8va JIDeTEV
Investigacion y Desarrollo Tecnológico,
Extensión, Vinculación y Muestra de la Producción

Proposta de um balanceador de carga para redução de tempo de execução de aplicações em ambientes paralelos

Vinicius R. S. dos Santos [a,*], Edson L. Padoin [a,b], Philippe O. A. Navaux [b]

[a] *Universidade Reg. do Noroeste do Estado do Rio G. do Sul (UNIJUI) - Ijuí - RS - Brasil*
[b] *Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS - Brasil*

e-mails: vinicius.ribas@unijui.edu.br, padoin@unijui.edu.br, navaux@inf.ufrgs.br

Resumo

Este artigo apresenta a proposta de um novo balanceador de carga que almeja a redução do tempo de execução e consumo de energia de aplicações paralelas quando executadas em ambientes de memória compartilhada. O algoritmo do balanceador coleta informações do sistema e da aplicação em tempo real e as utiliza para tomar decisões de balanceamento de carga dinamicamente. Para implementação foi utilizado o modelo de programação paralela Charm++. Os resultados preliminares apresentaram redução do tempo de execução de até 40,9% e até 47,7% no consumo de energia para três benchmarks utilizados nos testes.

Palabras Clave *– HPC, Charm++, Balanceamento de Carga, Consumo de Energia, Benchmark, Desempenho.*

1. Introdução

Cada vez mais aplicações científicas são simuladas em ambientes paralelos. Tais simulações computacionais representam atualmente um percentual significativo no total das demandas por processamento nos sistemas de HPC e, somente são possíveis com o emprego de programação paralela. Com paralelismo o processamento das aplicações é dividido em partes e executado em paralelo nas unidades de processamento disponíveis nos atuais de sistemas de HPC.

Para atender às grandes demandas de processamento, diferentes arquiteturas paralelas têm sido projetadas e construídas empregando processadores compostos de múltiplas unidades de processamento. No entanto, a maioria das aplicações paralelas apresentam características, como comportamento dinâmico, que acabam gerando desbalanceamento de cargas, ou excessiva comunicação entre os objetos. Tais características dificultam a eficiente utilização dos sistemas de computação que geralmente possuem unidades de processamento homogêneos.

Nesse sentido, este artigo apresenta a proposta de uma nova estratégia de balanceamento de carga denominado SmartLB. Nossa proposta almeja a redução do tempo total de execução e do consumo de energia por meio da migração de tarefas considerando a carga computacional média das unidades de processamento também denominados de cores ou núcleos.

O trabalho está organizado da seguinte forma. A Seção 2 discute o estado da arte das tecnologias a serem utilizadas e os trabalhos relacionados. A Seção 3 apresenta a estratégia, a metodologias de implementação e o algoritmo do balanceador de carga proposto. A Seção 4 descreve a metodologia

*Autor em correspondência.

e o ambiente utilizado na implementação e execução dos testes. Resultados alcançados são discutidos na Seção 5, seguidos das Conclusões e Trabalhos Futuros.

2. Trabalhos Relacionados

Muitas pesquisas têm sugerido o emprego de balanceamento de carga para reduzir o tempo de execução e o consumo de energia das plataformas quando executam aplicações paralelas.

O balanceamento de carga é uma técnica utilizada para a distribuição uniforme da carga computacional e da comunicação entre todos os cores ou processadores de uma máquina paralela. Estas estratégias podem ser empregadas quando as tarefas são criadas ou periodicamente durante a execução das aplicações paralelas [Zheng et al. 2010].

Estratégias como GreedyLB, RefineLB e RANDCENTLB são consideradas centralizadas, pois fazem decisões de balanceamento de carga em um único processador. Elas coletam informações de carga de toda a máquina em um core específico, o qual que executa um processo de decisão sequencial com base nessas informações [Zheng et al. 2010]. Outras estratégias, visando uma melhor escalabilidade quando aplicadas em sistemas de grande escala, adotam abordagens distribuídas onde os processadores distribuídos tomam decisões usando sua visão local do sistema. Nessas estratégias, os processadores trocam dados de balanceamento de carga apenas entre os vizinhos, como forma de descentralizar o processo de decisão [Cybenko 1989].

3. Balanceador de Carga SmartLB

Nossa proposta, busca reduzir o overhead de balanceamento a partir da implementação de melhorias nas estratégias utilizadas nos algoritmos GreedyLB, RefineLB e AverageLB. Utilizando uma abordagem centralizada almeja-se alcançar balanceamento de carga levando em consideração a média aritmética das cargas de cada unidade de processamento, reduzindo o número total de migrações, o tempo de execução e consequentemente o consumo de energia.

3.1. Metodologia de implementação

Para implementar a estratégia de balanceamento de carga proposta foi utilizado o ambientes de programação paralela Charm++. A escolha por este ambiente foi motivada pela sua madura estrutura de balanceamento de carga, a qual permite tanto a criação de novos balanceadores de carga, quanto a utilização dos balanceadores de carga disponibilizados pelo ambiente para comparações de resultados.

Dentre as informações armazenadas pelo SmartLB destacam-se o número total de tarefas, a carga de cada tarefa, o número total de núcleos e a carga total de cada núcleo. Estes dados são utilizados pela estratégia na tomada de decisões para definir quais tarefas devem ser migradas e qual o novo mapeamento de cada tarefa para que seja mitigado o desequilíbrio de carga.

3.2. Algoritmo

O SmartLB adota um *threshold* para definir um limite de desbalanceamento de carga aceitável. Caso o desbalanceamento seja maior que o *threshold* previamente definido, o algoritmo busca atingir o balanceamento levando em consideração a diferença de carga entre os cores com carga acima da média e abaixo da média. Quando o balanceador é chamado, ele usa informações disponibilizadas pelo Charm++ e calcula as cargas computacionais de cada core.

Quando a estratégia é aplicada, o algoritmo (linhas 1 a 3) busca informações sobre a quantidade de objetos mapeados em cada core e suas cargas para calcular o core com maior carga (PM), o core com menor carga (Pm) e a média das cargas (avg).

Tendo computado estes valores, a estratégia (linha 4) compara o desbalanceamento entre a carga do core mais carregado e o core menos carregado com o *threshold* . Caso essa razão for menor, o desbalanceamento de carga presente é menor do que o limite aceitável, assim nenhuma tarefa é migrada. Por outro lado, se a razão for maior que o threshold definido, o balanceador busca tarefas (linha 6) dos cores com cargas acima da média (linha 8) e realiza a migração desta tarefa para o core menos carregado (linha 13).

A estratégia (linha 11) também verifica antes de migrar se esta migração não deixará o core que recebe a tarefa com carga acima da média, para isso testa se a carga da tarefa é menor ou igual a diferença entre o core sub-carregado e core atual.

Algoritmo 1. Implementação do SmartLB.

```
1  PM = getCargaMaior();
2  Pm = getCargaMenor();
3  avg = getMediaCores();
4  if ((Pm/PM) > Threshold) {
5      for(i = 1; i <= nTarefas; i++) {
6          core = getCoreTarefa(i);
7          cc = getCargaCore(core);
8          if (cc >= avg) {
9              ct = getCargaTarefa(i);
10
11             if (ct <= (Pm - cc)) {
12                 numMigracoes++;
13                 migrarProcesso(i, core, Pm);
14                 Pm = getCargaMenor();
15             }
16         }
17     }
18 }
```

4. Metodologia

Nossos experimentos foram realizados em uma plataforma composta de um processador Intel Core i7-6500U com 4 núcleos físicos com 2 SMT/core, o que totaliza 8 núcleos de 2.5GHz.

A plataforma executa um sistema operacional Ubuntu Linux 18.04 com o kernel 4.15.0-23. Todos os benchmarks e balanceadores de carga foram compilados com o compilador GCC na versão 7.3.0. A versão do Charm++ utilizada para implementação foi a 6.5.1.

Para analisar o desempenho, a demanda de potência, o consumo de energia foi utilizada a ferramenta EMonDaemon [Padoin et al. 2014]. Esta ferramenta possibilita a coleta e análise do tempo de execução e da demanda de potência de cada processador durante a execução dos testes. Para realização dos testes a ferramenta EMonDaemon foi configurada para realizar medições de demanda de potência do processador a cada 1s.

Cada um dos testes realizados neste trabalho foram repetidos 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.

4.1.Benchmarks

Para avaliar o desempenho, demanda de potência e o consumo de energia do balanceador de carga proposto foram selecionados três benchmarks. Eles foram escolhidos devido à sua variada gama de padrões de comunicação e características da carga de trabalho. A descrição de cada benchmarks é apresentada na sequência:

- **Lb_test:** benchmark que apresenta desequilíbrio de carga onde pode-se usar diferentes padrões de comunicação. Neste artigo, foi utilizado um grafo de comunicação aleatória.
- **kNeighbor:** benchmark iterativo onde cada tarefa se comunica com "k"outras tarefas em cada etapa ou iteração. Seu padrão de comunicação é um anel.
- **ComprehensiveBench:** benchmark sintético configurável usado para criar aplicativos iterativos com diferentes parâmetros, como o número de tarefas, iterações, gráfico de comunicação, carga de tarefas e tamanhos de mensagens.

A Tabela 1 apresenta as características dos benchmarks e parâmetros utilizados em nossos experimentos. Diferentes frequências de balanceamento de carga foram escolhidas para diferentes aplicações, a fim de estabelecer um equilíbrio entre os benefícios das tarefas de remapeamento e os custos de mover tarefas entre cores bem como a computação de um novo mapeamento de tarefas. Nos testes com o balanceador de carga proposto foi adotado um *threshold* de valor igual a 5%.

Tabla 1. Parâmetros de entrada dos Benchmarks.

Benchmarks	Tarefas	Iterações	Frequência do LB
Lb_test	150	150	10
KNeighbor	150	150	10
ComprehensiveBench	50	30	5

4.2.Balanceadores de Carga

Os resultados de tempo de execução e a economia de energia alcançados com o BC proposto foram comparados com os balanceadores de carga GreedyLB, RefineLB e AverageLB.

5. Resultados

Nesta seção são apresentados as avaliações de desempenho, a redução do tempo de execução e o consumo de energia alcançado com o emprego do nosso balanceador de carga SmartLB na plataforma experimental apresentada na última seção.

Na Figura 1 são apresentados os tempos de execução dos testes realizados com o benchmark lb test e kNeighbor para diferentes quantidades de tarefas.

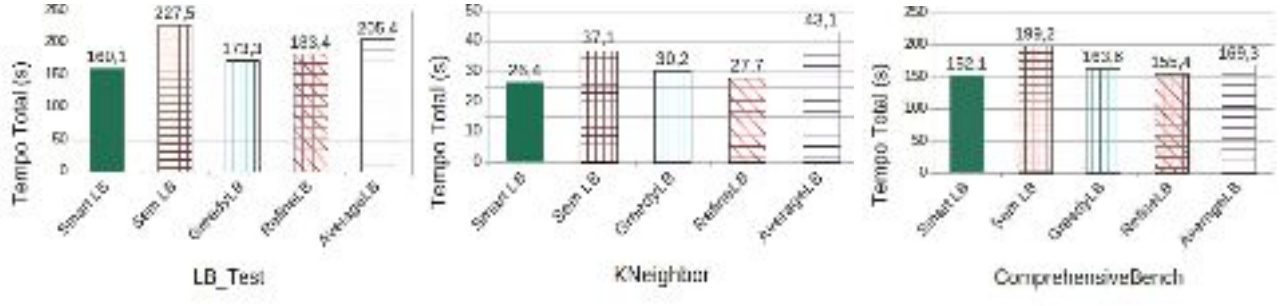

Fig. 1. Tempos de Execução mensurados durante a execução

O BC SmartLB apresentou melhor desempenho para ambos os benchmarks testados. Para lb test com 150 tarefas o SmartLB conseguiu reduzir o tempo de 227,5 para 160,1 segundos, o que representa uma redução de 29.6% em relação à execução sem balanceador.

Quando foi executado o benchmark kNeighbor, o SmartLB conseguiu uma redução significativa no tempo de execução. Com 150 tarefas o tempo total de execução foi reduzido de 37,1 para 26,4 segundos, o que representa uma redução de 28,8% em relação à execução sem balanceador. Para este benchmark, o SmartLB apresentou tempos 38,7% menor que o balanceador AverageLB e de 12,5% menor que o balanceador GreedyLB.

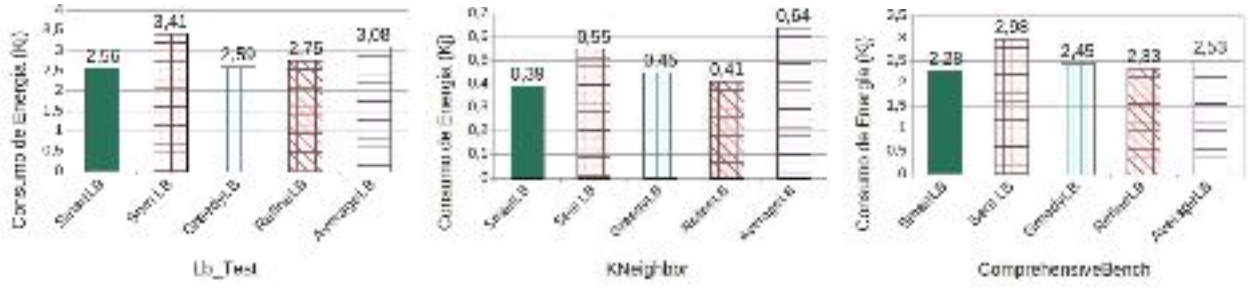

Fig. 2. Consumo de energia mensurado durante a execução dos benchmarks.

Uma redução equivalente e significativa foi também observada no consumo de energia quando aplicado o SmartLB com o benchmark kNeighbor. O consumo de energia foi reduzido em 39% em relação à execução sem balanceador, ou seja uma economia de 0,25KJ (de 0.64 KJ para 0.39 KJ).

O tempo de execução e o consumo total de energia das execuções também foi reduzido quando o SmartLB foi empregado na execução do benchmark ComprehensiveBench. O tempo total de execução apresentou uma redução de 23.4% em relação à execução sem balanceador. Para este benchmark, o uso da estratégia apresentou desempenho em média de 4,5% melhor que os alcançado pelos balanceador GreedyLB e RefineLB. Ganhos equivalentes foram observados no consumo de energia quando aplicado a nossa proposta.

Na Figura 3 são apresentados os desbalanceamentos de carga mensurados a cada chamada do balanceador de carga proposto. Em todos os testes, o SmartLB conseguiu concluir a execução dos benchmarks apresentando o menor de desbalanceamento.

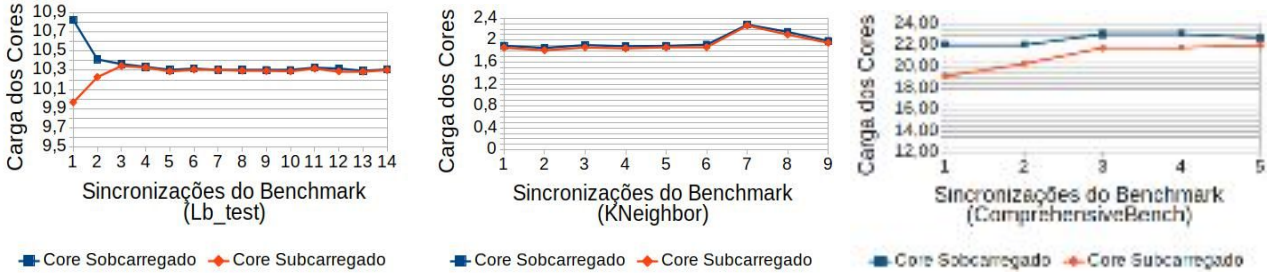

Fig. 3. Desbalanceamento mensurado durante a execução dos benchmarks.

Nos testes com o lb test, na primeira chamada do balanceador a diferenças entre o core mais carregado e o core menos carregado era de 0,9 unidades. O SmartLB conseguiu reduzir, até a última sincronização, essas diferenças para 0,007. Nos testes realizados com os benchmarks kNeighbor e ComprehensiveBench, o desbalanceamentos iniciais eram de 0,04 e 2,94 unidades. Mesmo assim, com a utilização do SmartLB, os desbalanceamentos foram reduzidos, finalizando a execução com 0,03 e 0,67 respectivamente.

6. Conclusões e trabalhos futuros

Este trabalho apresentou a proposta de um novo balanceador de carga denominado SmartLB. Nossos resultados demonstram que o uso do balanceador de carga SmartLB reduziu o tempo de execução e a quantidade de energia gasta quando aplicado em aplicações iterativas. O tempo total de execução e o consumo de energia foram reduzido em média 18,3% e 14,27% nos testes executados com três benchmarks. Os resultados alcançado pela nossa estratégia proposta foram melhores que os alcançados pelos balanceadores GreedyLB e RefineLB e AverageLB.

Como futuros trabalhos, pretende-se realizar melhorias no algoritmo de tomada de decisão do SmartLB de modo a melhorar o controle de migrações de tarefas. Pretende-se também realizar

testes em sistemas paralelos maiores utilizando problemas reais de computação científica bem como comparar com outros balanceadores de carga do estado da arte.

Agradecimentos

Trabalho apoiado por CNPq e CAPES com recursos do programa EU H2020 e do MCTI/RNP Brasil sob o projeto HPC4E de número 689772 e do edital de bolsa PIBIC da UNIJUI.

Referências

[1] Cybenko, G. (1989). Dynamic load balancing for distributed memory multiprocessors. Journal of parallel and distributed computing, 7(2):279–301.

[2] Padoin, E. L., Pilla, L. L., Castro, M., Boito, F. Z., Navaux, P. O. A., and Mehaut, J.-F. (2014). Performance/energy trade-off in scientific computing: The case of ARM big.LITTLE and Intel Sandy Bridge. IET Computers & Digital Techniques, 2(3):1–14.

[3] Pilla, L. L., Bozzetti, T. C., Castro, M., Navaux, P. O. A., and Méhaut, J.-F. (2015). Comprehensivebench: a benchmark for the extensive evaluation of global scheduling algorithms. Journal of Physics: Conference Series, 649(1):012007.

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