Proposta de Paralelismo de Stream Multi-GPU em Multi-Cores

Gabriel Rustick Fim[1], **Dalvan Griebler**[1]

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

gabriel.fim@edu.pucrs.br, dalvan.griebler@pucrs.br

Resumo. *Considerando a necessidade de tempos de processamento mais rápidos, a utilização de ambientes multi-aceleradores vem se tornando cada vez mais proeminente na literatura, infelizmente programar para estes tipos de ambientes apresenta uma série de desafios que fazem com que o desenvolvimento de códigos direcionados a multi-GPUs exija um maior esforço de programação. Propomos investigar como utilizar anotações C++ para simplificar a geração de código multi-GPU sem comprometer o desempenho.*

1. Introdução

Embora as GPUs estejam presentes em praticamente todas as máquinas nos últimos anos, escrever código para essas arquiteturas é uma tarefa desafiadora, pois envolve conhecer diferentes técnicas de programação dependendo de qual linguagem GPU é suportada pelo acelerador, além de conhecimento sobre o hardware, conceitos de programação multi-core, e também profundo conhecimento sobre o problema que se pretende resolver, muitas vezes necessitando expor o paralelismo massivo adequado para processamento nas GPUs.

Existem dois tipos de conectividade em sistemas multi-GPU: múltiplas GPUs conectadas através do barramento PCIe em um único nó e vários nós contendo GPUs conectadas através de um switch de rede em um cluster. Para projetar um programa que use várias GPUs com eficiência, é necessário particionar a carga de trabalho entre dispositivos. Isso pode resultar em dois padrões comuns de comunicação entre GPUs: nenhuma troca de dados é necessária entre partições ou troca parcial de dados entre partições, exigindo armazenamento redundante de dados.

Além das solução mais conhecidas como OpenACC, CUDA, OpenCL e OpenMP, existe a SPar[Griebler et al. 2017]. Ela é uma linguagem específica de domínio incorporada em C++ que oferece abstrações de alto nível para paralelismo de fluxo por meio de anotações de código. Inicialmente o SPar focou na geração de código para sistemas *multi-core* usando FastFlow. Trabalhos posteriores expandiram o escopo para outras arquiteturas usando GSParLib [Rockenbach et al. 2022] para GPUs e DSParLib [Löff et al. 2022] para arquiteturas distribuídas.

Atualmente, o SPar carece de suporte multi-GPU, principalmente no que diz respeito a otimizações e técnicas que permitem ao usuário extrair totalmente o poder computacional de seu ambiente. Considerando esses fatores, nosso objetivo é simplificar a geração de código multi-GPU sem comprometer o desempenho, através de anotações C++, como as fornecidas pelo SPar para paralelismo de *stream*.

2. Proposta de Pesquisa

Esta proposta pode ser definida em várias etapas. A primeira etapa envolve pesquisa exploratória para identificar técnicas e otimizações para programação multi-GPU. Estas englobam particionamento de dados, comunicação entre-GPUs, *scheduling* de tarefas entre GPUs e como executar as operações de maneira assíncrona nas GPUs. A segunda etapa envolve estudar as extensões de linguagem e bibliotecas que permitem a aceleração de código C++ com multi-GPUs atualmente presentes na literatura e otimizações estas implementam. Figura 1 mostra o que espera-se alcançar através das anotações providas pela SPar, uma maneira de diminuir o desafio de programação imposto ao usuário pela utilização de multi-GPU.

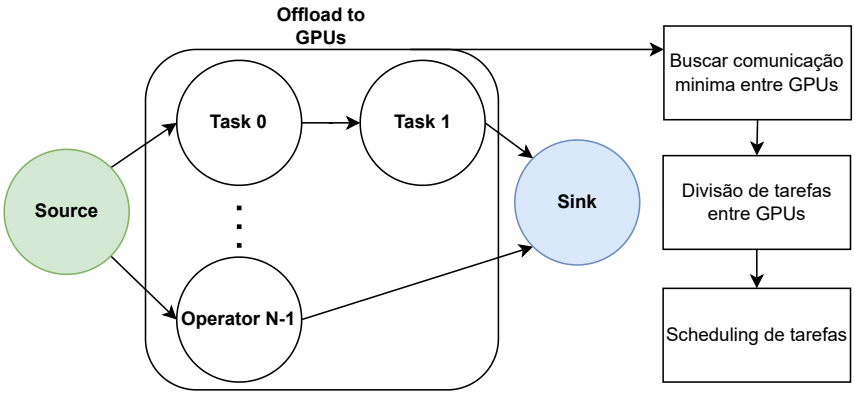

Figura 1. Arquitetura da Proposta.

A terceira etapa consiste em implementar as possíveis alterações necessárias para permitir que a SPar gere código multi-GPU por meio de suas anotações. O quarto passo é trazer quaisquer técnicas e otimizações encontradas nas etapas anteriores para a geração de código multi-GPU via SPar. Como último passo faremost uma análise de performance contra outras extensões de linguagem ou bibliotecas adicionais.

Esta proposta de pesquisa requer uma compreensão adicional da programação multi-GPU e geração de código utilizando as anotações de código SPar. O progresso deste trabalho pode proporcionar *insights* sobre a geração de código multi-GPU mais rápida e fácil com base em anotações C++. Os resultados devem fornecer geração de código multi-GPU em ambientes de nó único, com baixo impacto nos aspectos de programabilidade, bem como performance comparável as atuais ferramentas de programação multi-GPU presentes na literatura.

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.

Löff, J., Hoffmann, R. B., Pieper, R., Griebler, D., and Fernandes, L. G. (2022). DSParLib: A C++ Template Library for Distributed Stream Parallelism. *International Journal of Parallel Programming*, 50(5):454–485.

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.