Comparando o Tempo de Execução e Consumo de Energia de Aplicações Compiladas com GCC e PGI

Leonardo C. de Lima[1], **Luan Pereira**[1], **Fábio D. Rossi**[2], **Marcelo C. Luizelli**[1] **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

leonardolima.aluno@unipampa.edu.br

Resumo. *OpenACC é uma API que facilita muito o trabalho de quem atua na area de High Performance Computing.Neste sentido é importante ter conhecimento de qual compilador oferece melhor resultado. Assim, este trabalho compara os compiladores PGI e GCC mostrando que as aplicações selecionadas executam 31% mais rápido e consomem 25% menos energia com PGI que aplicações selecionadas compiladas com GCC.*[1]

1. Introdução

Na última década, as GPUs se tornaram dispositivos populares e presentes em arquiteturas que buscam prover o melhor desempenho possível. Desta maneira, o constante crescimento da comunidade de alto desempenho trouxe a necessidade de criação de ferramentas capazes de facilitar a exploração de paralelismo em tais arquiteturas [Reyes et al. 2013]. Uma destas ferramentas criadas é o OpenACC [OpenACC 2019], que através da inserção de diretivas de compilador no código fonte, tornou o desenvolvimento de aplicações paralelas mais simples.

Muito embora o uso de diretivas tenha facilitado o desenvolvimento de aplicações paralelas que executam em arquiteturas GPUs, uma outra variável tem se tornado importante quando se fala em obter o melhor desempenho e menor consumo de energia. Diferentes compiladores podem ser utilizados para a geração de códigos binários para executar em tais GPUs, como por exemplo, *The Portland Group* (PGI) e *GNU Compiler Collection* (GCC). Neste sentido, é importante ter conhecimento de qual compilador oferece o melhor resultado [Grillo et al. 2014].

Diferentes trabalhos têm avaliado a aplicação de OpenACC em diferentes compiladores. [Reyes et al. 2013] faz uma comparação entre PGI, CAPS OpenACC e accULL utilizando o benchmark EPCC, mostrando que em tempo de compilação o PGI foi 6 vezes mais rápido que accULL e 1,8 vezes mais rápido que CAPS. [Barba et al. 2017] apresenta *TORMENT*, uma ferramenta de análise que mostra relatórios de tempo de execução, desvio padrão e resultados de códigos sequenciais e CUDA, de códigos com OpenACC compilados com diferentes compiladores.

Neste sentido, este trabalho se torna complementar aos demais, pois objetiva comparar o desempenho e consumo de energia [Lorenzon et al. 2015] de aplicações OpenACC

[1]Este trabalho foi parcialmente financiado pela FAPERGS nos projetos 19/2551-0001224-1 e 19/2551-0001689-1, CNPq PIBIC e FAPERGS PROBIC.

compiladas com os compiladores PGI e GCC. Através da execução de quatro *kernels* do NAS *Parallel Benchmark* em uma GPU, nós mostramos que as aplicações compiladas com PGI podem ser até 31% mais rápidas que quando compiladas com GCC. Adicionalmente, elas consumiram até 25% menos energia.

2. Fundamentação Teórica

Inicialmente lançado em 2011 e desenvolvido pela organização sem fins lucrativos OpenACC.org, o OpenACC é um padrão de programação paralela que visa a praticidade em código, assim diminuindo o esforço do usuário na construção de códigos paralelos. Ele suporta C, C++ e FORTRAN [OpenACC 2019]. Utiliza-se de diretivas de compilador, divididas em regiões que são partes do códigos delimitadas por *pragmas* que desempenham certas funções [Wolfe 2010]. A região de dados é responsável pela movimentação dos dados entre o *host* e o dispositivo, e a região de computação é utilizada para a execução do código em paralelo no dispositivo [Alghamdi and Eassa 2019].

O compilador PGI baseia-se em diretivas *pragma* do OpenACC 2.6 para a programação em GPUs [PGI 2019]. Ele automaticamente analisa a estrutura do código e a separa em porções que serão executadas na CPU e GPU. Com a utilização de diretivas, o compilador mapeia o código identificando regiões suscetíveis à otimização. Elas também podem oferecer controle sobre alocação e hierarquia de memória [PGI 2019].

No compilador GCC, o código é transformado para a *independent abstract syntax tree*(AST). Então a AST é transformada para *register transfer language* (RTL), uma linguagem de baixo nível utilizada pelo GCC. Após isso o compilador gera o código Assembly a partir do RTL [GCC 2008]. GCC implementa as diretivas pragma do OpenACC em sua versão 2.5 [GCC 2019].

3. Metodologia

Os experimentos foram realizados utilizando 4 kernels da suite do *NAS Parallel Benchmarks*, implementados em OpenACC [Xu et al. 2015]: BT, CG, SP e EP. Utilizando uma máquina com o processador AMD Ryzen 7 1700, os testes foram realizados em uma GPU de modelo GeForce GT 1030, com 384 CUDA Cores, máximo de 1024 threads por bloco, dimensão de máximo de bloco 1024x1024x64, frequência de 1468 MHz e tamanho da cache L2 de 524288 bytes. As aplicações foram compiladas com PGI 19.10, utilizando as flags de compilação -acc -ta=tesla,cc60 -O3 -mcmodel=mediun e com GCC 9.2 utilizando as flags -fopenacc -lm -O3 -mcmodel=mediun -fPIC. Cada configuração (*kernel* e binário) foi executado cinco vezes.

4. Resultados

As Figuras 1 e 2 apresentam os resultados de desempenho e consumo de energia para todos os experimentos, respectivamente. O desvio padrão das execuções foi inferior a 0,5% do resultado obtido em cada configuração.

Considerando a média aritmética geral do tempo de execução de todos os *kernels*, as aplicações compiladas com PGI foram 31% mais rápidas que as aplicações compiladas com o GCC: O kernel BT compilado com PGI foi 54% mais rápido que sua versão em GCC; CG 85% mais rápido em PGI, SP 17% mais rápido em PGI e por fim EP 10% mais rápido em PGI.

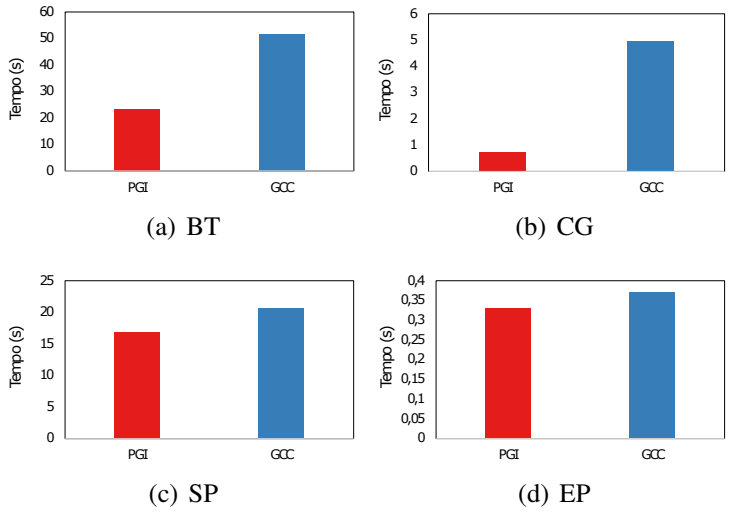

Figura 1. Tempo de execução Médio de cada aplicação

Quando consideramos o consumo de energia, as aplicações compiladas com PGI em média geral do consumo de energia consumiram 25% menos energia do que as compiladas com GCC: BT compilado com PGI tendo gasto 51% menos energia, CG gastando 84% menos energia, SP 14% menos energia gasta, e por fim, EP gastando 11% menos energia.

A partir dos resultados obtidos, é possível observar que o compilador PGI mostrou um melhor desempenho e menor consumo que o GCC em aplicações paralelizados com o OpenACC utilizando aceleradores, devido as otimizações realizadas pelo compilador presentes na flag -O3 como *Branch to branch elimination*, *Constant propagation*, *Copy propagation* etc... estas não realizadas pelo GCC em -O3. . Contudo, é válido ressaltar que o PGI é um produto desenvolvido pela NVIDIA, portanto possui um suporte nativo para o uso de bibliotecas como OpenACC, que é desenvolvido pela mesma empresa, enquanto o GCC só recebeu suporte a partir de sua versão 7.0. Sendo assim, o PGI

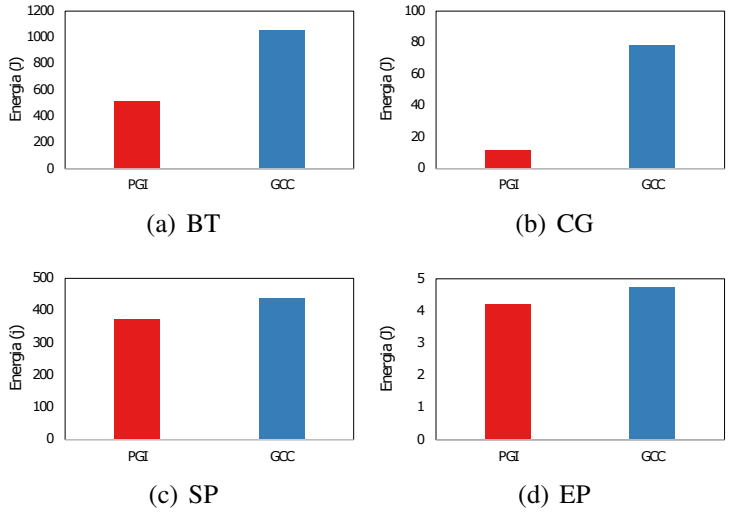

Figura 2. Consumo de Energia Médio de cada aplicação

mostrou-se uma ferramenta mais adequada para aplicações que utilizam o OpenACC.

5. Conclusão

Este trabalho comparou quatro *kernels* implementados com OpenACC, compilados utilizando PGI e GCC. Utilizando métricas como consumo de energia e o tempo de execução de cada aplicação. Foi possível constatar a partir dos resultados obtidos que PGI superou GCC em todas as comparações feitas neste trabalho. Baseando-se nos resultados obtidos neste experimento é possível concluir que por não ser sua ideia inicial implementar OpenACC e ainda não possuir suporte completo da API, GCC se mostra inferior quando se trata do desempenho utilizando. Como trabalhos futuros busca-se compreender outras aplicações, ambientes e métricas.

Referências

Alghamdi, A. M. and Eassa, F. E. (2019). Openacc errors classification and static detection techniques. *IEEE Access*, 7:113235–113253.

Barba, D., Gonzalez-Escribano, A., and Llanos, D. R. (2017). Torment openacc2016: A benchmarking tool for openacc compilers. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 246–250.

GCC (2019). https://gcc.gnu.org/wiki/OpenACC.

GCC, t. G. C. C. (2008). Structure of gcc. https://gcc.gnu.org/wiki/StructureOfGCC.

Grillo, L., Reyes, R., and d. Sande, F. (2014). Performance evaluation of openacc compilers. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 656–663.

Lorenzon, A. F., Cera, M. C., and Beck, A. C. S. (2015). Performance and energy evaluation of different multi-threading interfaces in embedded and general purpose systems. *Journal of Signal Processing Systems*, 80(3):295–307.

OpenACC (2019). About openacc. https://www.openacc.org/about.

PGI (2019). Pgi compilers and tools for high performance computing. https://www.pgroup.com/products/index.htm.

Reyes, R., López, I., Fumero, J. J., and de Sande, F. (2013). A preliminary evaluation of openacc implementations. *The Journal of Supercomputing*, 65(3):1063–1075.

Wolfe, M. (2010). Implementing the pgi accelerator model. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, pages 43–50. ACM.

Xu, R., Tian, X., Chandrasekaran, S., Yan, Y., and Chapman, B. (2015). Nas parallel benchmarks for gpgpus using a directive-based programming model. In Brodman, J. and Tu, P., editors, *Languages and Compilers for Parallel Computing*, pages 67–81, Cham. Springer International Publishing.