Proposta de um Framework para Avaliar Interfaces de Programação Paralela em Aplicações de *Stream*

Adriano Marques Garcia[1], **Dalvan Griebler**[1], **Claudio Schepke**[2], **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.

adriano.garcia@edu.pucrs.br

[2]Universidade Federal do Pampa (UNIPAMPA), Alegrete, RS, Brasil

Resumo. *Este trabalho propõe um framework que auxilia no desenvolvimento de benchmarks para avaliar Interfaces de Programação Paralela no domínio de paralelismo de stream em C++.*

1. Proposta do Framework

A popularização de *IoT* e *Edge Computing* causaram um aumento na demanda por processamento em tempo real. Esse tipo de processamento é feito por aplicações de processamento de *stream* (fluxo), que processam esses dados assim que eles são recebidos. Essas aplicações dividem o processamento em pequenos estágios, como um *pipeline*, e para melhorar o desempenho é possível paralelizar e replicar esses estágios. Entretanto, explorar o paralelismo entre diferentes estágios e também entre réplicas é uma tarefa complexa, geralmente reservada para especialistas [Andrade et al. 2014].

Existem limitações em termos de *benchmarks* para o domínio de Processamento de *Stream*. As *suites* de *benchmarks* atuais focam apenas em aplicações para processamento de eventos em tempo real, como sistemas de monitoramento, transações financeiras, mercado de ações, etc. [Garcia et al. 2021]. Essas aplicações são implementadas utilizando plataformas distribuídas com foco maior em reduzir a latência [Bordin et al. 2020]. Por outro lado, não existem *suites* de *benchmarks* representativas para processamento de *stream* onde o objetivo é alcançar uma maior vazão de processamento de dados, como aplicações de compressão de dados, processamento de vídeo/áudio/imagem, sequenciamento de genoma, etc. Para preencher essa lacuna, este trabalho propõe um *framework* para avaliação de Interfaces de Programação Paralela (IPPs) para processamento de *stream*. Através do *framework* é possível criar *benchmarks* customizáveis para explorar paralelismo com diferentes IPPs. Os *benchmarks* automaticamente incluem diversas funcionalidades, como diferentes configurações de *workload* e métricas de desempenho. Portanto, o objetivo do *framework* é dar a oportunidade para programadores utilizarem aplicações realísticas para avaliar IPPs, testar otimizações, técnicas de paralelismo, etc.

As aplicações do *framework* são implementadas de forma sequencial com uma Interface de Programação de Aplicações (API) que evidencia ao programador somente as características essenciais de uma aplicação de *stream*, como a estrutura dos operadores e os dados que são comunicados. Dessa forma, essa API adiciona uma camada de abstração que permite que o programador foque exclusivamente nos aspectos de paralelismo, deixando os detalhes de implementação da aplicação em si transparentes. Para isso, a API identifica e separa os possíveis operadores do código original e encapsula os dados que transitam entre eles. Dessa forma, as aplicações do *framework* são implementadas instanciando essa API e apenas fazendo chamadas à cada estágio, por exemplo: `operador_1(), operador_2(), operador_`n`()`. Cada estágio recebe ou envia um objeto genérico chamado “item”, que dentro da API pode ser um *frame*, imagem,

bloco de bytes, etc. Junto ao *framework* também existe uma interface de linha de comando para compilação, execução, coleta de métricas de execução (vazão, latência, uso de CPU e memória), fonte de dados (como disco, memória e rede), entre outros [Garcia et al. 2021].

No momento, o *framework* disponibiliza três aplicações de processamento de *stream*: Bzip2, Detecção de Pistas e Reconhecimento de Rostos. Para testar as funcionalidades, foram paralelizadas e avaliadas essas aplicações utilizando três IPPs (FastFlow [Aldinucci et al. 2017], TBB [Reinders 2007] e SPar [Griebler et al. 2017]) e executadas em uma máquina com 12 núcleos (24 *threads*). Nas versões paralelas, todos os estágios internos de cada aplicação foram combinados em um único estágio do pipeline. Portanto, todos os casos testados ficaram com uma estrutura de pipeline do tipo: leitura $\rightarrow$ estágios combinados $\rightarrow$ escrita. Os estágios combinados não possuem dependência de dados e podem ser replicados livremente. Foram realizados experimentos para avaliar o desempenho (itens processados por segundo) utilizando entre 1 e 24 réplicas (Figura 1. 0 réplica no gráfico representa a versão sequencial. As aplicações foram facilmente paralelizadas, apresentaram bom desempenho e não exigiram nenhuma modificação extra além da adição das próprias estruturas de paralelismo de cada IPP. Um estudo comparativo ainda é necessário para mostrar se há algum impacto negativo no desempenho quando utilizado o *framework*. Futuramente, espera-se incrementar o *framework* com mais aplicações, IPPs e expandiremos algumas funcionalidades.

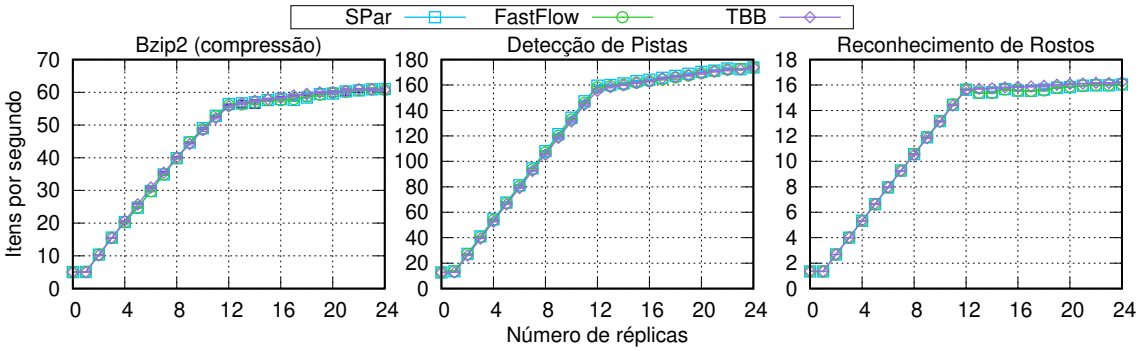

Figura 1. Itens processados por segundo com os três *benchmarks* e as três IPPs.

Referências

Aldinucci, M., Danelutto, M., Kilpatrick, P., and Torquati, M. (2017). *Fastflow: High-Level and Efficient Streaming on Multicore*, chapter 13, pages 261–280. John Wiley & Sons, Ltd.

Andrade, H. C., Gedik, B., and Turaga, D. S. (2014). *Fundamentals of stream processing: application design, systems, and analytics*. Cambridge University Press.

Bordin, M. V., Griebler, D., Mencagli, G., Geyer, C. F. R., and Fernandes, L. G. (2020). DSPBench: a Suite of Benchmark Applications for Distributed Data Stream Processing Systems. *IEEE Access*, 8(na):222900–222917.

Garcia, A. M., Griebler, D., Schepke, C., and Fernandes, L. G. (2021). Introducing a Stream Processing Framework for Assessing Parallel Programming Interfaces. In *29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, PDP'21, Valladolid, Spain. IEEE.

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.

Reinders, J. (2007). *Intel Threading Building Blocks*. O'Reilly, Sebastopol, CA, USA.