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: XXVI Seminário de Iniciação Científica

PROPOSTA DE UM BALANCEADOR DE CARGA PARA REDUÇÃO DE TEMPO DE EXECUÇÃO DE APLICAÇÕES EM AMBIENTES PARALELOS[1] PROPOSAL FOR A LOAD BALANCER TO REDUCE APPLICATION RUNTIME IN PARALLEL ENVIRONMENTS

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

[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 UNIJUÍ, vinirssantos@gmail.com
[3] Professor Orientador do Departamento de Ciências Exatas e Engenharias, padoin@unijui.edu.br
[4] Aludo do curso de Ciência da Computação e Bolsista PIBIC/UNIJUÍ, giovanerosalizott@hotmail.com
[5] Egresso do curso de Ciência da Computação UNIJUÍ, cleber.sartorio@hotmail.com

Palavras chave

HPC, Charm++, Balanceamento de Carga, Consumo de Energia, Benchmark, Desempenho.

Key Words

HPC, Charm++, Load Balance, Energy consumption, Benchmark, Performance.

Introdução

A medida que simulações computacionais mais complexas são desenvolvidas, aumenta a demanda por processamento dos sistemas de HPC. Um dos componentes de hardware que teve uma enorme evolução foi o processador, que passou a proporcionar a execução simultânea de aplicações em seus núcleos e execução simultânea de instruções com tecnologias *hyperthreading*. Assim, a programação paralela passa ser importante, possibilitando que aplicações sejam divididas em partes e executadas em paralelo nas unidades de processamento [Pilla e Meneses 2015].

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.

Evento: XXVI Seminário de Iniciação Científica

Balanceador de Carga SmartLB

Neste trabalho é apresentado uma proposta que 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.
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.
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 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 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 dos cores com cargas acima da média e realiza a migração desta tarefa para o core menos carregado.
A estratégia 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.

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.
Para avaliar o desempenho, demanda de potência e o consumo de energia do balanceador de carga proposto foram selecionados três benchmarks: Lb_test, KNeighbor e ComprehensiveBench. Eles foram escolhidos devido à sua variada gama de padrões de comunicação e características da carga de trabalho.
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

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.

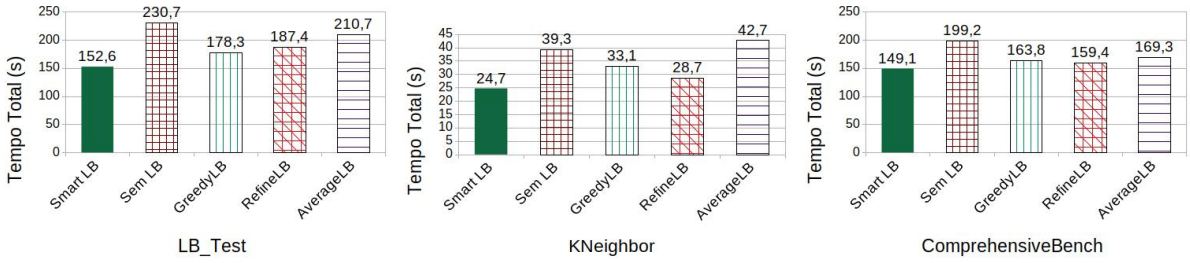

Fig. 1. Tempos de Execução mensurados durante a execução

O BC SmartLB apresentou melhor desempenho para ambos os benchmarks testados. Para lb test com 150 tarefas o SmartLB conseguiu reduzir o tempo de 230,7 para 152,6 segundos, o que representa uma redução de 45,6% 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 39,3 para 24,7 segundos, o que representa uma redução de 37,2% em relação à execução sem balanceador. Para este benchmark, o SmartLB apresentou tempos 42,2% menor que o balanceador AverageLB e de 25,4% menor que o balanceador GreedyLB.

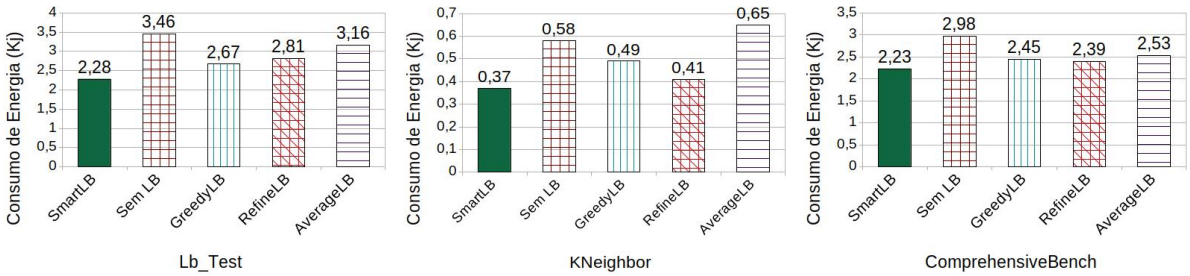

Fig. 2. Consumo de energia mensurado durante a execução dos benchmarks.

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 36,2% em relação à execução sem balanceador, ou seja uma economia de 0,19KJ (de 0.58 KJ para 0.37 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 25,6% em relação à execução sem balanceador. Para este benchmark, o uso da estratégia apresentou desempenho em média de 7,6% 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.

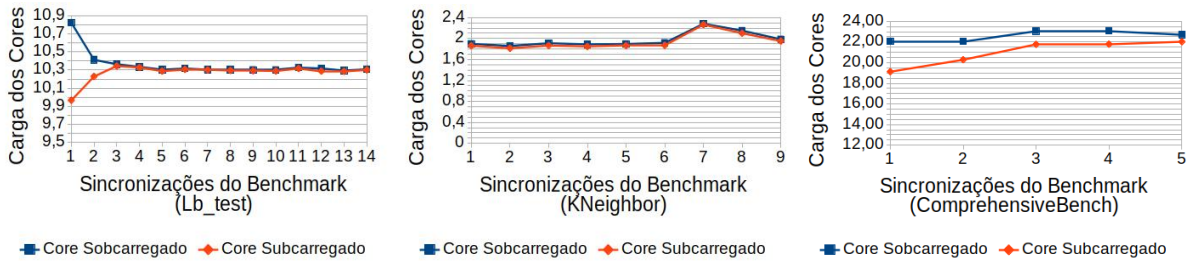

Fig. 3. Desbalanceamento mensurado durante a execução dos benchmarks.

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.

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 foi reduzido em média 22,5% 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

Pilla, L. L. and Meneses, E. (2015). **Programação paralela em charm++**. pages 1–20.

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.