Um Estudo do Impacto das Frequências de Operação do *uncore* na execução de Aplicações Paralelas*

Leonardo K. Sonco[1], Marcelo C. Luizelli[1], Fábio D. Rossi[2], Arthur F. Lorenzon[1]

[1]Universidade Federal do Pampa – Alegrete – RS – Brazil

[2]Instituto Federal Farroupilha - Campus Alegrete – Alegrete – RS – Brasil

leonardosonco.aluno@unipampa.edu.br

Resumo. *Nos dias atuais, o consumo de energia está se tornando uma das principais pautas no desenvolvimento de novos sistemas computacionais. Nesse sentido, diferentes estratégias têm sido desenvolvidas para reduzir o consumo de energia sem degradar o desempenho. Neste trabalho, é demostrando que o ajuste da frequência dos componentes uncore e CPU pode reduzir o consumo de energia em determinadas aplicações paralelas em aproximadamente 15%, tendo nenhum ou quase mínimo impacto no desempenho.*

1. Introdução

Os componentes de *hardware* têm se tornado cada vez mais velozes com o objetivo de melhorar o desempenho dos sistemas computacionais. No entanto, esse fator também pode acarretar no aumento do consumo de energia e temperatura de operação devido ao aumento na dissipação de calor proviniente das altas frequências do processador. Por conta disto, empresas tecnológicas acabam tendo uma boa parte do seu orçamento destinado para a conta de energia [Kepes 2015]. Portanto, novas estratégias têm sido desenvolvidas para preservar, ou ainda, melhorar o desempenho dos sistemas computacionais sem impacto negativo no consumo de energia.

Uma destas estratégias consiste da utilização de *softwares* controladores de frequência de operação da CPU e dos componentes *uncore* (que corresponde à funções de um processador que não são manipuladas pelo núcleo, por exemplo a memória *cache* L3). Tais *softwares*, como por exemplo, *LIKWID Performance Tools*, têm como objetivo possibilitar ao usuário configurar, conforme for necessário, a frequência da CPU e *uncore*. Nesse sentido, quando uma aplicação com comportamento *memory-bound* está sendo executada, uma alternativa pode ser aumentar a frequência do *uncore* e reduzir a frenquência da CPU. Por outro lado, para aplicações CPU-Bound, reduzir a frequência do *uncore* e aumentar a frequência da CPU permite acelerar a aplicação com redução no consumo de energia por parte do sistema de memória [Schwarzrock et al. 2020, Marques et al. 2021].

Poucos trabalhos têm avaliado o emprego da frequência dos componentes *uncore* em conjunto com modificações na frequência da CPU. Em [dos Santos Marques et al. 2017] a frequência dos componentes da CPU são modificadas em conjunto com a frequência da memória para otimizar o consumo de energia. Em [André et al. 2020], é proposta a utilização do DUF (*dynamic uncore frequency*), um processo *daemon* que adapta dinamicamente a frequência do *uncore* com a finalidade de reduzir o consumo de energia de uma aplicação, tendo um limite na possível degradação do desempenho. Já em [Sundriyal et al. 2018a], é usado o UFS (*uncore frequency scaling*) e DVFS *(dynamics voltage and frequency scaling*) para comparar quanto ao seu potencial

*Este trabalho foi parcialmente financiado pela FAPERGS nos projetos 19/2551-0001224-1, 19/2551-0001689-1 e 17/2551-0001193-7 e PROBIC-FAPERGS

de economia de energia por meio de experimentos em um aplicativo de química quântica *GAMESS*.

Considerando que a área de pesquisa ainda é incipiente, este artigo tem como objetivo avaliar o desempenho e consumo de energia de aplicações paralelas quando são executadas com diferentes configurações de frequência da CPU e do *uncore*. Para tanto, consideramos a execução de oito aplicações paralelas de diferentes características com relação ao uso de CPU e memória compartilhada no processador *Intel Xeon Silver 4214R*. Através da variação dos três principais *governors* DVFS disponíveis no Sistema Operacional Linux (e.g., *performance*, *ondemand* e *powersave*) e das frequências disponíveis do componente *uncore* (e.g., 1.2GHz, 1.8GHz e 2.4GHz), mostramos que o ajuste da frequência *uncore* em determinadas aplicações reduz o consumo de energia em até 15.69% sem impacto negativo no desempenho.

2. Fundamentação Teórica

Processadores multicore modernos utilizam da combinação de técnicas de *hardware* e *software* para controlar o consumo de energia do sistema. Um exemplo é o uso de DVFS, uma técnica que visa reduzir o consumo de energia ajustando dinamicamente a tensão e a frequência da CPU e *uncore* com mínimo impacto no desempenho. O DVFS fornece *governors* para controlar como ocorrerá o aumento e diminuição da frequência. Cada governor possui uma estratégia diferente para mudar a escala de frequência: No *governor ondemand*, a frequência do processador é modificada de acordo com a carga de trabalho da CPU. Isto é, quando há grande consumo do processador a frequência é aumentada, e caso contrário, a frequência é reduzida. Já no *governor performance*, a frequência da CPU é definida ao máximo buscando o melhor desempenho. Por fim, no *governor powersave*, a frequência é definida no menor nível possível para atingir a menor potência possível.

Por outro lado, o componente *uncore* descreve as funções de um processador que não são controladas pelo núcleo, como por exemplo a memória *cache L3*. A partir do processador *Intel Haswell*, os domínios de frequência de núcleo e *uncore* foram desacoplados [Sundriyal et al. 2018b]. Sendo assim, tais processadores possuem dois domínios de velocidade de *clock*, sendo possível modificar a frequência *uncore* sem alterar a frequência de operação da CPU, e vice versa. A frequência *uncore* tem um impacto significativo quando se trata em melhorar o desempenho das aplicações, porém, possui uma importância ainda maior quando o assunto é diminuir o consumo de energia. A alteração da frequência *uncore* pode ser feita através de *softwares* de terceiros, que por meio de comandos a modificação se torna simples e de fácil compreensão, e.g., o *LIKWID Performance Tools*.

3. Metodologia

Benchmarks. Foram utilizadas oito aplicações paralelas: **Três** *kernels* e pseudo-aplicações do *NAS parallel benchmark*: ep.C.x, lu.C.x e sp.C.x; **Cinco** aplicações de domínio geral: *The High Performance Conjugate Gradients* (HPCG), Método de Jacobi, *Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics* (LULESH), *n-body*, e *Stream* (ST). As aplicações possuem diferentes características com relação ao uso de CPU e memória compartilhada. Por exemplo, Jacobi, ST e sp.C.x são aplicações que fazem uso intensivo do barramento *off-chip bus*. Já as aplicações *n-body*, ep.C.x e lu.C.x fazem uso intensivo da CPU após realizar o carregamento dos dados da memória. Por outro lado, as aplicações HPCG e LULESH possuem carga de trabalho balanceada, alternando fases CPU- e *memory-bound*.

Ambiente de Execução. Os experimentos foram realizados no processador *Intel(R) Xeon(R) Silver 4214R*, que possui 12 núcleos físicos e 24 *threads*, além de memória

cache L3 de 16,5 MB. Nós consideramos três níveis de frequência *uncore* (1.2GHz, 1.8GHz e 2.4GHz) e três diferentes *governors* (*ondemand, performance e powersave*), totalizando 9 configurações distintas por aplicação. Para alterar a frequência do *uncore* nós utilizamos o *software LIKWID* através do comando *"likwid-setFrequencies –umin xx –umax xx"*, que estipula uma frequência máxima e mínima para o *uncore*. O sistema operacional utilizado foi o Linux Ubuntu, com o *kernel* v.5.10 com a versão do GCC,conjunto de compiladores de linguagens de programação ,foi a v.10.2. O consumo de energia foi obtido através do Intel RAPL enquanto que o tempo de execução foi obtido com a função *omp_get_wtime*. Cada configuração (aplicação, frequência de *uncore* e *governor* foi executada 10 vezes com um desvio padrão menor que 1.0%.

4. Resultados Experimentais

A Figura 1 apresenta os resultados de consumo de energia (eixo *y*) e tempo de execução (eixo *x*) para cada configuração (frequência da CPU e *uncore*) em cada aplicação. Portanto, quanto mais próximo da origem (*x=0* e *y=0*), melhor é o resultado. Por exemplo,a configuração *Psave-2.4* executou a aplicação *Jacobi* em 251 segundos com um consumo de energia de $19,78x10^3$ joules.

Para aplicações que fazem uso intensivo do *off-chip bus* (*Jacobi, STREAM e sp.C.x*),a alteração da frequência da memória faz com que ocorra uma redução marginal no tempo de execução, pois há aumento da largura do barramento de dados e aumento da velocidade da interface de dados. No entanto, como o principal fator que limita a escalabilidade destas aplicações é o barramento de comunicação e não a memória, aumentar a frequência do *uncore* leva a um maior consumo de energia. Sendo assim a melhor configuração de energia encontrada para esse tipo de aplicação é a *Psave-1.8* na aplicação *STREAM*, onde houve apenas uma perda marginal no desempenho. No entanto foi possivel economizar 12.4% de energia em relação a configuração que possui o melhor desempenho e o pior gasto de energia *Psave-2.4*.

Em aplicações que possuem fases balanceadas de uso intensivo da CPU e memória de dados para leitura e escrita (*n-body*, ep.C.x e lu.C.x), aumentar a frequência do *uncore* acelera a execução com nenhum ou quase mínimo impacto no consumo de energia.Por exemplo na aplicação *n-body* mesmo quando colocado a configuração que possui o melhor desempenho, *Perf-2.4*, há um redução no consumo de energia de 4.9% se comparado com a configuração *Perf-1.8*, tornando mais vantajoso manter a frequência *uncore* em 2.4 GHz. Por fim, nas aplicações onde o acesso a memória se dá inicialmente para leitura e apenas ao final da computação para a escrita dos dados (*HPCG e LULESH*), quando é aumentado a frenquência do *uncore* não ocorre uma interferência considerável na melhoria do desempenho. Por exemplo na aplicação HPCG, a configuração *Perf-2.4* melhora apenas 0.43% em comparação com a *Perf-1.8*. Este comportamento acontece pela razão de não haver um uso intenso da memória. No entanto o consumo de energia é aumentado em 13.5%. De uma maneira resumida, os melhores resultados são obtidos com a frequência do uncore em 1.8GHz pois as aplicações são balanceadas.

5. Conclusão

Este trabalho apresentou uma análise sobre o impacto da frequência *uncore* no desempenho e consumo de energia de aplicações paralelas. Por meio da execução aplicações com diferentes características, foi demonstrado que, ao executar uma aplicação com a configuração ideal (frequência do *uncore* e CPU),é possível chegar a uma redução de até 15.69% no consumo de energia sem impacto no desempenho quando comparado a configurações que buscam o melhor desempenho. Para trabalhos futuros, pretendemos avaliar o comportamento do *uncore* quando o grau de exploração do paralelismo da aplicação paralela mudar.

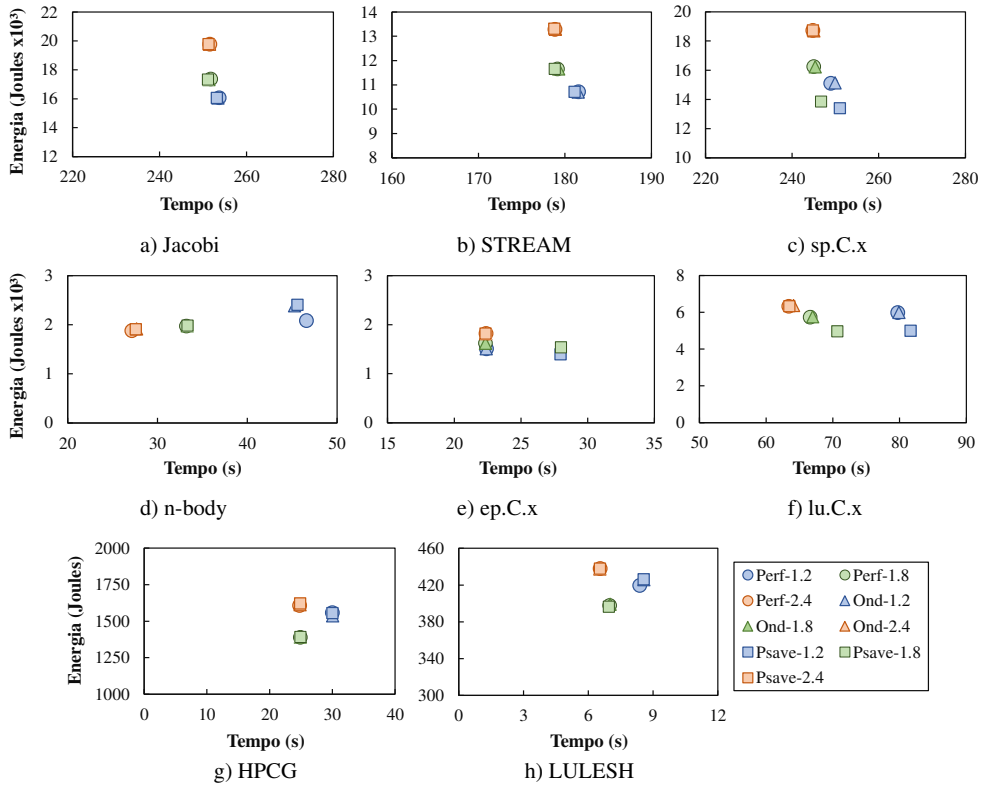

Figura 1. Tempo de execução (eixo x) e consumo de energia (eixo y) de cada configuração para cada aplicação

Referências

André, E., Dulong, R., Guermouche, A., and Trahay, F. (2020). DUF : Dynamic Uncore Frequency scaling to reduce power consumption. working paper or preprint.

dos Santos Marques, W., de Souza, P. S. S., Lorenzon, A. F., Beck, A. C. S., Rutzig, M. B., and Rossi, F. D. (2017). Improving edp in multi-core embedded systems through multidimensional frequency scaling. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE.

Kepes, B. (2015). Thirty per cent of servers are sitting ”comatose”according to research.forbes.

Marques, S. M., Medeiros, T. S., Rossi, F. D., Luizelli, M. C., Beck, A. C. S., and Lorenzon, A. F. (2021). Synergically rebalancing parallel execution via dct and turbo boosting. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 277–282.

Schwarzrock, J., de Oliveira, C. C., Ritt, M., Lorenzon, A. F., and Beck, A. C. S. (2020). A runtime and non-intrusive approach to optimize edp by tuning threads and cpu frequency for openmp applications. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1713–1724.

Sundriyal, V., Sosonkina, M., Westheimer, B., and Gordon, M. (2018a). Comparisons of core and uncore frequency scaling modes in quantum chemistry application gamess.

Sundriyal, V., Sosonkina, M., Westheimer, B., and Gordon, M. (2018b). Core and uncore joint frequency scaling strategy. *Journal of Computer and Communications*, 06:184–201.