Implementação da aplicação CFD para multi-CPU e multi-GPU com a interface HPSM

Daniel Di Domenico[1], **João V. F. Lima**[1]

[1]Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

{ddomenico,jvlima}@inf.ufsm.br

Resumo. *O presente trabalho visa implementar a aplicação CFD através da API HPSM, executando-a em um ambiente composto por multi-CPU e multi-GPU. Os experimentos possuem o intuito de avaliar o desempenho utilizando o processamento heterogêneo e simultâneo em ambas arquiteturas. Os resultados obtidos indicaram que a aplicação alcançou melhora de desempenho combinando CPUs e GPUs, superando o uso de apenas aceleradores (GPU).*

1. Introdução

A dinâmica de fluidos é um importante campo de pesquisa, tanto na indústria como na academia. Por isso, existem muitas aplicações nesta área para executar tarefas como a identificação de padrões no clima ou a simulação do comportamento dos fluxos de ar sobre aeronaves [Prugger et al. 2016]. Neste cenário, o paralelismo pode ser considerado uma alternativa para melhorar o desempenho destas aplicações, principalmente aproveitando o poder computacional dos aceleradores.

O objetivo deste trabalho é implementar, através da interface HPSM, a aplicação CFD (*Computational Fluid Dynamics*) empregando laços paralelos voltados a execução em multi-CPU e multi-GPU. Por meio de experimentos, pretende-se verificar se ocorrem ganhos de desempenho utilizando o processamento heterogêneo e simultâneo em CPUs e GPUs em comparação ao uso de apenas aceleradores (GPUs).

2. Interface HPSM, aplicação CFD e plataforma

A HPSM [Di Domenico and Lima 2016] é uma interface de programação C++ que oferece recursos para a construção de programas paralelos direcionados ao processamento em CPUs e GPUs. Suas principais características são a execução de laços e reduções paralelas simultaneamente em ambas arquiteturas, processamento por meio de diferentes *back-ends* como Serial, OpenMP e StarPU, portabilidade do código C++ entre os *back-ends* e as diferentes arquiteturas, além do uso de transferências de dados implícitas.

A CFD foi a aplicação utilizada nos experimentos deste estudo, onde adaptou-se a versão disponível na *suite* de benchmarks Rodinia[1]. Ela, que soluciona um problema de dinâmica de fluidos, é caracterizada por ser de grade não estruturada e de volume limitado, aplicando as equações de Euler sobre um fluido compressível em um ambiente de três dimensões [Che et al. 2010]. A entrada utilizada foi uma asa de aeromodelo do tipo NACA0012 em fluxo supersônico.

[1]Rodinia: https://www.cs.virginia.edu/~skadron/wiki/rodinia/index.php/Rodinia:Accelerating_Compute-Intensive_Applications_with_Accelerators

A plataforma onde os experimentos foram executados é uma máquina NUMA Dell PowerEdge T630 equipada com dois processadores Intel Xeon E5-2697 v3 de 2,60 GHz com 14 núcleos (totalizando 28 *cores*). O computador também possui quatro GPUs NVIDIA Titan X. Para as execuções com Kaapi, utilizou-se a *runtime* LIBKOMP.

3. Resultados

Os resultados apresentados na Figura 1 (média de 30 execuções) indicam que a aplicação CFD obteve escalabilidade com apenas CPUs para os três *back-ends* utilizados. Com somente GPUs (0 *thread*), também houve ganhos a medida que mais GPUs estavam disponíveis. Estes ganhos cresceram nas combinações de CPUs e GPUs, sendo o melhor desempenho alcançado com 4GPUs + 8 *threads* (*speedup* de 56,6). No entanto, observam-se quedas a partir de 12 (com 2 GPUs) e 10 (com 3 e 4 GPUs) *threads*. Sugere-se que a razão destas quedas seja a falta de afinidade de dados da *runtime* STARPU para a arquitetura NUMA, pois ela não aloca estes dados no mesmo nó (*socket*) onde está sendo executada a *thread* que irá acessá-los.

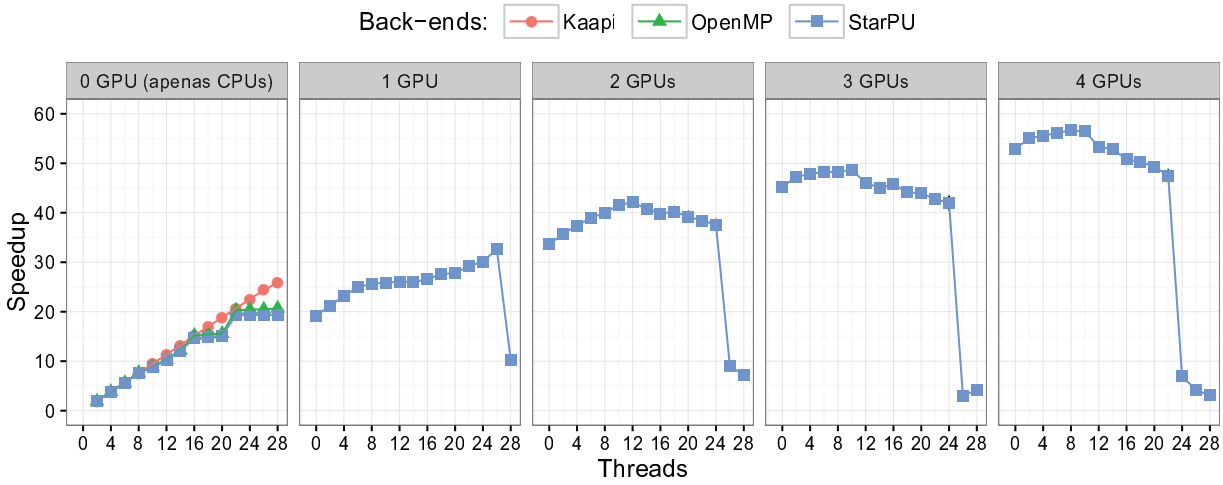

Figura 1. *Speedup* com GPUs e *threads*. Tamanho: 131.072. Bloco: 2.048. Iterações: 100.

Apesar das quedas, os experimentos mostraram que o processamento simultâneo em CPUs e GPUs resultou em melhoria do desempenho da aplicação CFD em comparação com o uso de apenas aceleradores (GPUs). Na continuidade deste trabalho, planeja-se realizar novos experimentos com a aplicação variando-se o tamanho da entrada e o do bloco para analisar se os ganhos são mantidos.

Referências

Che, S., Sheaffer, J. W., Boyer, M., Szafaryn, L. G., Wang, L., and Skadron, K. (2010). A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads. In *Proceedings of the IEEE IISWC'10*, pages 1–11.

Di Domenico, D. and Lima, J. (2016). Uma API em linguagem C++ para programas com laços paralelos e suporte a multi-CPUs e multi-GPUs. In *Proc. of the XVII Simpósio em Sistemas Computacionais de Alto Desempenho - WSCAD*, pages 100–111.

Prugger, M., Einkemmer, L., and Ostermann, A. (2016). Evaluation of the partitioned global address space (PGAS) model for an inviscid Euler solver. *Parallel Computing*. http://dx.doi.org/10.1016/j.parco.2016.11.001.