Análise do Consumo Energético de Aplicações Paralelas com Diferentes Versões de Compiladores

Arthur Mittmann Krause, Gabriel B. Moro, Lucas Mello Schnorr

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

{amkrause,gbmoro,schnorr}@inf.ufrgs.br

Resumo. *A escolha do compilador pode influenciar no consumo de energia de uma aplicação paralela. Neste trabalho é apresentada uma análise do impacto no consumo energético dos principais compiladores e níveis de otimização em aplicações com características diferentes. Os resultados indicam que a escolha do compilador deve ser realizada levando em consideração o tipo de aplicação, revelando o GCC 6.2.0 como o mais vantajoso de forma geral.*

1. Introdução

O consumo de energia é um assunto cada vez mais investigado em Computação de Alto Desempenho. Fatores como tipo de aplicação, balanceamento de carga, comunicação entre threads e outras características de plataforma impactam diretamente no tempo de execução e, consequentemente, no consumo de energia. Pode existir uma relação direta e proporcional entre tempo de execução e consumo de energia [Millani and Schnorr 2016]. Em geral, é difícil caracterizar se uma aplicação é limitada pelos acessos a memória ou pelo processador, uma vez que ela pode ter ambas as características ao mesmo tempo. Sendo assim, tenta-se calcular quais dos dois fatores (memória ou processador) limita de forma majoritária o desempenho da aplicação. Uma forma comum de realizar tal caracterização é utilizando valores obtidos por contadores de hardware. Medidas como instruções de ponto flutuante, taxa de cache misses, instruções por ciclo (IPC), entre outras, dão indícios se uma aplicação é limitada pela memória, processador ou por Entrada/Saída. Cada uma dessas categorias de programas faz uso distinto do hardware, portanto possuem um perfil de consumo de energia diferente.

O objetivo deste trabalho é analisar o impacto do consumo energético em diferentes versões de compiladores e diretivas de otimização em aplicações paralelas. Serão consideradas as aplicações NQueens [Duran et al. 2009], limitada pelo processador, e Graph500 [Murphy et al. 2010], limitada pela memória. A metodologia empregada é baseada em projeto experimental com uma rigorosa análise estatística e os principais resultados mostram que a escolha do compilador deve depender da aplicação.

O artigo está organizado em cinco seções. Trabalhos relacionados são apresentados na Seção 2. A metodologia e a plataforma em que o experimento foi executado são detalhados na Seção 3. A seção 4 apresenta a análise dos resultados obtidos. Enfim, a Seção 5 traz a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Bez et al. realizam uma análise de consumo de energia utilizando diferentes diretivas de otimização e frequências de processador em um cluster de processadores ARM

[Bez et al. 2016]. Nesse trabalho, os autores utilizam várias diretivas de otimização, sem variar o compilador e versão. Os resultados obtidos apresentam uma diferença significativa de desempenho e consumo de energia ao variar os níveis de otimização para uma mesma aplicação. Também no cenário de arquiteturas embarcadas, Lorenzon et al. analisam o desempenho e o consumo de energia de diferentes interfaces de programação paralela para algumas arquiteturas embarcadas [Lorenzon et al. 2015]. Os autores optaram por não utilizar nenhuma diretiva de otimização e apenas um compilador. Silveira et al. comparam a estimativa teórica de consumo de energia com a real, obtida por contadores de hardware. Os autores utilizam apenas a diretiva O3, não variando o compilador [Silveira et al. 2016]. Dentre os trabalhos investigados, não é possível encontrar uma análise de consumo de energia que explore diferentes compiladores e versões, com diferentes níveis de otimização para aplicações paralelas de memória compartilhada. Nesse sentido, o presente trabalho preenche esta lacuna, investigando o impacto no desempenho e na energia de uma aplicação quando se usa um determinado compilador/versão.

3. Metodologia e Plataforma Experimental

A metodologia utilizada nesse trabalho consiste na definição de um Projeto Experimental [Jain 1990]. Esse projeto possui como fatores as aplicações NQueens e Graph500, as versões dos compiladores Clang (3.8.0 e 3.9.0), GCC (4.4.7, 4.7.3, 5.4.0 e 6.2.0) e ICC (14.0.2, 16.0.4 e 17.0.1), os níveis de otimização (O0, O2 e O3) e a quantidade de threads (1, 6, 12, 18 e 24). Esses fatores foram escolhidos para permitir uma larga exploração dos valores. A carga de cada aplicação é dimensionada de forma a proporcionar um tempo de execução semelhante entre ambas, sendo $n = 15$ para o NQueens e dimensão 22 para a matriz utilizada na geração do grafo no Graph500. O experimento consiste em um projeto de fatorial completo, onde todas as combinações de fatores são executadas, empregando-se replicação. Cada combinação é executada dez vezes em ordem aleatória, fazendo com que anomalies afetam estatisticamente todos os experimentos de forma homogênea.

As ferramentas auxiliares utilizadas para configurar e conduzir o experimento foram o Spack e o Intel PCM. Spack [Gamblin et al. 2015] é um gerenciador de pacotes projetado especialmente para centros de supercomputação que permite a instalação de múltiplas versões e configurações de um software na mesma máquina sem conflitos. Através dele foram instaladas as versões 5.4.0 e 6.2.0 do GCC, além do Clang 3.8.0 e 3.9.0. Intel PCM [Willhalm et al. 2012] é uma ferramenta da Intel que através da leitura de contadores de hardware, fornece medidas como o consumo de energia na memória e no processador durante a execução de uma aplicação. O experimento foi executado na máquina `Orion3` do Instituto de Informática da UFRGS, que possui dois processadores Intel® Xeon® E5-2630 (12 cores no total), com os tamanhos de cache L1/L2/L3 192/1536/15360KB, sendo que somente a de nível três é compartilhada entre os cores, e 32GB de memória DDR3 1600MHz. O sistema operacional da máquina é o Ubuntu 12.04.5 LTS com a versão do Linux 3.13.0-85-generic.

4. Resultados

Apresentamos apenas um sumário dos resultados, por questões de espaço. A Figura 1 detalha o consumo de energia (eixo Y) por tempo de execução (eixo X), apenas para as aplicações paralelas compiladas com as últimas versões do Clang, GCC e ICC, e também o GCC 4.7.3. Cada ponto no gráfico representa a média entre as 10 execuções, enquanto

as elipses indicam as regiões de confiança de 95%. Somente a média com 12 e 24 threads é apresentada, sendo que a segunda representa a ativação de *hyperthreading*. Na grande maioria dos casos, o consumo energético é proporcional ao tempo de execução, portanto os compiladores que resultam em um melhor desempenho também proporcionam um menor consumo de energia. O caso em que se observa o maior desbalanceamento nessa relação é quando compara-se o Graph500 compilado com o ICC e as versões mais antigas do GCC (caso A no gráfico). O tempo de execução dessas versões com 24 threads é semelhante, porém o consumo de energia com o ICC é maior, principalmente na versão 16.0.4 (que não aparece na figura) onde a diferença chega a quase 10%. As causas da alta variabilidade das medidas com o ICC ainda estão sendo investigadas, mas sabe-se que a distribuição dos valores é gaussiana. Com 24 threads, é possível observar que o compilador que gera o maior consumo de energia é o Clang 3.9.0 (B). Em contraste, o compilador que apresentou menor consumo de energia foi o GCC 6.2.0 (C). Para a aplicação NQueens, o compilador GCC 4.7.3 apresentou o maior consumo de energia em todas as threads (pontos D). Além disso, percebe-se que conforme aumenta-se o paralelismo, os resultados entre as diferentes versões aproximam-se, indicando uma redução na influência do compilador utilizado nesse caso. Nota-se também que a aceleração com mais threads do que cores físicos é praticamente negligível, enquanto o consumo de energia aumenta de forma considerável. Isso se deve à característica da aplicação que, por ser limitada pelo processador, não se beneficia tanto do SMT [Li et al. 2005]. O compilador que apresentou o melhor resultado foi novamente o GCC 6.2.0 (ponto E).

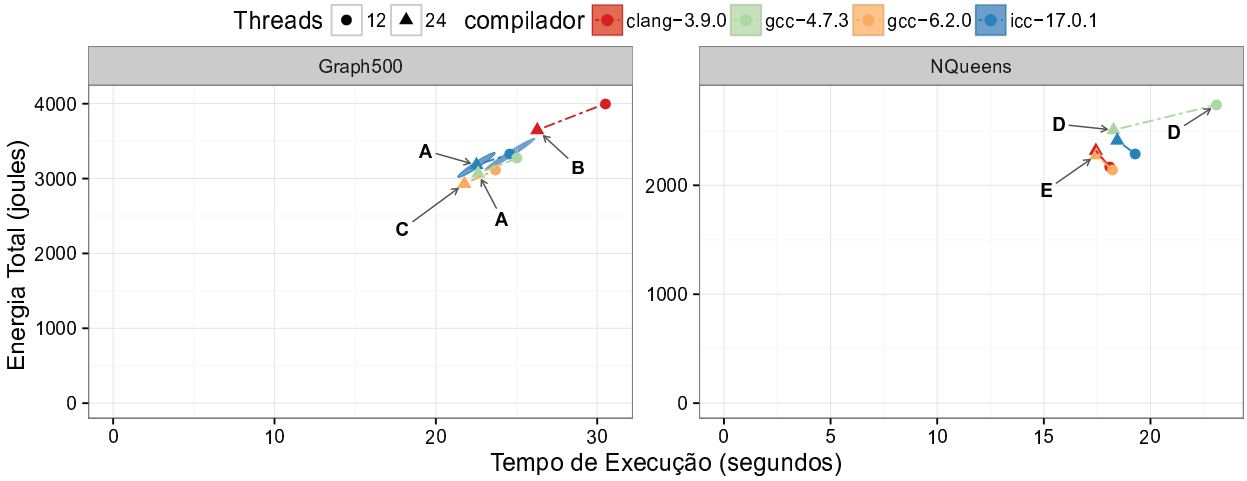

Figura 1. Energia/Tempo das aplicações usando quatro compiladores em O3.

5. Considerações Finais

O objetivo desse trabalho é comparar diferentes versões de compiladores, com diferentes níveis de otimização, analisando o impacto no consumo de energia em duas aplicações paralelas. Foram escolhidas duas aplicações com características diferentes: o Graph500, que depende mais de memória, pois possui uma taxa de cache misses alta, e o NQueens, que é mais limitada pelo processador pois possui uma baixa taxa de misses e um IPC mais alto. Com os resultados adquiridos foi possível visualizar que para as duas aplicações utilizadas, o compilador que apresentou a melhor relação entre consumo de energia e tempo de execução foi o GCC 6.2.0, sendo essa a versão mais recente do compilador GCC. Também foi possível constatar uma tendência de melhora das versões desse compilador,

com exceção da versão 4.7.3 que para determinados casos pode ser pior que suas antecessoras. A partir desse trabalho, é possível compreender que a escolha do compilador deve ser realizada levando em consideração o tipo de aplicação, pois ao variarmos a aplicação, novas vantagens e desvantagens são observadas nas diferentes versões de compiladores.

Como trabalho futuro, pretende-se investigar as diferentes políticas de frequência com a técnica *Dynamic Voltage and Frequency Scaling*. A partir dos resultados, será possível analisar o impacto dos diferentes versões de compiladores, níveis de otimização e políticas de frequência para aplicações limitadas mais pela memória ou por processameto.

Referências

[Bez et al. 2016] Bez, J. L., Bernart, E. E., dos Santos, F. F., Schnorr, L. M., and Navaux, P. O. A. (2016). Performance and energy efficiency analysis of hpc physics simulation applications in a cluster of arm processors. *Concurrency and Computation: Practice and Experience*.

[Duran et al. 2009] Duran, A., Teruel, X., Ferrer, R., Martorell, X., and Ayguadé, E. (2009). Barcelona openMP tasks suite: A set of benchmarks targeting the exploitation of task parallelism in openMP. *Proceedings of the International Conference on Parallel Processing*, pages 124–131.

[Gamblin et al. 2015] Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., de Supinski, B. R., and Futral, S. (2015). The spack package manager: Bringing order to hpc software chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 40. ACM.

[Jain 1990] Jain, R. (1990). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley.

[Li et al. 2005] Li, Y., Skadron, K., Brooks, D., and Hu, Z. (2005). Performance, energy, and thermal considerations for smt and cmp architectures. In *11th International Symposium on High-Performance Computer Architecture*, pages 71–82. IEEE.

[Lorenzon et al. 2015] Lorenzon, A. F., Sartor, A. L., Cera, M. C., and Beck, A. C. S. (2015). The Influence of Parallel Programming Interfaces on Multicore Embedded Systems. *Proceedings - International Computer Software and Applications Conference*, 2:617–625.

[Millani and Schnorr 2016] Millani, L. F. and Schnorr, L. M. (2016). Computation-aware dynamic frequency scaling: Parsimonious evaluation of the time-energy trade-off using design of experiments. In *Euro-Par 2016: Parallel Processing Workshops: Euro-Par 2016 International Workshops, Grenoble, France, August 22-23, 2016, Revised Selected Papers*. Springer International Publishing.

[Murphy et al. 2010] Murphy, R. C., Wheeler, K. B., Barrett, B. W., and Ang, J. A. (2010). Introducing the graph500. *Cray User's Group (CUG)*.

[Silveira et al. 2016] Silveira, D. S., Moro, G. B., Navaux, P. O. A., Schnorr, L. M., and Bampi, S. (2016). Energy Consumption Estimation in Parallel Applications: an Analysis in Real and Theoretical Models. pages 124–135.

[Willhalm et al. 2012] Willhalm, T., Dementiev, R., and Fay, P. (2012). Intel Performance Counter Monitor - A better way to measure CPU utilization.