Proposta de Tolerância a Falhas Baseada em Log para Paralelismo de Stream

Eduardo M. Martins[1], **Dalvan Griebler**[1]

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

e.martins01@edu.pucrs.br, dalvan.griebler@pucrs.br

Resumo. *Sistemas de processamento de stream precisam ser capazes de lidar com grandes volumes de dados e por tempo indeterminado. A necessidade de analises em tempo real tornam o paralelismo e mecanismos de tolerância a falhas essenciais nesse contexto. Nesta pesquisa propomos implementar suporte à um protocolo de tolerância a falhas baseado em log em uma biblioteca para processamento de stream de alto nível para C++.*

1. Introdução

O processamento de *stream* é um paradigma voltado para a coleta, filtragem, processamento e análise de fluxos de dados contínuos, heterogêneos e de alto volume. Esse modelo atende à crescente demanda por sistemas digitais automatizados ou orientados à informação, impulsionada pela evolução dos métodos de armazenamento e organização de dados [Andrade et al. 2014]. Como a geração e a análise desses registros ocorrem em tempo real, técnicas de paralelismo tornam-se fundamentais para atender os requisitos de desempenho. Sistemas de processamento contínuo e de duração indefinida também requerem mecanismos para garantir a execução ininterrupta e a corretude dos resultados, mesmo em caso de falhas. Ferramentas como Apache Flink, Apache Spark, Apache Stream e Google Dataflow oferecem suporte à tolerância a falhas por meio de estratégias como *checkpointing* e *logging*, além de proverem outras garantias como a semântica de entrega *exactly-once*. Os protocolos baseados em *log* podem ser classificado em abordagem: pessimista, onde os processos precisam bloquear sua execução até que o evento seja armazenado; abordagem otimista, onde os processos não são bloqueados e os eventos persistem em armazenamento de forma assíncrona; e abordagem causal, que visa um equilíbrio entre ambos métodos.

SPar é uma linguagem de domínio específico em C++ para expressar o paralelismo de *stream*, que possibilita a inserção de anotações de alto nível no código sequencial para gerar automaticamente código paralelo. O objetivo principal dessa ferramenta é equilibrar desempenho com programabilidade e produtividade [Griebler et al. 2017]. Inicialmente desenvolvida para ambientes de memória compartilhada, a ferramenta foi estendida para oferecer suporte a ambientes distribuídos por meio da DSParLib [Löff et al. 2022], que é uma biblioteca baseada em MPI e que não possui suporte a tolerância a falhas. A ResiFlow é a mais recente extensão em desenvolvimento para o ecossistema SPar. Trata-se de uma biblioteca em C++ voltada para o processamento distribuído de *stream*, também baseada em MPI, mas com suporte à tolerância a falhas por meio de *checkpoints*. Atualmente a ResiFlow suporta fluxos de execução no padrão de *pipelines* lineares e garantias de entrega. Dentro desse contexto, há oportunidade de agregar mais expressividade e

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

recursos à essas ferramentas, aproveitando os benefícios de programabilidade e produtividade que elas já oferecem. Planeja-se atribuir ao ambiente da ResiFlow o suporte à um protocolo de tolerância a falhas de nó alternativo baseado em *log*.

2. Proposta de pesquisa

Este projeto inicia-se com uma pesquisa bibliográfica aprofundada, visando compreender o estado da arte dos mecanismos de recuperação de falhas baseados em *log* no contexto de *streaming*. Atualmente, as ferramentas mais relevantes que adotam essa abordagem são o Google DataFlow e o MillWheel. Além disso, será necessário estudar o ecossistema da SPar e analisar a implementação da ResiFlow para identificar formas mais eficientes de integrar esse recurso. A hipótese desta etapa do projeto é que, ao tornar a ResiFlow uma ferramenta híbrida em relação ao processo de recuperação de falhas, além de oferecer um *framework* equilibrado entre desempenho e programabilidade, possibilitará ao usuário uma solução personalizada para suas necessidades.

A eficiência entre abordagens de *checkpointing* e *logging* pode variar de acordo com o tipo e o contexto da aplicação. Estratégias baseadas em *logging* geralmente apresentam maior granularidade, permitindo a restauração do sistema no exato ponto da falha. No entanto, essa abordagem pode impor um *overhead* maior durante a coleta e o armazenamento dos *logs*. A combinação dessas técnicas possibilita o desenvolvimento de um sistema de recuperação dinâmica, conciliando as vantagens de cada implementação.

Para avaliar a performance das soluções propostas, além das melhorias na ResiFlow, pretende-se portar um conjunto robusto de *benchmarks* para C++. Durante essa pesquisa, os *benchmarks* do DSPBench [Bordin et al. 2020], composto por 15 aplicações escritas em Java, serão convertidos para C++ a fim de fornecer uma base confiável e padronizada para a avaliação de *frameworks* de processamento de *stream* para essa linguagem. Serão realizadas análises de desempenho, comparando as diferentes implementações do protocolo de recuperação de falhas da ResiFlow, bem como a ResiFlow em si com outros *frameworks*. As métricas consideradas serão latência e *throughput*.

Os resultados desta etapa da pesquisa devem fornecer uma alternativa de protocolo de tolerância a falhas baseado em *log*, além de outras melhorias para a biblioteca ResiFlow. Essas implementações fortalecerão o ambiente de processamento de *stream* distribuído da SPar, tornando-o mais robusto e completo, sem comprometer os benefícios de programabilidade e produtividade já existentes. Em paralelo a isso, será disponibilizado a versão C++ do DSPBench.

Referências

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., and Fernandes, L. G. L. (2020). Dspbench: A suite of benchmark applications for distributed data stream processing systems. *IEEE Access*, 8:222900–222917.

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.