Em direção a um modelo de programação paralela único para CPUs e GPUs em processamento de stream

Gabriell Araujo[1], **Dalvan Griebler**[1], **Luiz G. Fernandes**[1]

[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

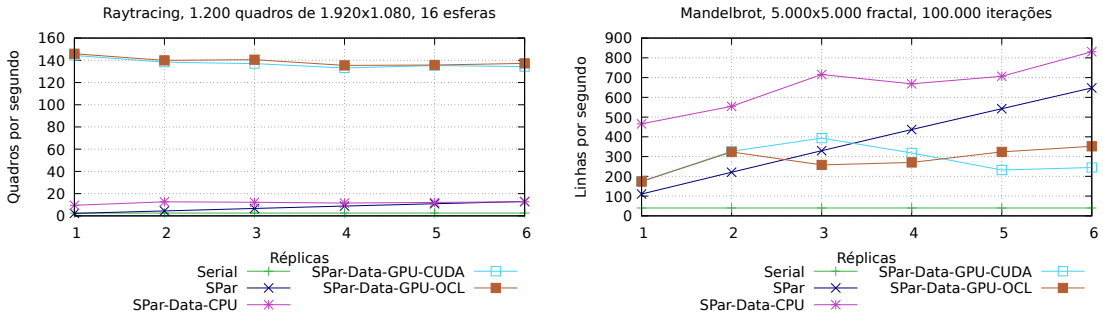
Resumo. *Este trabalho apresenta resultados parciais da pesquisa em andamento, a qual está utilizando a Linguagem Específica de Domínio (DSL) SPar para prototipar um modelo de programação paralela único direcionado a CPUs e GPUs em processamento de stream. Por meio do protótipo inicial, já é possível gerar código paralelo para CPUs e GPUs em processamento de stream.*

1. Contexto

Aplicações de *stream* processam fluxos contínuos de dados, provenientes de diversas fontes, tais como mídias sociais, compras *online* e sensores. Essas aplicações demandam poder computacional considerável para processar os dados em tempo real, necessitando de programação paralela para aproveitar eficientemente o *hardware* do computador. No entanto, a programação paralela é desafiadora para os programadores, exigindo conhecimento do *hardware* e técnicas avançadas de programação, além de refatoração extensiva do código fonte. Ao longo dos anos, diversos trabalhos de pesquisa têm buscado simplificar a programação paralela. Um desses trabalhos é a Linguagem Específica de Domínio (DSL) SPar [Griebler et al. 2017], a qual é embarcada em C++ e permite expressar paralelismo de *stream* por meio de anotações no código fonte utilizando atributos C++. Inicialmente, a SPar gerava código apenas para paralelismo de *stream*. Posteriormente, a SPar foi expandida para possibilitar a combinação de paralelismo de *stream* com paralelismo de dados, permitindo a aceleração das computações de um item enquanto gerencia-se o fluxo de itens da aplicação. Por exemplo, em uma aplicação de vídeo, o processamento de stream gerencia o processamento de diferentes quadros simultaneamente, enquanto o paralelismo de dados acelera as computações de cada quadro. No entanto, as soluções existentes para a combinação de paralelismo de *stream* e dados são específicas ou para CPUs ou para GPUs, exigindo diferentes formas de programação e versões da SPar. A programação de CPUs e GPUs é substancialmente diferente, pois GPUs requerem um paradigma de programação paralela distinto, dada a sua quantidade massiva de núcleos de processamento e seu funcionamento. Além disso, a escolha entre paralelismo de dados em CPU ou GPU depende da natureza da computação ou carga de trabalho [NVIDIA 2024]. Este trabalho visa prototipar um modelo de programação paralela unificado na SPar, capaz de gerar código paralelo para CPUs e GPUs em processamento de *stream*. Os resultados parciais deste trabalho serão apresentados na próxima seção.

2. Resultados Parciais

Esta seção discute os resultados parciais do desenvolvimento do modelo de programação único. O compilador da SPar, responsável pela geração de código paralelo, passa por diversas etapas, incluindo análise semântica, extração de informações relevantes do código fonte, construção de uma árvore sintática abstrata, e substituição de trechos de código anotados com atributos C++ por código paralelo. As principais modificações no compilador concentram-se nas etapas de extração de dados e substituição de código. Durante a extração de dados, o compilador coleta informações necessárias para ambas as arquiteturas alvo, tais como variáveis utilizadas, funções chamadas e definições de estruturas. Já na



(a) Itens por segundo, aplicação RT.

(b) Itens por segundo, aplicação MB.

Figura 1. Desempenho do código gerado pela SPar.

substituição de código, o compilador aplica um conjunto único de regras de transformação para ambas as arquiteturas paralelas. Embora as regras sejam as mesmas, a geração de código para cada arquitetura é distinta; por exemplo, no mapeamento de *threads*, utiliza-se granularidade grossa para CPUs e granularidade fina para GPUs [NVIDIA 2024].

A Figura 1 apresenta resultados parciais com a nova versão da SPar nas aplicações *Raytracing* (RT) e *Mandelbrot* (MB) adaptado para *stream*. O eixo-x indica a quantidade de réplicas dos estágios *stateless* do *pipeline*, enquanto o eixo-y indica a quantidade de itens processados por segundo. A respeito das versões. `Serial` representa a execução sequencial da aplicação; `SPar` explora apenas paralelismo de *stream*; `SPar-Data-CPU` combina paralelismo de dados e *stream* utilizando apenas a CPU; `SPar-Data-GPU-CUDA` e `SPar-Data-GPU-OCL` combinam paralelismo de dados e *stream* utilizando CPU e GPU (CUDA e OpenCL). Os experimentos foram conduzidos em uma máquina equipada com um processador AMD Ryzen 5 5600x (6 cores / 12 threads), 32 Gigabytes de memória RAM e uma GPU NVIDIA RTX 3090 (10.496 *cores*). Na aplicação RT, observa-se um aumento significativo de desempenho ao utilizar-se GPUs, pois cada item da aplicação trata-se uma matriz de pixels, ideal para processamento em GPUs. Em contrapartida, a versão que combina paralelismo de dados e *stream* apenas na CPU não apresenta desempenho superior à versão que utiliza apenas processamento de *stream* , visto que o paralelismo de *stream* já explora todo o potencial da CPU. Já na aplicação MB, onde os itens são linhas de uma imagem *fractal*, não há carga de trabalho intensiva o suficiente para justificar o uso de GPUs. Assim, o desempenho da versão para GPU é inferior às versões que exploram apenas o paralelismo na CPU. Por outro lado, os melhores resultados são obtidos ao combinar-se o paralelismo de dados e *stream* na CPU. Com o paralelismo de dados aplicado a uma linha da matriz, cada *thread* da CPU carrega apenas algumas posições da linha em vez da linha inteira, melhorando a taxa de *cache hits*. A prototipação de um modelo único de programação para CPUs e GPUs na SPar está em estágio inicial, mas já apresenta resultados promissores para melhorar o desempenho em aplicações de *stream*. Os resultados indicam que a geração de código para diferentes arquiteturas alvo é mais vantajosa, dependendo das características da aplicação.

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.

NVIDIA (2024). CUDA C Programming Guide. Technical report, NVIDIA Corporation.