Aplicação do balanceador de carga GROUPLB nas aplicações LASSEN e LULESH*

Giovane Lizot[1], **Edson L. Padoin**[1]

[1]Universidade Reg. do Noroeste do Estado do Rio G. do Sul (UNIJUI) - Ijuí - RS - Brasil

giovanerosalizott@hotmail.com, padoin@unijui.edu.br

Resumo. *Este artigo apresenta uma análise dos ganhos de desempenho alcançado com a aplicação do balanceador de carga* GROUPLB *na execução das aplicações* LASSEN *e* LULESH*. Buscando a redução do tempo de execução de aplicações iterativas executadas em ambientes paralelos de memória compartilhada. Os resultados iniciais da aplicação sobre as aplicações* **LULESH** *e* **LASSEN** *apresentaram ganhos de até 64.958% comparado aos testes sem uso de balanceador de carga.*

1. Introdução

Dado o crescimento de aplicações paralelas que demandam de uma grande quantidade de processamento, se torna necessário a implementação de simulações que buscam soluções cada vez mais otimizadas. Sabendo que a demanda das aplicações paralelas apresentam comportamentos dinâmicos através de cálculos baseados em fórmulas complexas, desta forma, acabam exigindo continuamente mais poder de processamento dos sistemas de computação de alto desempenho *(High Performance Computing - HPC)* [Arruda et al. 2015].

Deste modo, algumas aplicações, quando paralelizadas, apresentam excessiva comunicação entre suas tarefas e desbalanceamento de carga. Com o desbalanceamento de carga presente, surgem diversos problemas, como a impossibilidade do uso eficiente de todos os recursos dos sistemas paralelos. Em vista disso, alguns núcleos tem a possibilidade de receber tarefas com menores cargas, ou permanecer inativos, enquanto outros executam as aplicações, causando uma ineficiência no uso dos sistemas [Padoin et al. 2014]. Diante deste problema, *balanceadores de carga* são desenvolvidos almejando uma melhor distribuição de cargas das tarefas entre as unidades de processamento.

Nesse sentido, este artigo apresenta os resultados da aplicação de um novo *balanceador de carga* denominado GROUPLB na execução das aplicações reais LASSEN e LULESH. Nossa proposta almeja a redução do tempo total de execução da aplicação por meio da migração de tarefas (alocação de tarefas de um core sobrecarregado para outro com menor carga) considerando uma divisão das cargas em três grupos, de acordo com o tamanho e peso das cargas.

O restante do trabalho está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta a proposta do *balanceador de carga* GROUPLB. A Seção 4 descreve a metodologia utilizada na implementação e o ambiente de execução utilizado na realização dos testes. Resultados são discutidos na Seção 5, seguidos das conclusões e trabalhos futuros.

*Trabalho desenvolvido com recursos do edital MCTIC/CNPq - Universal 28/2018 sob número 436339/2018-8 e do edital da VRPGPE bolsa PIBIC/UNIJUI.

2. Trabalhos Relacionados

A grande maioria das estratégias propostas utilizam uma forma centralizada de *balanceamento de carga*. Elas consideram um processador específico para a tomada de decisões de *balanceamento de carga*, tomando como base dados de carga, comunicação e tempo total de execução [Zheng et al. 2010]. Outras estratégias, por sua vez, recorrem a esquemas de *balanceamento de carga* com estratégias distribuídas ou ainda constroem um modelo de carga de trabalho para orientar a atribuição de tarefas aos nodos de processamento.

Algumas propostas de *Balanceadores de carga* tem sido implementadas e testadas utilizando o ambiente de programação CHARM++. Nesse sentido, para o desenvolvimento deste trabalho considerou-se os balanceadores de carga GREEDYLB e AVERAGELB [Bang-Jensen et al. 2004], REFINELB [Freytag et al. 2015], AVERAGELB [Freytag et al. 2015], e SMARTLB [dos Santos et al. 2018] já implementados no CHARM++ para construção e desta proposta.

3. GROUPLB

A estratégia utilizada para implementação do balanceamento de carga GROUPLB divide dos processos em três grupos de acordo com as suas cargas computacionais [da Rosa Lizot et al. 2018]. Quando o balanceador é executado, primeiramente ele verifica as cargas dos núcleos de processamento determinando a diferença de cargas entre os núcleos, ou seja o desbalanceamento presente.

Numa segunda etapa, as tarefas são divididas em *Small* (S), *Medium* (M) e *Bigger* (B) de acordo com as suas cargas computacionais. Nesta etapa é realizado um somatório das cargas de todos dos processos do grupo *S* (*"smaller"*), e um somatório das cargas de todos dos processos do grupo *matriz B* (*"bigger"*). Isto permite a determinação de um *"delta"*, ou seja a diferença entre as cargas dos processos *Bigger* e *Smaller*.

Tendo calculado o desbalanceamento atual entre nodos de processamento e um *"delta"*, o algoritmo passa migrar as tarefas alocadas no grupo *M* para o grupo *S*. Assim, o grupo *M* é considerado como um *vetor pivô*, que de acordo com as tomada de decisões pode receber tarefas dos grupos *S e B* para reduzir o desbalanceamento

A solução proposta, busca equilibrar as cargas entre os nodos de processamento reduzindo as migrações e diminuindo o tempo total de execução da aplicação.

4. Metodologia

O equipamento utilizado nos testes possui um processador Intel Core i7-4790 de 4 cores. Foi instalado sistema operacional Linux Ubuntu 18.04 com kernel versão 4.4.33. Para implementação foi utilizado CHARM++ versão 6.5.1 com compilador g++ de versão 5.4.1.

Para validação e análise de desempenho foram utilizadas duas aplicações reais. A primeira aplicação é **LULESH** que foi originalmente desenvolvida como um dos cinco problemas de desafio na *DARPA Ubiquitous High Performance Computing (UHPC)* program [Dosanjh et al. 2014]. Ela pode ser utilizado para testar recursos de hardware, salientando a vetorização do compilador. Simula uma variedade de problemas na área da ciência e de engenharia que requerem a modelagem hidrodinâmica descrevendo o movimento de materiais quando aplicados a uma determinada força.

A segunda é **LASSEN**, uma aplicação que explora os dados com equilíbrio de carga variável. Ela modela e simula a propagação através de uma onda que propagada em torno de uma malha não estruturada. A carga computacional da malha é subdividida em domínios, que são atribuídos aos processadores. Portanto, isso se torna um problema desafiador para efetivamente paralelizar uma vez que a carga de trabalho é submetida a constantes mudanças [McCandless 2015].

Nos testes, o balanceador GROUPLB, foi configurado com carga computacional calculada pela aplicação de forma que os pesos das cargas eram alocados aleatoriamente. Foi levando em consideração o tempo de execução e a quantidade de migrações de tarefas. A sincronização do balanceador foi definida a cada 10 iterações. Os resultados foram calculados realizando a média dos tempos de 15 execuções.

5. Resultados

Na Figura 1 são apresentados os tempos de execução mensurados nas execuções das aplicações LULESH e LASSEN com diferentes balanceador de carga. Para ambos os testes, o balanceador GROUPLB apresentou ganhos reduzindo o tempo de execução.

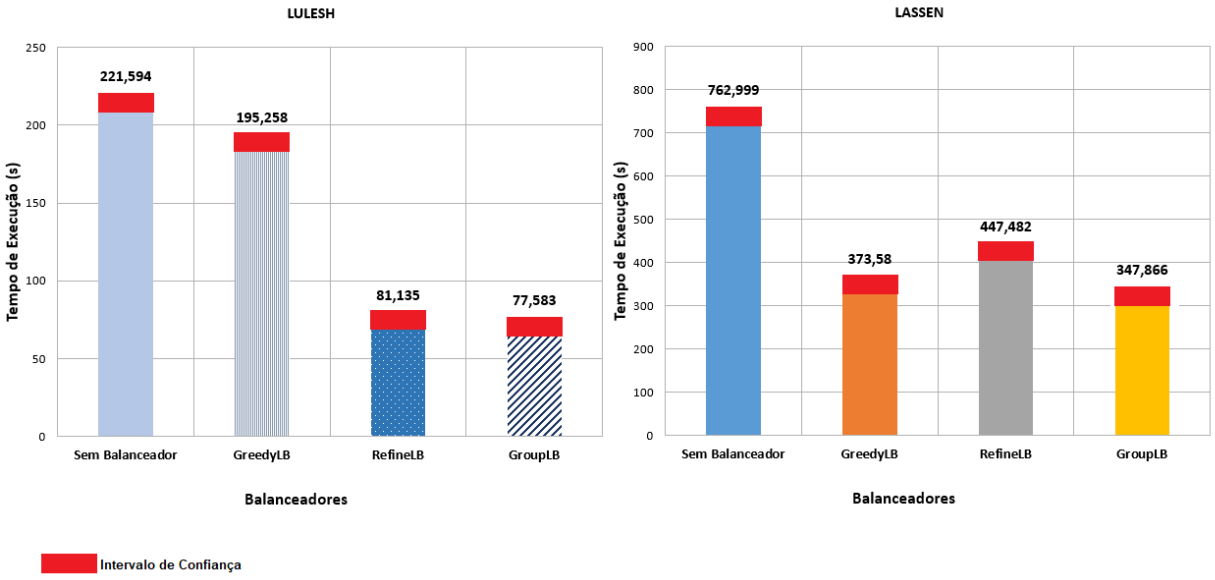

Figura 1. Resultados das execuções com as aplicações LASSEN e LULESH

Na execução da aplicação LULESH, GROUPLB conseguiu reduzir o tempo de execução de $221,5$ para $77,5$ segundos, um *speed-up* de $2,85$. Quando executada a aplicação LASSEN, o *speed-up* foi de $2,19$. O GROUPLB também obteve os menores tempos de execução, reduzindo o tempo de execução de $762,9,5$ para $347,86$ segundos.

6. Conclusões e trabalhos futuros

Este trabalho apresentou uma análise da aplicação do balanceador de carga GROUPLB na execução das aplicações reais *Lulesh* e *Lassen*. Os resultados mostraram-se consistentes quanto as métricas analisadas, conseguindo o algoritmo proposto reduzir o tempo de execução das duas aplicações com ganhos superiores a outros balanceadores do esta da arte. Além da redução do

desbalanceamento de carga, percebeu-se também uma redução da quantidade de migração de tarefas o que resultou na redução do tempo de execução das duas aplicações reais testadas.

Como futuros trabalhos, pretende-se realizar melhorias no algoritmo do GROUPLB de modo a implementar o aprendizado de máquina na solução. Também pretende-se analisar os ganhos alcançados no consumo energético quando utilizada a estratégia proposta considerando outros sistemas paralelos com uma maior quantidade de processadores e outras aplicações reais de computação científica.

Referências

Arruda, G., Padoin, E. L., Pilla, L. L., Navaux, P. O. A., and Mehaut, J.-F. (2015). Proposta de balanceamento de carga para redução de migração de processos em ambientes multiprogramados. In *XVI Simpósio de Sistemas Computacionais (WSCAD-WIC)*, pages 1–8.

Bang-Jensen, J., Gutin, G., and Yeo, A. (2004). When the greedy algorithm fails. *Discrete Optimization*, 1(2):121–127.

da Rosa Lizot, G., Mastella, V. M., Pavan, P. J., and Padoin, E. L. (2018). Grouplb: Load balancer proposal to reduce the execution time of parallel applications. In *XVI Parallel and Distributed Processing Workshop (WSPPD)*, pages 1–4, Porto Alegre, RS.

dos Santos, V. R. S., Padoin, E. L., Navaux, P. O. A., and Méhaut, J.-F. (2018). Smartlb: Proposta de um balanceador de carga para redução de tempo de execução de aplicações em ambientes paralelos. In *Congresso da Sociedade Brasileira de Computação (CSBC) - Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPERFORMANCE)*, pages 1–14.

Dosanjh, S., Barrett, R., Doerfler, D., Hammond, S., Hemmert, K., Heroux, M., Lin, P., Pedretti, K., Rodrigues, A., Trucano, T., et al. (2014). Exascale design space exploration and co-design. *Future Generation Computer Systems*, 30:46–58.

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.

McCandless, B. (2015). Lassen mini-app. *Co-design at Lawrence Livermore National Lab*.

Padoin, E. L., Castro, M., Pilla, L. L., Navaux, P. O., and Méhaut, J.-F. (2014). Saving energy by exploiting residual imbalances on iterative applications. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE.

Zheng, G., Meneses, E., Bhatele, A., and Kale, L. V. (2010). Hierarchical load balancing for charm++ applications on large supercomputers. In *2010 39th International Conference on Parallel Processing Workshops*, pages 436–444. IEEE.