Avaliando Consumo de Energia no Mapeamento de Processos*

Eduardo H. M. Cruz[1], **Edson L. Padoin**[2], **Francieli Z. Boito**[1], **Philippe O. A. Navaux**[1]

[1]Universidade Federal do Rio Grande do Sul (UFRGS) – Porto Alegre – RS – Brasil

[2]Universidade Regional do Noroeste do Estado do Rio Grande do Sul – Ijuí – RS – Brasil

{padoin}@unijui.edu.br, {ehmcruz, fzboito, navaux}@inf.ufrgs.br

1. Introdução

Nos últimos anos, o uso de arquiteturas multi-core cresceu rapidamente. Nessas arquiteturas, o subsistema de memória é o provedor de dados para as aplicações, além do meio pelo qual as *threads* que estão executando concorrentemente são capazes de se comunicar. Devido às diversas camadas de memória cache que podem estar presentes, existem diferenças no tempo necessário para trocar dados entre os *cores* de acordo com que níveis de memória eles compartilham. Nesse contexto, um correto mapeamento de *threads* em *cores* contribui para melhorar o desempenho da aplicação. Mas, além de possuir um bom desempenho, é desejável que o mapeamento seja o melhor do ponto de vista de potência consumida. Esse trabalho objetiva avaliar mapeamento estático de processos sob o ponto de vista do consumo energético.

2. Proposta

O mapeamento de processos é uma técnica que possibilita o aumento de desempenho em aplicações paralelas através de uma alocação mais eficiente dos recursos. Escalonando as *threads* que compartilham o mesmo espaço de memória em núcleos que compartilham uma mesma memória *cache* propicia um melhor desempenho de que quando nenhuma memória *cache* é compartilhada. Além disso, o tempo para dois núcleos dentro de um mesmo *chip* se comunicarem é menor do que quando as *threads* encontram-se em processadores separados. Essas diferenças no desempenho se devem ao fato de que além de um melhor aproveitamento de espaço nas memórias *cache*, um menor número de invalidações deverá ocorrer a cada modificação dos dados, ou seja, apenas os níveis superiores (mais próximos do processador) receberão os valores atualizados, reduzindo assim a sobrecarga imposta por protocolos de coerência. Dessa forma, as *threads* que mais compartilham memória devem ser escalonadas em núcleos mais próximos em relação à hierarquia de memória adotada.

Em um trabalho anterior [Cruz et al. 2010], foi apresentado um método que utiliza traços de acessos à memória feitos por uma aplicação para criar matrizes de compartilhamento entre as *threads* da mesma e um algoritmo de emparelhamento máximo de custo mínimo em grafos para determinar o melhor mapeamento. Os resultados mostrados no trabalho em questão (para o NAS Parallel Benchmarks[1]) indicam até 42% de ganho em relação ao escalonador nativo do sistema operacional.

A proposta consiste em aproveitar esses cálculos de mapeamentos para as aplicações do NAS para analisar o mapeamento do ponto de vista de consumo energético.

*Trabalho parcialmente apoiado por CNPq, CAPES e FAPERGS.

[1]http://www.nas.nasa.gov/Resources/Software/npb.html

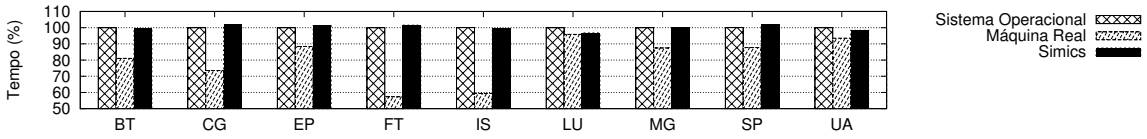

Figura 1. Tempo de execução normalizado da carga de trabalho NPB.

Para tal, optou-se por usar modelos de consumo de energia disponíveis na literatura (como os mostrados em [Šimunić et al. 1999] e [Hong et al. 2002]) e estatísticas dos acessos à memória durante a execução dos testes para obter estimativas de quanto cada escalonamento consome. Como esses dados de acessos não foram obtidos anteriormente, os testes precisam ser repetidos.

Entretanto, as máquinas reais não provêm mecanismos eficazes para monitoramento do ambiente de execução. O uso de simuladores permite, então, a extração de dados estatísticos relevantes para o mapeamento, tais como informações sobre o uso da memória cache, tempo médio que o processador fica esperando por dados, quantidade de invalidações geradas pela coerência da cache, entre outras. Para este trabalho, o simulador Simics[2] foi o escolhido como plataforma de testes.

Os gráficos da Figura 1 mostram os tempos normalizados obtidos pelo mapeamento para os benchmarks executando no Simics e na máquina real. Nota-se uma grande diferença entre eles, indicando que o Simics não simula adequadamente o comportamento da máquina real para mapeamento. Isso ocorre porque ele não implementa latências em barramentos, não havendo diferença, então, no custo da comunicação entre cores do mesmo processador e de processadores diferentes.

3. Conclusão e Trabalhos Futuros

Esse artigo apresentou uma proposta de trabalho que consiste em executar aplicações com diferentes estratégias de escalonamento de *threads* em núcleos de processamento usando o simulador Simics. Do simulador, pode-se obter detalhes sobre o uso da hierarquia de memória, que podem ser adaptados em modelos existentes de consumo energético para analisar o mapeamento sob esse aspecto. Não foi possível chegar a resultados sobre consumo energético, pois resultados preliminares mostram que o Simics não simula o mesmo comportamento observado na máquina real. Melhorias devem ser feitas no simulador antes que se prossiga com o estudo.

Referências

Cruz, E. H. M., Alves, M. A. Z., and Navaux, P. O. A. (2010). Process mapping based on memory access traces. In *WSCAD-SSC: XI Simpósio em Sistemas Computacionais*.

Hong, I., Kirovski, D., Qu, G., Potkonjak, M., and Srivastava, M. (2002). Power optimization of variable-voltage core-based systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(12):1702–1714.

Šimunić, T., Benini, L., and De Micheli, G. (1999). Cycle-accurate simulation of energy consumption in embedded systems. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 867–872. ACM.

[2]https://www.simics.net/