Avaliação de Desempenho da Arquitetura CUDA com o Benchmark Embarrassingly Parallel*

Laércio Lima Pilla[1], **Philippe Olivier Alexandre Navaux**[1]

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

Abstract. *Graphics processors (GPUs) are being used as accelerators for general-purpose computations (GPGPU). The CUDA architecture is an example of this technology. This architecture has differentiated characteristics and, despite the several applications that were already ported for it, there is no consensus about which kinds of applications can take profit of its potential performance. Based on this issue, this paper presents an initial evaluation of the CUDA architecture with the use of the Embarrassingly Parallel benchmark from NAS. The results show speedups up to 21 times with CUDA when compared to the use of the OpenMP parallel library, what indicates a possible compatibility between CUDA and this kind of application.*

Resumo. *Processadores gráficos (GPUs) vem sendo agora usados como aceleradores para computações de propósito geral (GPGPU). A arquitetura CUDA é um exemplo desta tecnologia. Esta arquitetura possui características diferenciadas e, apesar das diversas aplicações já portadas para ela, não há um consenso sobre quais tipos de aplicações podem tirar proveito de seu potencial desempenho. Partindo desta questão, este artigo apresenta uma avaliação inicial da arquitetura CUDA com o uso do benchmark Embarrassingly Parallel do NAS. Os resultados mostram speedups de até 21 vezes na versão para CUDA quando comparada a versão original em OpenMP, o que indica uma possível compatibilidade entre a arquitetura CUDA e este tipo de aplicação.*

1. Introdução

Uma alternativa atual para aumentar o desempenho de sistemas de computação envolve o uso de processadores gráficos como aceleradores paralelos para computações de propósito geral (**GPGPU**). A arquitetura **CUDA** (*Compute Unified Device Architecture*)[nvi 2009] é um exemplo de destaque entre as GPGPUs. Diversas aplicações já foram portadas para esta arquitetura, como algoritmos de alinhamento de sequências [Schatz et al. 2007] e dinâmica de fluídos [Tölke 2007]. Porém, a arquitetura apresenta aspectos diferentes dos processadores encontrados usualmente nos computadores, como uma hierarquia de memória complexa e limitações de precisão. Além disso, as aplicações nela executadas necessitam de características como alto potencial de paralelismo de dados e intensidade aritmética. Contudo, não há um consentimento sobre quais aplicações podem tirar proveito do desempenho de CUDA, sendo inviável testar todas as aplicações possíveis.

Neste contexto, um estudo sobre o comportamento de aplicações executando no ambiente CUDA necessitaria beneficiar-se da existência de modelos paralelos e

*Pesquisa parcialmente financiada pela empresa Microsoft

conjuntos de *benchmarks* representativos, com o intuito de evitar esforços repetitivos e expandir os resultados. Uma alternativa encontra-se no uso dos ***NAS Parallel Benchmarks*** (NPB) [Bailey et al. 1994], utilizados para a avaliação de sistemas paralelos, e na classificação ***Dwarf Mine*** [Asanovic et al. 2006], que organiza métodos numéricos segundo seus padrões de comportamento. Baseado nessas idéias, este artigo apresenta uma avaliação inicial do desempenho da arquitetura CUDA com o *benchmark* ***Embarrassingly Parallel*** (EP) dos *NAS Parallel Benchmarks*, o qual se encontra incluso na categoria ***Monte Carlo*** na classificação *Dwarf Mine*.

2. CUDA - *Compute Unified Device Architecture*

CUDA [nvi 2009] é tanto um arquitetura de processador gráfico da empresa NVIDIA, quanto uma biblioteca para o desenvolvimento de aplicações de propósito geral que executam nesta arquitetura. Algumas características da arquitetura são apresentadas ao programador de forma opaca (não transparente), o qual deve otimizar questões como o uso da hierarquia de memória e tamanho dos blocos de *threads*.

A Figura 1(a) ilustra a hierarquia de processamento de CUDA. As *threads* são organizadas em blocos, os quais são organizados em uma grade. A grade corresponde a um *kernel*, função que executa em paralelo na GPU. Fisicamente, a GPU possui multiprocessadores (SM), cujo número varia conforma o modelo da placa gráfica. Cada SM contém 8 núcleos (SPs) controlados pela mesma Unidade de Instruções, ou seja, que executam a mesma instrução. Um bloco de *threads* é mapeado para execução em um SM, assim como uma grade acaba mapeada para uma GPU. O número de *threads* ou blocos pode ser maior do que a capacidade de execução em paralelo da GPU.

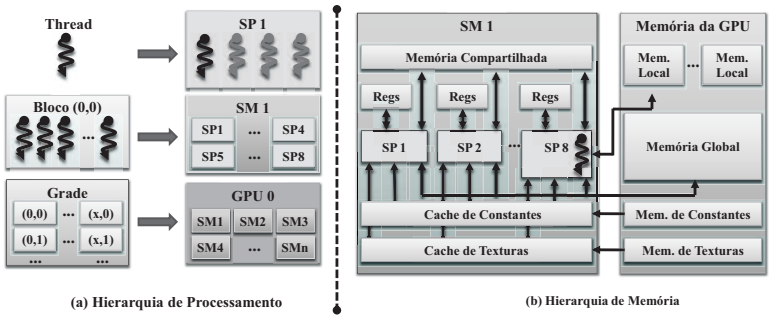

Figura 1. Características da arquitetura CUDA.

A hierarquia de memória de CUDA é apresentada na Figura 1(b). Um SM possui memórias de baixa latência e pequeno tamanho. Dentro dele, cada SP possui seu próprio banco de registradores. Além disso, há uma memória compartilhada entre seus SPs e *caches* para as memórias somente de leitura (constantes e texturas) . A memória principal da GPU, de grande tamanho e alta latência, compreende uma memória global compartilhada entre todos SMs e uma memória local por *thread*. Caso *threads* consecutivas façam acesso a uma área contígua da memória global, estes acessos podem ser coalescidos (agregados) em uma só leitura da memória, aumentando a banda passante.

3. *NAS Parallel Benchmarks*

Os NPB [Bailey et al. 1994] são um conjunto de perfis desenvolvidos para a avaliação de desempenho de sistemas paralelos. Entre suas qualidades, estão sua representatividade, inclusão de códigos de verificação e medição de desempenho (tempo e milhões de operação por segundo, ou Mops), gratuidade e a existência de múltiplos tamanhos de problema. Atualmente, este conjunto inclui 11 *benchmarks*. Uma característica importante dos NPB é que todas as operações de ponto flutuante devem usar precisão dupla.

O *benchmark* EP (*Embarrassingly Parallel*) foi escolhido para implementação para CUDA. Ele gera pares de números aleatórios baseados em desvios gaussianos [Bailey et al. 1994]. Esse *benchmark* é utilizado para estimar o pico de desempenho alcançável com ponto flutuante de precisão dupla em sistemas. O *benchmark* EP encontra-se na categoria *Monte Carlo* na classificação *Dwarf Mine* [Asanovic et al. 2006]. Ele apresenta alta intensidade aritmética, regularidade nas computações e nos acessos à memória, independência de dados e comunicação quase ausente.

4. Metodologia

Os experimentos comparam o desempenho entre a versão 2.3 do *benchmark* EP na linguagem C com OpenMP e a versão implementada para CUDA. Eles foram executados para as classes W, A e B do *benchmark*, os quais tratam da geração de 2^{26}, 2^{29} e 2^{31} números aleatórios, respectivamente. O ambiente de testes inclui um processador Intel Core 2 Duo E8500 ($3,16$ GHz), 4 GB de memória e uma GPU GeForce GTX 280, executando o SO Ubuntu 9.04 32 bits. A versão OpenMP foi compilada com o gcc versão 4.3.1 e executa 2 *threads*. A versão para CUDA foi compilada com $nvcc$ versão 0.2.1221 e utiliza o driver CUDA 2.3. Ambas versões utilizam a otimização $-O3$. Os resultados apresentam uma confiança de 95% e erro relativo de 10%, com um mínimo de 6 execuções.

5. Resultados

A Figura 2(a) apresenta os tempos de execução obtidos para as duas versões do *benchmark*. O eixo horizontal representa o tempo em segundos e o eixo vertical as diferentes classes do problema. As linhas ligando os extremos das barras representam o *speedup* obtido. As interrupções nas colunas significam que seus valores estão acima do comportado pelo gráfico. Quanto menor o tempo, melhor o desempenho. Na figura, pode-se ver uma grande diferença entre as duas versões. Tem-se um ganho de $15\times$ utilizando CUDA para a menor classe (W). Esta diferença cresce para até $21\times$ para a classe B.

A quantidade de números aleatórios gerados por segundo no testes (operações consideradas pelo *benchmark* na métrica Mops) é apresentada na Figura 2(b). Através da figura, fica visível que a vazão de operações mantém-se constante para as diferentes execuções com OpenMP. Isto significa que mesmo com a menor instância do *benchmark*, já se utiliza todos os recursos da CPU. Diferentemente, na versão para CUDA, este valor cresce conforme o tamanho da instância. Isto acontece porque, mesmo com o crescimento das instâncias, os sobrecustos da transferência de memória entre CPU e GPU mantém-se constantes. Além disso, a instância W não oferece paralelismo o bastante para utilizar todos os recursos da GPU, o que já acontece a partir da instância A.

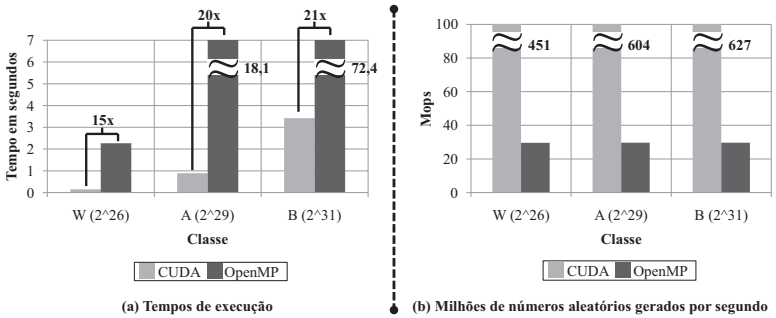

Figura 2. Medições de desempenho do *benchmark* EP.

6. Conclusões

Este artigo apresentou uma avaliação inicial de desempenho da arquitetura CUDA com o *benchmark Embarrassingly Parallel* - integrante da categoria *Monte Carlo* na classificação *Dwarf Mine*. Obtiveram-se ganhos de até 21 vezes no uso da GPU para este *benchmark*, quando comparado ao uso dos dois núcleos do processador com OpenMP. Isso foi possível graças a regularidade das computações e acessos à memória, além da independência de dados. Quanto maiores as instâncias do problema, melhores foram os resultados, pois os custos de transferência entre as memórias da CPU e da GPU são absorvidos pelo tempo de execução do *kernel*. Esses resultados indicam uma compatibilidade entre os algoritmos pertencentes a categoria *Monte Carlo* e a arquitetura CUDA.

Trabalhos futuros incluem a implementação de outros *benchmarks* do NAS, pertencentes a diferentes categorias da classificação *Dwarf Mine*, para uma melhor compreensão de quais tipos de aplicação podem tirar proveito do potencial desempenho da arquitetura CUDA. Além disso, considera-se a expansão do estudo para outras arquiteturas, podendo compará-las com os resultados já obtidos para CUDA.

Referências

(2009). Nvidia compute unified device architecture (cuda) programming guide 2.3. Technical report, NVIDIA Corporation.

Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A. (2006). The landscape of parallel computing research: A view from berkeley. *Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report*, 18(2006-183):19.

Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., et al. (1994). The nas parallel benchmarks. *NASA Ames Research Center, RNR Technical Report RNR-94-007*, pages 94–007.

Schatz, M., Trapnell, C., Delcher, A., and Varshney, A. (2007). High-throughput sequence alignment using graphics processing units. *BMC Bioinformatics*, 8(1).

Tölke, J. (2007). Implementation of a lattice boltzmann kernel using the compute unified device architecture developed by nvidia. *Computing and Visualization in Science*.