Avaliação do Desempenho e Consumo de Energia de Aplicações Implementadas com OpenMP SIMD

Gustavo L. Tamiosso[1], Gustavo P. Berned[1], Marcelo C. Luizelli[1], Fábio D. Rossi[2], Jorji Nonaka[3] e Arthur F. Lorenzon[1]

[1]Laboratório de Otimização de Sistemas - Universidade Federal do Pampa Campus Alegrete (UNIPAMPA) - Alegrete – RS – Brasil

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

[3]RIKEN Center for Computational Science, Japan

gustavotamiosso.aluno@unipampa.edu.br

Resumo. *Explorar o paralelismo tornou-se uma alternativa para reduzir o consumo de energia e melhorar o desempenho de aplicações. Este trabalho tem como objetivo avaliar as estatísticas de desempenho e energia de aplicações implementadas utilizando operações vetoriais. Através da execução de quatro aplicações conhecidas, mostramos que o uso de operações vetoriais pode reduzir o tempo de execução em até 37% e reduzir o consumo de energia em 35%.*[1]

1. Introdução

A exploração do paralelismo no nível de *threads* tem sido amplamente utilizada para melhorar o desempenho de processadores com vários núcleos. No entanto as arquiteturas adquiriram a habilidade de realizar instruções de forma vetorial, com o uso de registradores embutidos em sua construção. Esta computação, também conhecida como SIMD (*single instruction, multiple data*), utiliza unidades vetoriais para acelerar operações aritméticas que são particularmente realizadas sob estruturas de dados vetoriais.

Diferentes otimizações e extensões para compiladores e interfaces de programação paralela (e.g., *Open Multiprocessing* – OpenMP) têm sido desenvolvidas para facilitar o uso de operações SIMD por parte dos programadores. O OpenMP permite aos programadores explorarem estas instruções através do uso da diretiva *#pragma omp simd*, que indica ao compilador quais regiões de código devem ser aplicadas a vetorização. Esta diretiva pode ser utilizada em uma região sequencial ou associada à regiões paralelas (interna a regiões *#pragma omp parallel for*). Neste último, cada *thread* irá explorar o uso de operações vetoriais em seu núcleo de processamento [Chapman et al. 2007].

Sendo assim, este trabalho tem como objetivo principal comparar o desempenho e consumo de energia de aplicações implementada com OpenMP SIMD. Para tanto, nós utilizamos quatro aplicações com diferentes características relacionadas às estruturas de dados utilizadas e operações vetoriais/matriciais. Através da execução das aplicações em um processador AMD Ryzen 9 com suporte a execução de até 24 *threads* e operações AVX-2 (256-bits), nós mostramos que comparando o melhor resultado obtido com e sem o uso de operações SIMD, o tempo de execução pode ser reduzido em até 37% e o consumo de energia em 35%.

[1]O presente trabalho teve financiamento parcial pela FAPERGS nos projetos 19/2551-0001224-1 e 19/2551-0001689-1, CNPq PIBIC e FAPERGS PROBIC.

2. Trabalhos Relacionados

Os trabalhos propostos por [Inoue 2016], [Ponte et al. 2017], puderam demonstrar que ao se fazer uso de instruções do tipo SIMD, há significativas reduções de *load* e *store*. Além do mais, em [Ponte et al. 2017], demonstrou-se que há ganhos de desempenho de até 6.3 vezes ao se utilizar as diretivas OpenMP SIMD em comparação a autovetorização realizada apenas pelo compilador.

Em [Abbo 2004], utilizando processadores Xetal SIMD e os escalando (adicionando unidades de processamento), foi demonstrado que com o uso do paralelismo é possível reduzir gastos de energia. Ainda assim, a redução é intrínseca com a complexidade dos algoritmos, pois com um número específico de unidades de processadores a diferença se estabiliza.

Já em [Jakobs and Rünger 2018], cinco implementações de um mesmo algoritmo (eliminação Gaussiana) foram executadas em três processadores Intel Core i7 de especificações similares mas arquiteturas diferentes. Observou-se que cada processador se distingue pelas diferentes (porém semanticamente parecidas) instruções para *stores* e *loads* da API AVX (Um conjunto de instruções SIMD da Intel). Eles demonstraram que inclusive entre as próprias instruções SIMD com as condições certas, pode ser reduzido o tempo de execução e o consumo de energia de aplicações.

O trabalho tem como diferencial o fato de realizar os testes a fim de comparar o uso ou não de instruções vetoriais. Além disso foi levado em consideração tanto o desempenho quanto à eficiência energética.

3. Metodologia

Quatro aplicações com diferentes características foram selecionadas: **CFD Solver**, consiste em um *solver* de volume finito para *grids* não estruturadas para equações tridimensionais de Euler [Che et al. 2009]. **Hotspot**, é uma aplicação baseada em uma *grid* estruturada, utilizada para estimar a temperatura do processador com base no *floorplan* da arquitetura e em medidas de consumo de potência [Che et al. 2009]. **Método de Jacobi**, um algoritmo iterativo para determinar a solução de sistemas lineares envolvendo uma grande porcentagem de coeficientes zero. **Decomposição LU (LUD)**, que consiste de um algoritmo para calcular as soluções de um conjunto de equações lineares, seu kernel decompõe uma matriz em um produto entre uma matriz triangular superior e uma inferior [Che et al. 2009]. De maneira geral estes testes são realizados utilizando vetores de tipos numéricos para adquirir os dados do *profile*.

Os experimentos foram realizados em um ambiente com processador AMD Ryzen 9 3900x contendo 12 núcleos e 24 *threads* com 32GB de memória e com Sistema Operacional Ubuntu. O compilador utilizado foi o GCC 9.2.0 com flags de otimização *-O3* e *-fopenmp-simd* (aplicações que exploram SIMD). Cada aplicação foi executada 10 vezes com diferentes números de *threads*: 1, 2, 4, 6, 8, 10, 12, 24 com o *governor* do DVFS (*Dynamic voltage and frequency*) mantido em *ondemand* e com as threads OpenMP pinadas, a fim de não superaquecer o processador e seus componentes.

4. Resultados Experimentais

As Figuras 1 e 2 apresentam os resultados de desempenho e consumo de energia para as quatro aplicações em função do número de *threads*. De uma maneira geral, pode-

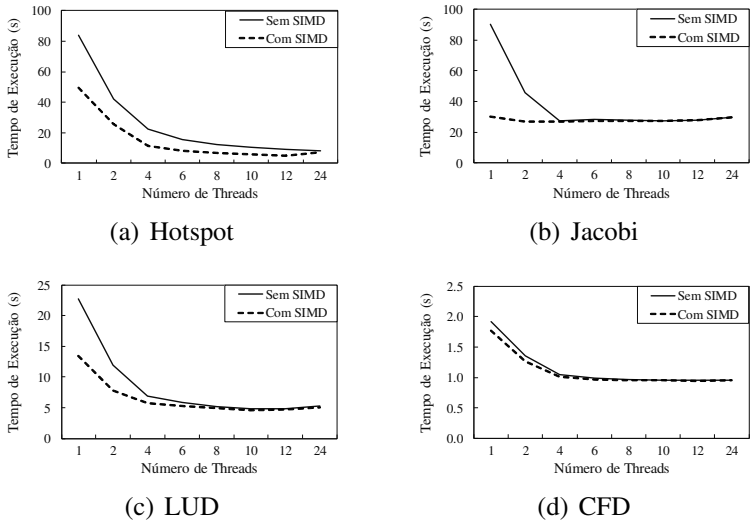

Figura 1. Tempo de execução das aplicações sob diferentes números de *threads*

.

mos observar que a exploração de operações SIMD proporcionaram um melhor desempenho e menor consumo de energia, quando comparado a execução das aplicações sem operações SIMD, na maioria dos casos. É possível notar na Figura 2.a que se pode melhorar em 41.17% do valor total quando utilizando um único *thread*. Se mantém um Δt (i.e. variação de tempo) relevante, porém se reduz a partir da utilização de **4 *threads*** enquanto que a partir dos 12 *threads* a diferença se reduz a um nível desprezível. Já nas Figuras 2.b e 2.c, percebe-se uma redução de 66.66% em Jacobi e 38.88% em LUD ambos quando executam com uma única *thread*. Possuem signficativo Δt até **4 *threads***. Por fim, na Figura 2.d a utilização de SIMD mostra uma aceleração de 3.84% quando utilizando 1 *thread*, tendo algum diferencial até **4 *threads***, a partir disso deixa de ter um Δt visível.

Quando o consumo de energia é considerado, a diferença do uso de SIMD aumenta. Na Figura 3.a se mantém um diferencial de até 42.85% com 1 *thread*, sendo estável até a execução com 12 *threads*. Na Figura 3.b, há uma redução considerável na diferença do consumo de energia (64.27% de com 1 *thread*) conforme o número de *threads* aumenta. Novamente, na Figura 3.c, o ΔW (i.e. diferença de energia) tem seu pico em 38.88% com 1 *thread*, tornando a se reduzir a partir de **4 *threads***. Por outro lado, na Figura 3.d, pode-se observar um ganho de eficiência energética acerca de 3.50% até **4 *threads***. Após isso, ocorre uma redução na diferença em até 8 *threads*, onde nota-se uma diferença negativa de 2.56%, ou seja, torna-se ineficiente energeticamente, retornando a um ΔW = 0 aos 12 *threads*.

É perceptível a redução, tanto de tempo quanto ao gasto de energia. Nas análises foi possível verificar que em todos os casos, até a execução com 4 *threads* o uso de operações SIMD traz ganhos pertinentes. Dentre os testes, o CFD possui o menor aproveitamento das operações SIMD, devido a sua natureza. Em contrapartida, no Hotspot é possível ver resultados estáveis independente do número de *threads* sendo utilizado.

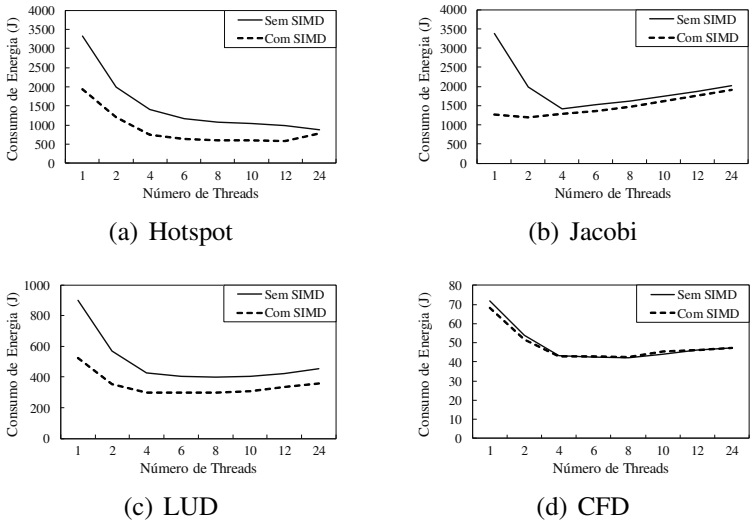

Figura 2. Consumo de energia das aplicações sob diferentes números de *threads*

5. Conclusão

Com base nos resultados apresentados, é possível notar que a utilização dos registros vetoriais pode resultar em uma redução de energia significativa e uma melhoria no desempenho de determinadas aplicações. Entretanto, existem situações específicas onde o uso das operações SIMD tornam a aplicação ineficiente, tanto no quesito de consumo de energia quanto no tempo de execução, sendo necessário uma análise do projeto de software visando os prós e contras, a fim de tomar uma decisão a adotar ou não o uso de operações vetoriais.

Referências

Abbo, A. A. e. a. (2004). Power consumption of performance-scaled simd processors. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pages 532–540. Springer Berlin Heidelberg.

Chapman, B., Jost, G., and Pas, R. V. D. (2007). *Using OpenMP - Portable Shared Memory Programming*.

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 Int. Symp. on Workload Characterization*, pages 44–54, DC, USA. IEEE Computer Society.

Inoue, H. (2016). How simd width affects energy efficiency: A case study on sorting. In *COOL CHIPS XIX*, pages 1–3.

Jakobs, T. and Rünger, G. (2018). On the energy consumption of load/store avx instructions. In *FedCSIS*, pages 319–327.

Ponte, C., González-Domínguez, J., and Martín, M. J. (2017). Evaluation of openmp simd directives on xeon phi coprocessors. In *HPCS*, pages 389–395.