Impacto do uso da Memória Virtual Unificada CPU-GPU

Jorge Pires Correia[1], **Wagner M. Nunan Zola**[1]

[1]Universidade Federal do Paraná (UFPR)
Curitiba/PR

{jpcorreia,wagner}@inf.ufpr.br

Resumo. *A facilidade proporcionada pela Memória Virtual Unificada (UVM) através da transparência de movimentações de dados entre o* host *e a GPU pode ser de grande utilidade em processamentos heterogêneos ou com grandes conjuntos de trabalho. Contudo, observamos neste trabalho que a UVM pode apresentar bom desempenho em certas aplicações ou pode piorar em até 27 vezes, demonstrando que a arquitetura da implementação é de fundamental importância para o desempenho de algoritmos que utilizam esta funcionalidade.*

1. Introdução

Atualmente, a GPU é uma das principais plataformas para execução de códigos paralelos. Várias dessas aplicações são heterogêneas ou processam grandes volumes de dados. Assim, a memória da GPU pode ser insuficiente, sendo necessário o uso da memória da CPU como recurso complementar. A Memória Virtual Unificada (UVM, i.e. *Unified Virtual Memory*) provida pelo ambiente CUDA [Harris 2013] fornece gerenciamento automático das movimentações de dados entre memória da GPU e da CPU, via barramento PCI Express, facilitando a implementação de algoritmos. Contudo, o mecanismo de paginação implementado por CUDA pode gerar grandes *overheads* [Landaverde et al. 2014], muitas vezes determinando o uso de *frameworks* específicos para certos tipos de problemas [Ganguly et al. 2021] [Li et al. 2019]. Neste trabalho, analisamos a interferência da paginação com Memória Unificada CUDA em arquiteturas recentes e barramento PCIe 4.0 16x, que suporta maior vazão. Analisamos *microbenchmarks* para os algoritmos de Soma de Prefixos e Ordenação (Radix Sort), implementados na biblioteca Thrust e um kernel Multiplicação de Matrizes implementado em CUDA.

2. Resultados e discussão

Para os experimentos, utilizamos o sistema Linux em processador Intel i7-11700F com 16GiB de RAM, barramento PCIe 4.0 16x e GPU NVIDIA RTX 3060 (memória de 12GiB). Cada experimento foi repetido 10 vezes e reportamos sua média. Os resultados obtidos para o código da Soma de Prefixos podem ser analisados na Figura 1a. Percebe-se que o desempenho do algoritmo utilizando a UVM e inicialização na GPU é equivalente ao desempenho utilizando somente a memória física da GPU, quando esta comporta todos os dados. Contudo, percebe-se que quando o vetor de entrada é inicializado na memória do *host*, o desempenho é afetado em quase 3 vezes. Isso se dá pela necessidade de paginação dos dados da memória do *host* para a GPU. Para vetores de entrada maiores que a memória da GPU, o desempenho é bastante afetado, causando uma perda de desempenho em torno de 27 vezes. Tal comportamento se deve à alta quantidade de *page faults* em relação à quantidade de processamento durante a execução do algoritmo, de forma que a largura de banda do PCI Express influencia diretamente o desempenho geral. Esse kernel tem alta dependência da vazão de RAM, realizando pouco

trabalho para cada acesso à memória. A Figura 1b apresenta os resultados obtidos durante a execução do algoritmo de Ordenação Radix Sort. Em relação ao experimento anterior percebe-se que as *page faults* são menos impactantes. A perda de desempenho para esses testes variam entre 8 vezes e 18 vezes. O kernel de ordenação realiza mais trabalho para cada acesso à memória. Os resultados da execução da Multiplicação de Matrizes são mostrados na Figura 1c. Praticamente inexiste perda de desempenho pela característica do algoritmo: a prevalência da quantidade de operações aritméticas em relação à quantidade de movimentações de memória. Isso permite que a capacidade de processamento da GPU seja melhor explorada paralelamente a cada movimentação de memória, suprimindo a latência dos tratamentos de *page faults*.

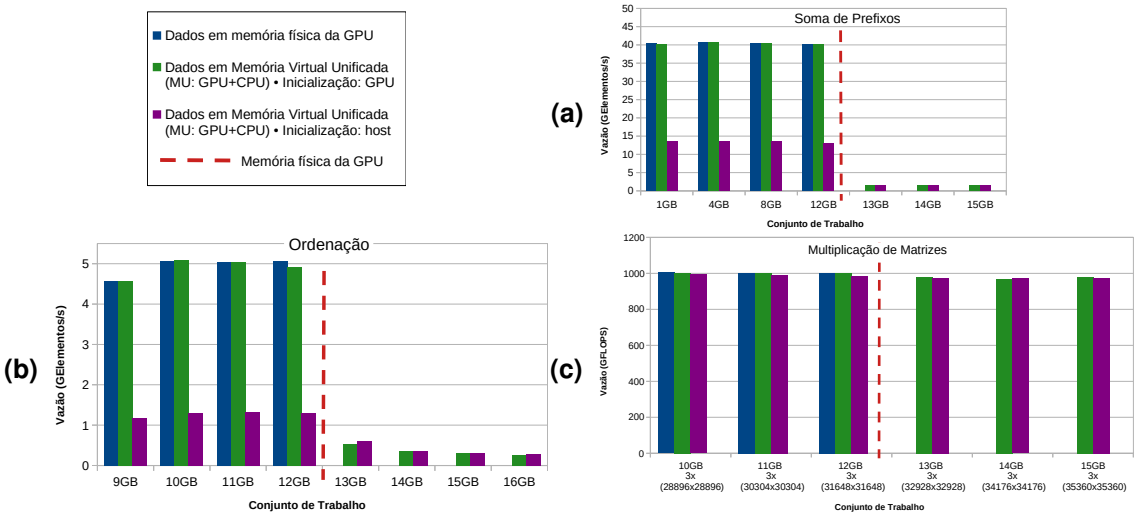

Figura 1. Resultados de desempenho dos Kernels (*microbenchmarks*), com e sem super-alocação de memória e uso de UVM

3. Conclusão

Verificamos que o desempenho dos algoritmos que utilizam Memória Virtual Unificada é bastante variável. As arquiteturas recentes de GPU em conjunto com barramento PCI Express de maior vazão possibilitam melhor desempenho no uso de UVM. A facilidade proporcionada pela transparência das movimentações de memória pode ser explorada em algoritmos que não são *memory bound*.

Referências

Ganguly, D., Melhem, R., and Yang, J. (2021). An adaptive framework for oversubscription management in cpu-gpu unified memory. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1212–1217. IEEE.

Harris, M. (2013). Unified memory in cuda 6. In *NVIDIA Technical Blog*.

Landaverde, R., Zhang, T., Coskun, A. K., and Herbordt, M. (2014). An investigation of unified memory access performance in cuda. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE.

Li, C., Ausavarungnirun, R., Rossbach, C. J., Zhang, Y., Mutlu, O., Guo, Y., and Yang, J. (2019). A framework for memory oversubscription management in graphics processing units. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 49–63.