Impacto de técnicas de otimização de software em arquiteturas multicore e manycore*

Matheus S. Serpa, Eduardo H. M. Cruz, Philippe O. A. Navaux

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970, Porto Alegre – RS – Brasil

{msserpa, ehmcruz, navaux}@inf.ufrgs.br

Resumo. *A utilização de arquiteturas paralelas impõe diversos desafios para se obter um alto desempenho. A adaptação do código fonte para a arquitetura é uma opção para melhorar o desempenho de uma aplicação. Nossa proposta é aplicar e investigar o impacto de diferentes técnicas de otimização. Desta forma, será possível propiciar um melhor suporte para desenvolvedores, para que os mesmos tenham melhores condições de otimizar suas aplicações.*

1. Introdução

A preocupação com o gasto energético tem crescido com o objetivo de se atingir a computação *exascale* de forma sustentável. Entretanto, as tecnologias até então utilizadas não possibilitam atingir tal objetivo devido ao alto custo energético de se aumentar a frequência e estágios de *pipeline*, assim como a chegada nos limites de exploração do paralelismo a nível de instrução [Borkar and Chien 2011].

A fim de se solucionar tais problemas, arquiteturas paralelas e heterogêneas foram introduzidas nos últimos anos. Contudo, a utilização dessas arquiteturas impõe diversos desafios para se obter um alto desempenho. As aplicações precisam ser codificadas considerando as particularidades e restrições de cada ambiente, assim como considerando suas características arquiteturais distintas [Mittal and Vetter 2015].

Neste sentido, este artigo descreve a proposta de aplicação e avaliação do impacto de diferentes técnicas de otimização em *software*, a fim de se propiciar um melhor suporte para desenvolvedores, para que os mesmos tenham melhores condições de otimizar suas aplicações. A Seção 2 apresenta brevemente a metodologia a ser utilizada. Por fim, a Seção 3 apresenta a conclusão e os resultados esperados.

2. Metodologia

A análise dos trabalhos relacionados indica que uma classe importante de otimização de código envolve *blocking* para *caches*, reduzindo o uso da memória principal. Outras técnicas incluem a eliminação de operações de *gather* e *scatter* devido aos acessos irregulares. Em casos que os códigos não podem ser vetorizados (SIMD) devido a dependências, a técnica *loop interchange* podem ser utilizadas, permitindo a vetorização.

As técnicas citadas serão aplicadas no *benchmark* Rodinia [Che et al. 2009]. Esse *benchmark* foi escolhido devido as suas aplicações apresentarem características diferentes

*Este trabalho foi parcialmente financiado por recursos do projeto HPC4E (www.hpc4e.eu), financiamento nº 689772 do acordo internacional entre o programa H2020-EU e o MCTI/RNP-Brasil.

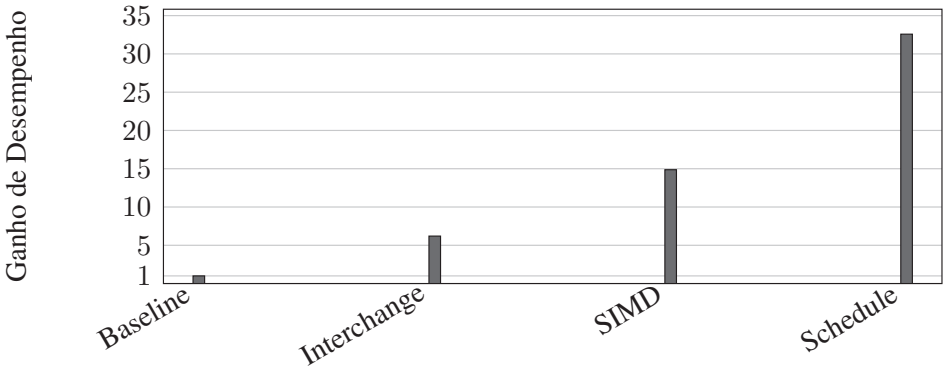

Figura 1. Ganho de desempenho de algumas técnicas combinadas.

e cargas de trabalho real. As arquiteturas alvo dessa pesquisa serão duas. Uma máquina Haswell, com dois processadores Intel Xeon E5-2640 v2, cada processador tem 10 *cores* físicos, permitindo execução de 40 *threads* com *Hyper-Threading*. Uma máquina Knights Corner, com um coprocessador Intel Xeon Phi 3120P com 57 *cores* com 4-SMT, executando até 228 *threads*.

Os experimentos estão sendo executados e são a média de 30 execuções aleatórias. O desvio padrão é calculado pela distribuição *t-student* com intervalo de confiança de 95%. Além de tempo de execução, dados de potência e energia estão sendo coletados com auxílio da ferramenta Intel PCM [Intel 2012].

3. Resultados Preliminares e Conclusão

A Fig. 1 apresenta resultados preliminares de ganho de desempenho de uma aplicação de propagação de onda executada na máquina Haswell. A *baseline* é a versão paralela original. As técnicas foram aplicadas de forma incremental, dessa forma, cada versão utiliza todas otimizações citadas anteriormente. Ganhos de desempenho de até 32.6x em relação a versão paralela original foram obtidos após as otimizações.

Ao final da pesquisa, espera-se listar as otimizações aplicadas em cada *benchmark* e caracterizar o impacto das mesmas apresentando o porquê da redução ou do aumento de tempo de execução. Além de *benchmarks* sintéticos, aplicações reais serão otimizadas baseando-se no estudo realizado.

Referências

[Borkar and Chien 2011] Borkar, S. and Chien, A. A. (2011). The future of microprocessors. *Communications of the ACM*, 54(5):67–77.

[Che et al. 2009] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S.-H., and Skadron, K. (2009). Rodinia: A benchmark suite for heterogeneous computing. In *IEEE International Symposium on Workload Characterization (IISWC)*, pages 44–54.

[Intel 2012] Intel (2012). Intel Performance Counter Monitor - A better way to measure CPU utilization.

[Mittal and Vetter 2015] Mittal, S. and Vetter, J. S. (2015). A survey of cpu-gpu heterogeneous computing techniques. *ACM Computing Surveys (CSUR)*, 47(4):69:1–69:35.