Proposta de uma suíte de benchmarks para avaliar o paralelismo de stream em GPUs

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

[1] Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, Brasil

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

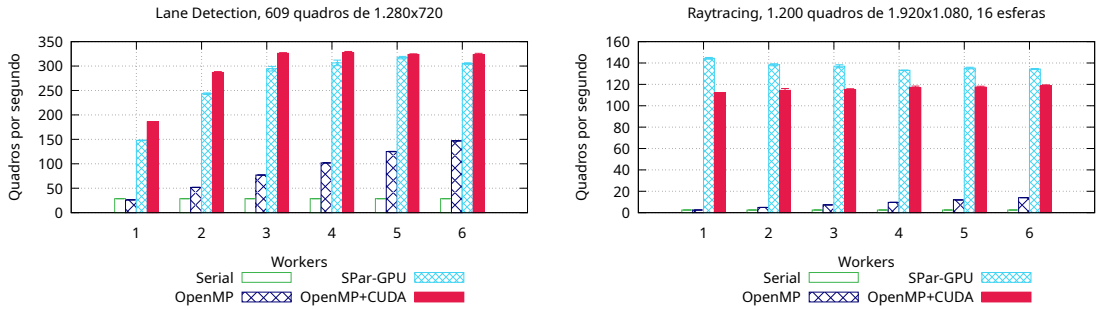
Resumo. *GPUs são essenciais para permitir o processamento de dados em tempo real em aplicações de stream. Porém, a literatura apresenta diferentes limitações concernentes a este tópico. Este trabalho propõe o desenvolvimento de uma suíte de benchmarks de aplicações de stream aceleradas por GPUs, bem como uma investigação de técnicas de programação e desafios nesse domínio.*

1. Contexto

Nos dias atuais, os computadores estão cada vez mais presentes na vida das pessoas e grandes volumes de dados são gerados a todo momento. Nesse contexto, aplicações de stream são caracterizadas pelo processamento contínuo desses dados, os quais podem ser oriundos de diferentes fontes, tais como redes sociais, compras online e mercado financeiro[Griebler et al. 2017]. No entanto, para permitir o processamento eficiente de dados gerados em tempo real, é necessário o uso de programação paralela para extrair o máximo de desempenho dos computadores modernos. Ao longo dos anos, as Unidades de Processamento Gráfico (GPUs) tem se destacado por prover grandes melhorias de desempenho em aplicações de diferentes domínios, incluindo aplicações de stream. Tratando-se do domínio de stream, é possível encontrar diferentes lacunas na literatura. Uma delas trata-se da quantidade limitada de trabalhos explorando técnicas e otimizações para GPUs nas aplicações desse domínio, assim como a inexistência de suítes de benchmarks desenvolvidos para avaliar processamento de stream acelerado por GPUs. Em contraste, benchmarks são ferramentas fundamentais para a investigação e validação de técnicas e estratégias no desenvolvimento de hardware e software[Griebler et al. 2017]. Nesse sentido, este trabalho visa desenvolver uma suíte de benchmarks de stream acelerados por GPUs. Além de prover benchmarks que permitam avaliar o desempenho de GPUs no domínio de stream, o trabalho visa investigar e avaliar técnicas e estratégias de paralelismo para acelerar aplicações de stream utilizando GPUs. Neste documento, são apresentados resultados preliminares para dois benchmarks de stream.

2. Resultados Preliminares

Nesta seção são apresentados os resultados iniciais deste projeto. Duas aplicações foram implementadas e organizadas na forma de benchmarks: 1) Lane Detection (LD) captura imagens em tempo real através de uma câmera instalada em um veículo autônomo e utiliza filtros para detectar as linhas da pista; 2) Raytracing (RT) calcula a iluminação em imagens 3D continuamente recebidas pelo programa. LD e RT são benchmarks compostos por três estágios: Estágio um lê uma imagem da entrada; Estágio dois opera transformações na imagem recebida; Estágio três escreve a imagem na saída. Foram desenvolvidas quatro versões das aplicações: Uma versão sequencial (`Serial`); Uma versão paralela para CPU utilizando OpenMP (`OpenMP`); Duas versões utilizando paralelismo na CPU e GPU, uma utilizando a Linguagem Específica de Domínio (DSL) para stream chamada SPar (`SPar-GPU`)[Griebler et al. 2017] e outra utilizando OpenMP



(a) Quadros por segundo, aplicação LD.

(b) Quadros por segundo, aplicação RT.

Figura 1. Desempenho obtido com diferentes versões paralelas.

combinado com CUDA (`OpenMP+CUDA`). 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*). Cada teste foi executado 10 vezes para que diferentes métricas fossem coletadas, tais como tempo de execução e desvio padrão. A Figura 1 apresenta os resultados preliminares deste projeto. O eixo-x significa a quantidade de réplicas dos estágios da aplicação, e o eixo-y significa a quantidade de quadros processados por segundo. As aplicações foram paralelizadas, primeiramente utilizando o padrão paralelo pipeline, o qual é executado pela CPU e permite o processamento simultâneo de diferentes imagens. Enquanto a versão nomeada como `OpenMP` explora apenas o paralelismo de pipeline, as versões nomeadas como `SPar-GPU` e `OpenMP+CUDA` combinam o paralelismo de pipeline na CPU com o paralelismo de dados na GPU para calcular as transformações das imagens, aumentando o desempenho das aplicações. A quantidade de réplicas impactou consideravelmente o benchmark LD, mas não impactou o benchmark RT. Enquanto LD possui diversas rotinas que são executadas pela CPU no estágio de transformação das imagens, todas as operações de transformação são executadas pela GPU no benchmark RT. Concernente ao processo de implementação, OpenMP e CUDA apresentaram desafios adicionais quando comparados com a SPar. Embora OpenMP ofereça abstrações de alto-nível, elas não contemplam paralelismo de stream. Dessa forma, o programador precisa implementar manualmente um pipeline utilizando paralelismo de tarefas (por exemplo, usando anotações como `task`), bem como prover uma estrutura de comunicação entre os estágios (por exemplo, usando filas). Ao combinar OpenMP com CUDA, além da estrutura do pipeline, o programador precisa programar manualmente a GPU, administrando o lançamento de kernels, transferências de memória, e mapeamento das threads da GPU. Em contraste, a DSL SPar permite geração de código paralelo por meio de três anotações. `ToStream` cria um pipeline, `Stage` cria um estágio do pipeline, e `Pure` paraleliza um laço usando a GPU. A investigação inicial deste projeto demonstrou o potencial das GPUs no processamento de stream, bem como proveu uma análise preliminar sobre os desafios do processamento de stream ao abordar ferramentas de programação com diferentes níveis de abstração[1].

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.

[1]Esta pesquisa foi financiada com o apoio de FAPERGS 09/2023 PqG (Nº 24/2551-0001400-4) e CNPq Research Program (Nº306511/2021-5)