Provendo melhorias na geração de código para GPUs na SPar

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. *Neste trabalho são apresentados resultados parciais do estudo que está sendo conduzido para prover melhorias na geração de código paralelo para GPUs na SPar. Foram paralelizadas quatro aplicações de processamento de Stream utilizando a versão original da SPar, e a versão modificada. A versão contendo novas otimizações obteve melhora de até 718% no desempenho de aplicações computacionalmente intensivas para GPUs.*

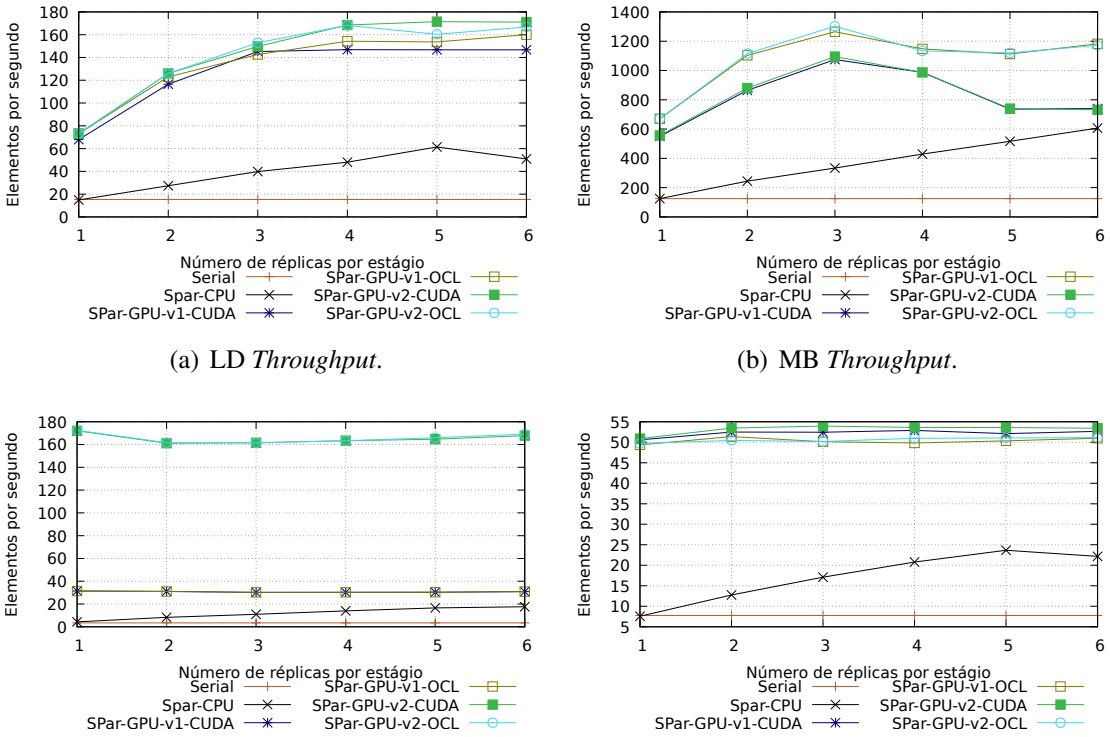
1. Contexto

O processamento de Stream é caracterizado por um fluxo contínuo de dados, os quais são oriundos de fontes como sensores ou internet. Esse paradigma de processamento requer grande poder computacional e está presente em diversos domínios de aplicações tais como veículos autônomos, os quais continuamente detectam objetos e tomam decisões em frações de segundos para evitar acidentes. Nesse sentido, a utilização de Unidades de Processamento Gráfico (GPUs) é essencial para permitir o processamento em tempo real nas aplicações. No entanto, explorar adequadamente o poder computacional das GPUs no processamento de Stream é um desafio para os programadores. SPar é uma Linguagem Específica de Domínio (DSL), cujo objetivo é facilitar a programação de processamento de Stream, gerando código paralelo a partir de anotações no código-fonte utilizando atributos C++ [Griebler et al. 2017]. Atualmente, a SPar possui suporte inicial para geração de código para GPUs [Rockenbach et al. 2022]. Neste documento são apresentados resultados parciais da avaliação ao suporte para GPUs na SPar, bem como apontamentos para possíveis melhorias na DSL.

2. Limitações e melhorias na SPar

Para avaliar a SPar, foram paralelizadas quatro aplicações, Detecção de Pistas (LD), Mandelbrot (MB), Servidor Militar (MS), e Raytracing (RT). Os experimentos foram conduzidos em uma máquina equipada com um processador Intel Xeon E5-2620 (6 núcleos e 12 *threads*), e uma GPU NVIDIA Titan X Pascal (3584 núcleos) com 12 Gigabytes de memória. Figura 1 apresenta os resultados parciais deste trabalho. O eixo x indica a quantidade de réplicas de cada estágio *stateless* no pipeline da aplicação. O eixo y indica a quantidade de elementos processados por segundo. As versões são nomeadas da seguinte forma. Versão Sequencial como `Serial`. Versão Paralela usando a SPar com a CPU como `SPar-CPU`. Versões com a SPar original usando a CPU e a GPU como `SPar-v1-CUDA` e `SPar-v1-OCL`. Versões com a SPar modificada usando a CPU e a GPU como `SPar-v2-CUDA` e `SPar-v2-OCL`.

Foram detectadas limitações de usabilidade e desempenho na SPar. Por exemplo, quanto a usabilidade, a sintaxe não é unificada entre a CPU e a GPU, o que não permite



(a) LD *Throughput*.

(b) MB *Throughput*.

(c) MS *Throughput*.

(d) RT *Throughput*.

Figura 1. Comparação de desempenho.

a portabilidade de código-fonte entre diferentes arquiteturas. Quanto ao desempenho, a SPar não possui otimizações para GPUs. Por exemplo, o código gerado pela SPar opera uma quantidade excessiva de transferências de memória entre a CPU e a GPU, bem como lançamentos e cópias de kernels. Para resolver as limitações da SPar, este trabalho está pesquisando e implementando novas estratégias para a geração de código para GPUs. Os experimentos demonstram que as otimizações implementadas até o momento apresentam melhorias de desempenho significativas em aplicações computacionalmente intensivas para GPUs. A aplicação MS é composta por cinco estágios, e dez rotinas são paralelizadas na GPU. Neste caso, a nova versão da SPar foi até 718% mais rápida que a versão original (Figura 1(c)). A aplicação LD, possui três estágios, e duas rotinas são paralelizadas na GPU. Neste caso, a nova versão da SPar foi até 14.52% mais rápida que a versão original (Figura 1(a)). As aplicações RT e MB também possuem três estágios, porém, apenas uma rotina é paralelizada na GPU. Nesses dois casos, a melhoria de desempenho foi de apenas 1.54% em RT (Figura 1(d)), e 0.0% em MB (Figura 1(c)). Os resultados parciais deste trabalho demonstram o potencial oriundo do estudo que está avaliando e aperfeiçoando a geração de código para GPUs na SPar.

Referências

Griebler, D., Danelutto, M., Torquati, M., and Fernandes, L. G. (2017). SPar: A DSL for High-Level and Productive Stream Parallelism. *Parallel Processing Letters*, 27(01):1740005.

Rockenbach, D. A., Löff, J., Araujo, G., Griebler, D., and Fernandes, L. G. (2022). High-Level Stream and Data Parallelism in C++ for GPUs. In *XXVI Brazilian Symposium on Programming Languages (SBLP)*, SBLP'22, pages 41–49, Uberlândia, Brazil. ACM.