33-1. A versão do Charm++ utilizada foi 6.5.1 e compilador g++ na versão 6.2.1.
Cada um dos testes realizados neste trabalho foi repetido 5 vezes, para atingirmos um erro relativo menor que 5% e 95% de confiança estatística para uma Student’s t-distribution. Entre cada um dos testes foi deixado o sistema em idle por no mínimo 20 segundos, de modo que a demanda de potência do sistema se estabilize.

Evento: Bolsistas de Iniciação Científica e Iniciação Tecnológica da Unijuí

Figura 1. Estratégia do Balanceador de Cargas



Fonte: O Autor.

RESULTADOS

Com a intenção de testar o funcionamento do balanceador de carga GroupLB, o mesmo foi aplicado na execução do benchmark LB_Test e comparado a outros dois balanceadores disponibilizados pelo ambiente de programação do Charm++. Sendo estes:

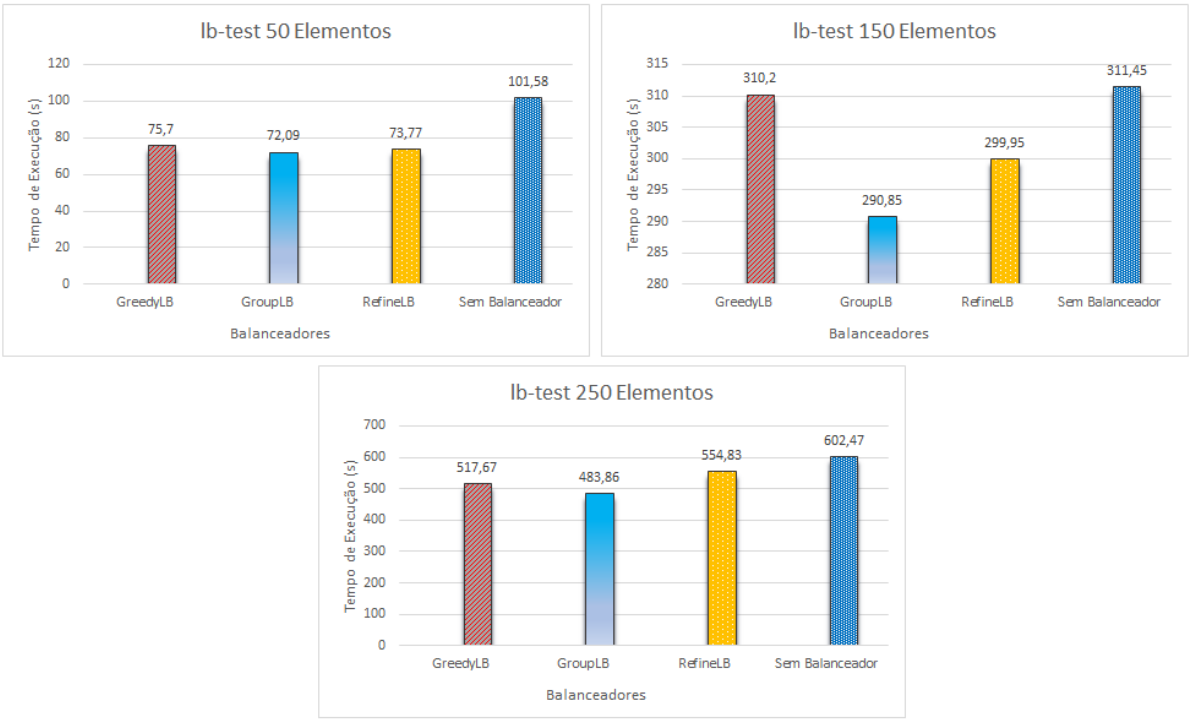
• RefineLB: Balanceador que possui abordagem centralizada e é baseado em refinamento. Esse BC move objetos dos cores mais sobrecarregados para os menos carregados até atingir uma média, que é definida através de um método específico, limitando o número do objetos migrados [ARRUDA 2015].

• GreedyLB: É um algoritmo de balanceamento de carga guloso. Esse BC remove todas as tarefas de seus núcleos e as mapeia em ordem decrescente de carga entre os núcleos com as menores cargas [FREITAS 2016]. Essa estratégia migra objetos mais pesados para o processador com a menor carga, até que a carga de todos os processadores esteja próxima à carga média.

Na Figura 2 são apresentados os resultados dos testes realizados com o benchmark lb_test na plataforma já descrita.

Evento: Bolsistas de Iniciação Científica e Iniciação Tecnológica da Unijuí

Figura 2 – Tempo de execução processador com 4 Núcleos



Fonte: O Autor.

Nos testes realizados o balanceador de carga GroupLB também conseguiu um melhor desempenho com o benchmark testado. Com 50 tarefas o tempo foi reduzido em 29,03% em relação ao teste sem balanceador de carga. Em relação ao balanceador RefineLB ficou menor em 2,28% e 4,77% em relação ao GreedyLB. Com 150 tarefas o GroupLB também apresentou redução em relação ao teste sem balanceador, com um percentual de 6,62%, já comparado ao balanceador RefineLB obteve um percentual de 3,04%, e por fim obteve 6,23% em relação ao balanceador GreedyLB.
Da mesma forma, nos testes com 250 tarefas, o GroupLB também obteve ganhos. Reduziu o tempo em 19,68% em relação a execução sem balanceador, 12,80% em relação ao balanceador RefineLB e 6,54% em relação ao balanceador GreedyLB.

CONSIDERAÇÕES FINAIS

Este artigo apresentou uma proposta de um novo balanceador de cargas desenvolvido para o

modelo de programação Charm++, com o objetivo de reduzir o tempo total de execução da aplicação, bem como reduzir o número de migrações de processos. Nos testes realizados com o benchmark lb-test, tal balanceador apresentou resultados superiores aos balanceadores RefineLB e GreedyLB. Como futuros trabalhos, pretende-se realizar testes em sistemas paralelos, utilizando outros benchmarks e problemas reais de computação científica.

Palavras-chave
Charm++, Balanceamento de Carga, Computação de Alto Desempenho.
Keywords
Charm++, Load Balance, High Performance Computing.
Agradecimentos
Trabalho apoiado com recursos do programa do edital de bolsa PIBIC da UNIJUI.

REFERÊNCIAS

ARRUDA, G. H. S. (2015). **Balanceamento de carga em sistemas multiprocessadores utilizando o modelo de programação Charm++**. In Salão do Conhecimento.

DONG, Y., Chen, J., and Tang, T. (2010). **Power measurements and analyses of massive object storage system**. In Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, pages 1317–1322. IEEE.

FREYTAG, G., Arruda, G., Martins, R. S. M., and Padoin, E. L. (2015). **Análise de desempenho da paralelização do problema de caixeiro viajante**. In XV Escola Regional de Alto Desempenho (ERAD), pages 1–4, Gramado, RS. SBC.

PADOIN, E. L., Castro, M. B., Pilla, L. L., Bozzetti, T. C., Navaux, P. O. A., and Méhaut, J.-F. (2014). **Balanceamento de carga visando redução do consumo de energia para o modelo de programaçao charm++**. XIV Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul, Alegrete, RS, Brasil.

PADOIN, E., Castro, M., Pilla, L., Navaux, P., and Mehaut, J.-F. (2014). **Saving energy by exploiting residual imbalances on iterative applications**. In High Performance Computing (HiPC), 2014 21st International Conference on.

SANTOS, V. R. S. SmartLB.(2017): **Proposta de um Balanceador de Carga para redução de tempo de execução em ambientes de memória compartilhada**.In Salão do Conhecimento.