Proposta de Suporte à Parametrização no NPB com CUDA

Gabriell Araujo, Dalvan Griebler, Luiz G. Fernandes

[1] Escola Politécnica, Grupo de Modelagem de Aplicações Paralelas (GMAP),
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, Brasil

gabriell.araujo@edu.pucrs.br,{dalvan.griebler,luiz.fernandes}@pucrs.br

Resumo. *Este trabalho propõe a introdução de parâmetros configuráveis para GPUs no NPB. A etapa inicial do estudo contemplou a parametrização do número de threads por bloco e seu impacto no desempenho de GPUs.*

1. Contexto

As unidades de processamento gráfico (GPUs) são arquiteturas populares e oferecem paralelismo massivo a baixo custo, porém, sua utilização eficiente é um desafio. *Benchmarks* são uma forma de suportar a pesquisa de GPUs. NAS Parallel Benchmarks (NPB) é um consolidado conjunto de *benchmarks* baseado em computações de dinâmica de fluidos [Bailey et al. 1994] (CFD) que está recebendo atenção no contexto de GPUs com artigos publicados recentemente [Do et al. 2019, d. Araujo et al. 2020].

Uma maneira eficiente de fornecer melhores informações sobre o desempenho em *benchmarks* é oferecer parâmetros configuráveis. Nesse sentido, este trabalho propõe a introdução de parâmetros para GPUs no NPB, iniciando pela parametrização do número de *threads* por bloco e avaliação do seu impacto no desempenho.

2. NPB com CUDA e Número Configurável de Threads

A implementação seguiu boas práticas de programação para GPUs. Ao usuário é permitido especificar o número de *threads* por bloco, através de um arquivo de texto com os valores para cada uma das funções que são paralelizadas para GPU. Os experimentos foram realizados em uma máquina equipada com uma GPU NVIDIA Titan X Pascal (3584 *CUDA Cores*). A classe C foi utilizada como carga de trabalho. Cada teste foi repetido dez vezes. Foram exploradas cinco estratégias para a escolha do número de *threads*. `Warp` utiliza o tamanho do *warp* da GPU. `Max` utiliza o número máximo de *threads* por bloco suportado pela GPU. `Manual` utiliza a "intuição" do programador. `Profiling` realiza *profiling* individual de cada uma das funções da GPU, e coleta o melhor valor para cada função. `Exhausting` testa todas as combinações possíveis de *threads* e coleta a combinação que apresenta o menor tempo de execução.

As estratégias foram aplicadas na implementação deste trabalho e também foram realizados experimentos com os trabalhos da literatura (valores estáticos, pois não suportam parametrização). Cada versão do NPB é nomeada como segue: Versão sequencial [Griebler et al. 2018] como `Serial`. Versão OpenMP [Griebler et al. 2018] como `OpenMP`. Versão OpenCL [Seo et al. 2011] como `OpenCL`. Versão OpenCL [Do et al. 2019] como `OpenCL-2019`. Versão OpenACC [Xu et al. 2015] como `OpenACC`. Versão CUDA [d. Araujo et al. 2020] como `CUDA`. Versão CUDA [Do et al. 2019] como `CUDA-2019`. Versão CUDA deste trabalho com cinco diferentes estratégias como `Warp`, `Max`, `Manual`, `Profiling` e `Exhausting`.

A Figura 1 apresenta o tempo de execução e o desvio padrão através de barras de erro. Foi observado aperfeiçoamento de desempenho relevante em três *benchmarks* ao mudar as estratégias de *threads*, até 17% em EP, 105% em BT, e 554% em CG. EP consome um grande número de registradores e obtém melhor desempenho com quantidades menores de *threads*. BT possui *overhead* de sincronizações que é aliviado ao utilizar quantidades maiores de *threads*. CG gera pequenas tarefas de tamanhos aleatórios, e ao utilizar-se grandes quantidades de *threads*, muitas delas permanecem inativas. Os demais *benchmarks* sofreram impactos menores, pois apresentam limitações moderadas.

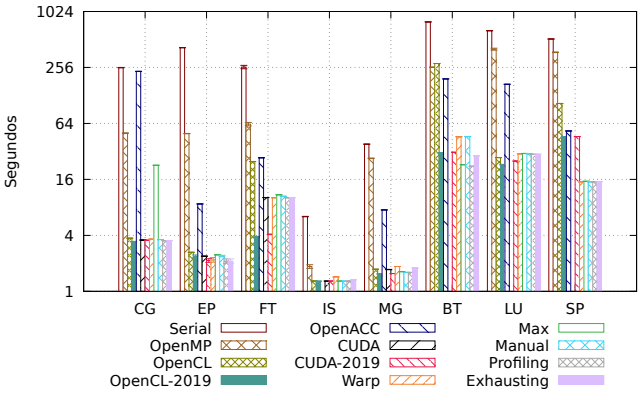

Figura 1. Tempo de execução com o desvio padrão.

3. Conclusões

Demonstrou-se que o número de *threads* pode impactar no desempenho dos *benchmarks* e também fornecer informações relevantes sobre o *software* e o *hardware*. Como trabalhos futuros, novos parâmetros para GPUs podem ser adicionados e avaliados no NPB.

Referências

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. K. (1994). The NAS Parallel Benchmarks RNR-94-007. Technical report, NASA Advanced Supercomputing Division.

d. Araujo, G. A., Griebler, D., Danelutto, M., and Fernandes, L. G. (2020). Efficient NAS Parallel Benchmark Kernels with CUDA. In *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 9–16.

Do, Y., Kim, H., Oh, P., Park, D., and Lee, J. (2019). SNU-NPB 2019: Parallelizing and Optimizing NPB in OpenCL and CUDA for Modern GPUs. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pages 93–105.

Griebler, D., Loff, J., Mencagli, G., Danelutto, M., and Fernandes, L. G. (2018). Efficient NAS Benchmark Kernels with C++ Parallel Programming. In *26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, PDP'18, pages 733–740, Cambridge, UK. IEEE.

Seo, S., Jo, G., and Lee, J. (2011). Performance Characterization of the NAS Parallel Benchmarks in OpenCL. In *2011 IEEE International Symposium on Workload Characterization (IISWC)*, pages 137–148.

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.