Tolerância a Falhas para Paralelismo de Stream de Alto Nível

Lucas M. Alf[1], **Dalvan Griebler**[1]

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

lucas.alf@edu.pucrs.br, dalvan.griebler@pucrs.br

Resumo. *Dada a necessidade dos sistemas de processamento de stream serem executados por longos períodos de tempo, possivelmente indefinidamente, realizar o reprocessamento de todos os dados em caso de falha pode ser altamente custoso ou até mesmo inviável. Nesta pesquisa, propomos investigar como fornecer mecanismos de tolerância a falhas e garantias de consistência para paralelismo de stream distribuído em alto nível.*

1. Introdução

Stream processing pode ser definido como um paradigma de computação que envolve a coleta, processamento e análise de um fluxo contínuo e heterogêneo de dados em grande volume, com o objetivo de extrair informações valiosas ou percepções em tempo real. Este paradigma é resultante dos últimos 50 anos de evolução constante nas tecnologias utilizadas para armazenar, organizar e analisar a crescente quantidade de dados gerados por organizações [Andrade et al. 2014].

Devido à necessidade de sistemas de processamento de *stream*, como o Apache Flink, Apache Spark, Apache Storm, Google DataFlow, dentre outros, serem executados por longos períodos de tempo, possivelmente indefinidamente, realizar o reprocessamento de todos os dados em caso de falha pode ser altamente custoso ou até mesmo impraticável. Portanto, é essencial que sistemas de processamento de *stream* não apenas se recuperem após uma falha, mas também assegurem que os resultados gerados estejam corretos.

A SPar consiste em uma linguagem de domínio específico para C++ que possui como objetivo simplificar a geração de código para aplicações de processamento de *stream* paralelo. Esta linguagem foi criada reconhecendo os desafios associados à escrita de aplicações de *stream* paralelo de alto desempenho, como o equilíbrio entre a facilidade de uso e o desempenho, e a falta de bibliotecas de alto nível que facilitem a criação destas aplicações [Griebler et al. 2017]. Inicialmente, a SPar possuía como foco apenas a geração de código para sistemas *multi-core*, porém trabalhos futuros expandiram o escopo para outras arquiteturas, como a DSParLib [Löff et al. 2022], que possibilita a geração de código para arquiteturas distribuídas.

A DSParLib (*Distributed Stream Parallelism Library*) consiste em uma biblioteca de paralelismo de *stream* para arquiteturas distribuídas, que utiliza o OpenMPI, uma implementação de código aberto da biblioteca MPI (*Message Passing Interface*) para a geração de código distribuído. Atualmente, a DSParLib carece de mecanismos de tolerância a falhas e não possui garantias de entrega de mensagens.

Considerando esses fatores, o objetivo deste trabalho consiste em explorar como outros sistemas de processamento de *stream* oferecem mecanismos de tolerância a falhas e garantias de consistência, e como esses mecanismos podem ser implementados no ambiente SPar distribuído, preferencialmente de maneira que o usuário não precise alterar o código de sua aplicação para se beneficiar do mecanismo de tolerância a falhas. Este trabalho fará parte da dissertação de mestrado do autor em 2024.

2. Proposta de pesquisa

Este trabalho começa com uma pesquisa exploratória em livros, artigos e revistas, com o objetivo de identificar o atual estado da arte em relação a sistemas de processamento de *stream*, e entender como esses sistemas oferecem tolerância a falhas e garantias de consistência. Esta etapa é seguida por um estudo de como o ambiente SPar distribuído atualmente funciona. A segunda etapa envolve a análise e identificação de quais mecanismos de tolerância a falhas são adequados para o ambiente SPar distribuído, seguido da implementação do mecanismo escolhido. Como ilustrado na Figura 1, dentre as atuais hipóteses está a utilização do mecanismo de *Write-Ahead Log* em conjunto ao mecanismo de *checkpoint* em cada um dos operadores, sobre um sistema de arquivos distribuído.

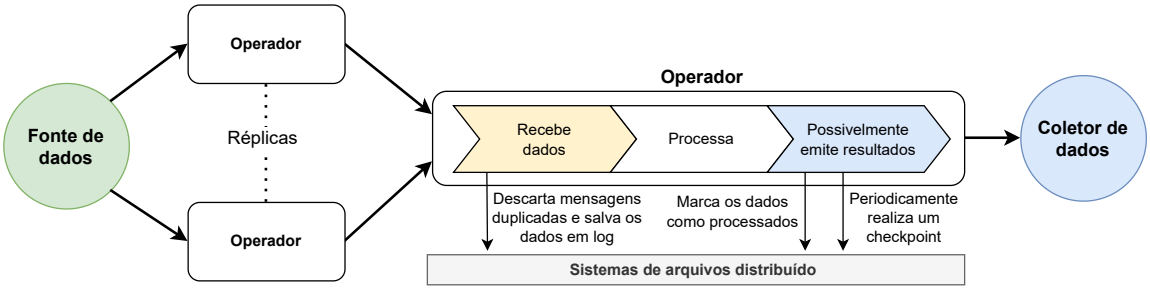

Figura 1. Exemplo de mecanismo de tolerância a falhas

A terceira etapa envolve a preparação e execução de um conjunto de aplicações de processamento de *stream* que representem problemas relevantes no mundo real, seguido de uma análise quantitativa das métricas de desempenho (latência, *throughput* e *speed-up*) comparando a implementação original da SPar com a implementação que possui recursos de tolerância a falhas. Esta análise quantitativa tem como objetivo identificar como o mecanismo de tolerância a falhas impacta o desempenho das aplicações. A etapa final envolve uma análise qualitativa dos aspectos de programabilidade. Essa análise qualitativa tem como objetivo identificar se o mecanismo implementado influencia fatores como facilidade de uso e número de linhas de código.

Esta proposta de pesquisa exige um entendimento aprofundado sobre o tema de tolerância a falhas e garantias de consistência em sistemas de processamento de *stream*. Avançar nesta pesquisa pode proporcionar novos *insights* sobre tolerância a falhas em sistemas de processamento de *stream* baseados em C++. Os resultados desta pesquisa devem fornecer um mecanismo de tolerância a falhas para o ambiente SPar distribuído, com baixo impacto nos aspectos de programabilidade.

Referências

Andrade, H. C. M., Gedik, B., and Turaga, D. S. (2014). *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge University Press.

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.