Proposta de geração de código paralelo tolerante a falhas com SPar para aplicações de stream

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. *Devido a necessidade de sistemas de processamento de stream serem executados por longos períodos de tempo, até mesmo de forma indefinida, faz se necessário a presença de mecanismos de tolerância a falhas para lidar com possíveis imprevistos. Atualmente, a SPar não fornece tolerância a falhas ou garantias de entrega de mensagens. Motivado por esse fator, o principal objetivo desta pesquisa envolve a exploração de formas de introduzir aspectos de tolerância a falhas na geração de código paralelo com a SPar.*

1. Introdução

O processamento de *stream* pode ser definido como um paradigma computacional que envolve a coleta, o processamento e a análise de um fluxo contínuo de dados heterogêneos em alto volume, com o objetivo de extrair percepções ou informações valiosas em tempo real [Andrade et al. 2014]. A SPar consiste em uma linguagem de domínio específico para C++ desenvolvida com o objetivo de simplificar a programação de aplicações de processamento de *stream* para diferentes arquiteturas paralelas [Griebler et al. 2017]. Inicialmente, a SPar focava na geração de código para sistemas *multi-core*. Porém, trabalhos posteriores expandiram o escopo para outras arquiteturas, como a GSParLib [Rockenbach et al. 2022], que permite a geração de código para GPUs, e a DSParLib [Löff et al. 2022], que viabiliza a geração de código para arquiteturas distribuídas.

Atualmente, nenhuma ferramenta do ecossistema SPar fornece tolerância a falhas ou garantias de entrega de mensagens. Motivado por esse fator, o principal objetivo desta pesquisa envolve a exploração de formas de introduzir aspectos de tolerância a falhas no ecossistema de *software* SPar, de forma que o usuário final não precise alterar seu código existente. Para alcançar esse objetivo, a ResiFlow foi desenvolvida como uma biblioteca C++ para processamento de *stream* distribuído tolerante a falhas. A ResiFlow é integrada ao ecossistema de software SPar como uma alternativa à geração de código para arquiteturas distribuídas, utiliza a biblioteca MPI (Message Passing Interface) para a comunicação entre processos, e implementa o protocolo *Asynchronous Barrier Snapshotting* (ABS) [Carbone et al. 2015] para criar *snapshots* periódicos do estado da aplicação.

2. Conclusões

Diferente da GSParLib e DSParLib, este trabalho implementa o suporte à geração de código com ResiFlow sobre uma versão experimental da SPar construída com base no *front-end* do compilador CLang, em vez da infraestrutura de compilador CINCLE. O projeto SPar CLang foi iniciado como um esforço para trazer a SPar a um *front-end* de compilador de nível comercial, uma vez que o projeto CINCLE foi desenvolvido inicialmente como um protótipo e não possui uma equipe dedicada para manter o projeto atualizado.

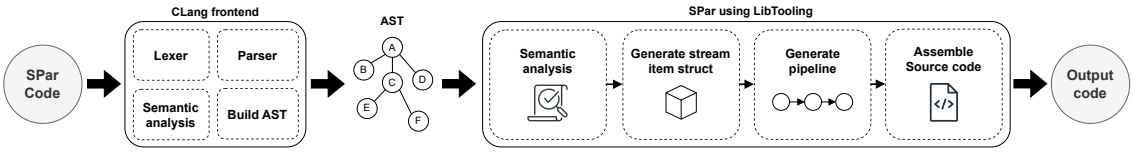

Figura 1. SPar code generation pipeline

Como ilustrado na Figura 1, o projeto SPar CLang envolve duas principais etapas para geração de código. Na primeira etapa, o código-fonte de entrada é processado pelo *front-end* de compilador CLang, que foi modificado para aceitar os atributos SPar como código fonte válido. O *front-end* CLang realiza a análise léxica e semântica, *parsing*, e gera uma Árvore de Sintaxe Abstrata (AST). Na segunda etapa, a AST resultante é processada por uma implementação da SPar desenvolvida utilizando a biblioteca LibTooling, que facilita a criação de ferramentas baseadas no CLang. Nesta segunda etapa, é realizada uma nova análise semântica para identificar as anotações SPar presentes na AST. As variáveis de entrada e saída das anotações SPar são identificadas e transformadas em uma *struct* que representa os dados trocados entre os estágios de processamento. As anotações de estágios SPar são transformados em um pipeline para a interface de programação de destino. Atualmente, o projeto SPar CLang suporta apenas a ResiFlow e o Intel TBB como interfaces para a geração de código. Por fim, a *struct* de dados de entrada e saída e o pipeline resultante são transformados no código-fonte de saída.

Para implementar o suporte à geração de código ResiFlow, somente as etapas relacionadas ao LibTooling precisaram ser modificadas. As modificações incluem alterações na geração da *struct* de entrada e saída, que foi adaptada para suportar a serialização de dados, e a etapa de geração do pipeline também foi modificada para gerar classes de operadores ResiFlow. Todas as aplicações que a SPar gera utilizando a ResiFlow como interface de programação se beneficiam do mecanismo de tolerância a falhas. Em caso de falha, a ResiFlow automaticamente reinicia a aplicação, que continua a sua execução a partir do último *snapshot* realizado.

Referências

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

Carbone, P., Fóra, G., Ewen, S., Haridi, S., and Tzoumas, K. (2015). Lightweight Asynchronous Snapshots for Distributed Dataflows.

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.

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