Proposta de Framework para Processamento de Stream Distribuído em C++ utilizando o MPI

Júnior Löff[1], **Dalvan Griebler**[1], **Luiz G. Fernandes**[1]

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

junior.loff@edu.pucrs.br,{dalvan.griebler,luiz.gustavo}@pucrs.br

Resumo. *Este trabalho apresenta uma proposta de framework para processamento de stream distribuído em C++ com MPI. A etapa inicial do estudo aborda a problemática de pesquisa e a concepção da arquitetura do framework.*

1. Introdução

Recentemente, aplicações modernas de diferentes domínios vêm demandando cada vez mais poder de processamento com requisitos mínimos de latência e de consumo energético [Zeuch et al. 2019]. Visto que processadores *multi-core* enfrentam barreiras físicas para atender tal demanda, foi uma consequência natural transicionar para ambientes distribuídos. O estado-da-arte em computação distribuída apresenta diferentes soluções, cada uma com suas vantagens e desvantagens. Por exemplo, o Apache Hadoop foi um dos pioneiros, introduzindo o padrão MapReduce para modelar aplicações de Big Data. Entretanto, apresenta desempenho limitado via acessos constantes ao disco. O Apache Spark evoluiu essa questão através de uma nova estratégia baseada em RDDs (*Resilient Distributed Datasets*), que funcionam como uma espécie de *cache* que mantém os dados em memória. Porém, o Apache Spark não suporta processamento em tempo-real, para isso existem soluções como Apache Flink e Apache Storm. Por sua vez, essas duas ferramentas são implementadas em Java, apresentando desempenho inferior se comparado à linguagens como C/C++ [Zeuch et al. 2019].

No quesito desempenho computacional, o atual estado-da-arte em sistemas distribuídos é o MPI [Hori et al. 2021, Zeuch et al. 2019]. Porém, a principal desvantagem do MPI é em relação a sua programabilidade. A interface de abstrações requer que o programador implemente manualmente os mecanismos de comunicação, serialização, tolerância à falha e outros. Devido a essa característica, somente programadores especialistas em computação paralela e distribuída conseguem explorar eficientemente o desempenho prometido pelo MPI. Ademais, até mesmo desenvolvedores experientes não usufruem todo os recursos oferecidos pela ferramenta. Por exemplo, a partir da especificação MPI 2.0 é possível fazer o gerenciamento dinâmico de processos no MPI, criando ou destruindo processos em tempo de execução. Entretanto, de acordo com a *survey* [Hori et al. 2021], essa funcionalidade é raramente utilizada pelos desenvolvedores.

Nesse trabalho estamos interessados no paradigma de processamento de *stream*. Aplicações de *stream* executam durante longos períodos de tempo, muitas vezes infinitos, e com possíveis variações na carga de trabalho. O sistema deve adaptar-se aos picos de dados e possuir resiliência frente às falhas. Na literatura, nota-se que existe uma lacuna entre MPI e o domínio de processamento de *stream* [Hori et al. 2021]. O objetivo deste trabalho é conceber um novo Framework para processamento de *stream* distribuído em C++ utilizando o MPI. Entretanto, não pretende-se criar uma nova solução para competir com

sistemas de processamento de *stream* como o Apache Flink ou Apache Spark. Ao invés disso, os esforços se concentrarão em projetar um Framework que ofereça os mecanismos mínimos para implementação de protocolos para recuperação de falhas e estratégias auto-adaptativas. No futuro, outros pesquisadores poderão utilizar esse Framework para implementar e testar suas próprias estratégias e algoritmos.

2. Proposta de Pesquisa

O estudo está no seu estágio inicial e será elaborado no formato de dissertação de mestrado. A Figura 1 ilustra a proposta de arquitetura do Framework e está organizada de acordo com três níveis e cinco camadas de abstrações. No nível mais alto, espera-se oferecer os mecanismos para que o programador consiga modelar e paralelizar aplicações de processamento de *stream* sem envolvê-lo em detalhes de arquitetura. Por exemplo, o Framework oferece padrões paralelos que já implementam mecanismos auto-adaptativos e de tolerância à falhas. Além disso, linguagens-específicas de domínio (DSL) são capazes de alavancar essas funcionalidades a fim de aumentar ainda mais o nível de abstração.

No nível intermediário, encontra-se a camada para desenvolvedores de sistemas. Essa camada oferece uma interface de baixo-nível em formato de módulos que podem ser re-implementados com novas estratégias auto-adaptativas, de tolerância à falhas, de serialização, e outros. Essa camada representa uma das principais contribuições desse trabalho, já que pretende-se oferecer os mecanismos necessários para implementação de estratégias auto-adaptativas e de tolerância a falhas. No nível mais baixo de abstração, encontra-se a *runtime* do Framework. Nesse nível, a camada mais inferior representa a comunicação e será integrada com o MPI. Na camada logo acima encontra-se o motor de processamento do Framework, que será responsável por suportar a integração entre os módulos do Framework e orquestrar a computação.

Ainda é necessário evoluir o estudo e as descobertas podem implicar em alterações no Framework. Espera-se que os resultados da pesquisa contribuam com o desenvolvimento de sistemas distribuídos e toda cadeia de soluções que alavancam esse eco-sistema.

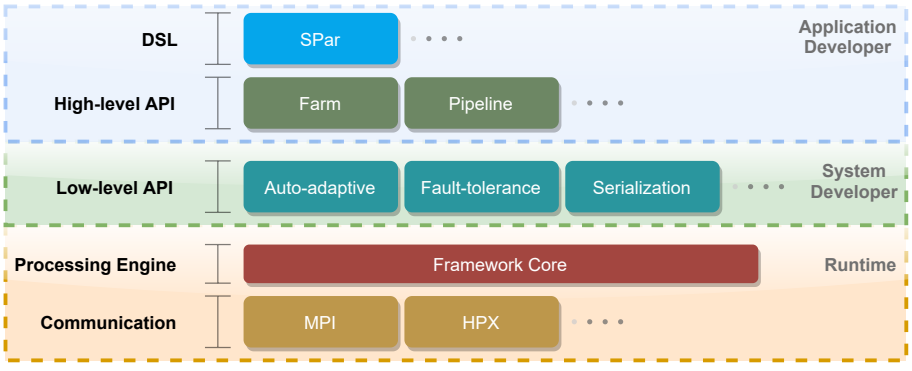

Figura 1. Proposta de arquitetura do Framework.

Referências

Hori, A., Jeannot, E., Bosilca, G., Ogura, T., Gerofi, B., Yin, J., and Ishikawa, Y. (2021). An international survey on mpi users. *Parallel Computing*, 108:102853.

Zeuch, S., Monte, B. D., Karimov, J., Lutz, C., Renz, M., Traub, J., Breß, S., Rabl, T., and Markl, V. (2019). Analyzing efficient stream processing on modern hardware. *Proceedings of the VLDB Endowment*, 12(5):516–530.