Proposta de Implementação de Grau de Paralelismo Adaptativo em uma DSL para Paralelismo de Stream

Adriano Vogel[1], **Dalvan Griebler**[1], **Luiz Gustavo Fernandes**[1]

[1]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS),
Grupo de Modelagem de Aplicações Paralelas (GMAP), Porto Alegre – RS – Brasil

adriano.vogel@acad.pucrs.br

Resumo. *A classe de aplicações de stream possuem características únicas, como variação nas entradas/saídas e execuções por períodos indefinidos. Este paradigma é utilizado com intuito de diminuir os tempos de execução e aumentar a vazão das aplicações. Nesse estudo é proposto o suporte adaptativo do grau de paralelismo de stream na DSL (Domain-Specific Language) SPar.*

1. Introdução

O paradigma de processamento de *streams* surgiu devido a mudanças nas características das aplicações e avanços tecnológicos, principalmente em bancos de dados e sistemas distribuídos [Andrade et al. 2014]. Isso ocorre pois um grande número de áreas do conhecimento necessitam coletar e tratar informações em tempo real de diversos dispositivos (sensores, câmeras, radares, entre outros). Aplicações de *streaming* possuem aspectos peculiares que as diferenciam das demais, apresentando principalmente um processamento contínuo, em tempo real e com variações constantes nas entradas de dados. A busca por qualidade dos serviços nessas aplicações frequentemente utiliza métricas como latência, vazão dos fluxos de dados, e utilização de processador e memória [Chakravarthy and Qingchun 2009].

Em busca de atingir bons resultados de desempenho nestes tipos de aplicação, a complexidade de programação somada a necessidade de conhecimento da arquitetura dificultam a tarefa de exploração do paralelismo. Por isso, diversos *frameworks* surgiram buscando facilitar a implementação do paralelismo, como o *FastFlow* e Intel *Thread Building Blocks* (TBB). Existem também DSLs, que são linguagens específicas de domínio que oferecem abstrações e facilidade no desenvolvimento.

SPar [Griebler 2016] é uma DSL interna para paralelismo de *stream*. Ela busca facilitar o desenvolvimento de aplicações de *streaming* e aumentar a produtividade de código. A ideia é que os programadores apenas anotem potenciais regiões a serem paralelizadas no código sequencial. As anotações da SPar são oferecidas através do mecanismo de atributos do padrão C++11. O compilador da DSL se encarrega de reconhecer os atributos da SPar e gerar código paralelo com o auxílio da biblioteca FastFlow. O paralelismo de *stream* ocorre no formato pipeline, onde o mesmo pode conter estágios replicados que delimitam o grau de paralelismo. Atualmente, o número de réplicas é definido estaticamente pelo programador [Griebler et al. 2016].

Embora existam abstrações de alto nível, as aplicações com códigos gerados pela SPar mostram um desempenho bom [Griebler 2016]. Não obstante, algumas aplicações de *stream* podem apresentar perdas de desempenho usando um número estático de réplicas devido ao princípio dinâmico das cargas de trabalho, que podem variar na execução.

Esta proposta compartilha características com trabalhos relacionados, explorar o *framework* FastFlow visto também nos estudos de [Sensi et al. 2016] e

[De Matteis and Mencagli 2016], e o domínio voltado para paralelismo de *Stream* também é usado em [Schneider et al. 2009] e [Su et al. 2015]. Porém, contrastando com os estudos relacionados que apresentam algoritmos com execução adaptativa, essa proposta busca também abstrair dos usuários o grau de paralelismo adaptativo.

2. Proposta

O objetivo é evitar que o desenvolvedor necessite especificar o grau de paralelismo adequado para uma aplicação de *streaming* na DSL SPar, adaptando-se a melhor configuração possível de desempenho. Além de abstrair, a dinamicidade também se faz necessária neste tipo de aplicação, pois a execução é normalmente indefinida e as cargas de trabalho variam durante a execução. Desta forma, o grau de paralelismo deve ser elástico para manter um bom desempenho na vazão do processamento dos elementos do *stream*.

A proposta de suporte adaptativo irá contemplar possivelmente uma *thread*/processo que irá fazer o monitoramento da aplicação. A mesma será responsável pela tomada de decisão e ação no aumento ou diminuição do grau de paralelismo. Na SPar, isso está diretamente ligado com o número de réplicas que um determinado estágio em uma região de paralelismo de *stream*.

O monitoramento pode estar relacionado com métricas como vazão, latência e uso de CPU. Após um estudo do estado da arte, será definido um modelo para calcular o grau de paralelismo ideal a partir dos resultados e métricas coletadas. Além disso, a proposta será habilitada através de uma *flag* de compilação, para que o compilador seja capaz de identificar quando gerar o código paralelo com suporte ao grau de paralelismo adaptativo.

Referências

[Andrade et al. 2014] Andrade, H., Gedik, B., and Turaga, D. (2014). *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge University Press.

[Chakravarthy and Qingchun 2009] Chakravarthy, S. and Qingchun, J. (2009). *Stream Data Processing: A Quality of Service Perspective: Modeling, Scheduling, Load Shedding, and Complex Event Processing*. Advances in Database Systems. Springer US.

[De Matteis and Mencagli 2016] De Matteis, T. and Mencagli, G. (2016). Keep calm and react with foresight: Strategies for low-latency and energy-efficient elastic data stream processing. *SIGPLAN*, 51(8):1–12.

[Griebler 2016] Griebler, D. (2016). *Domain-Specific Language & Support Tool for High-Level Stream Parallelism*. PhD thesis, Faculdade de Informática - PPGCC - PUCRS, Porto Alegre, Brazil.

[Griebler et al. 2016] Griebler, D., Danelutto, M., Torquati, M., and Fernandes, L. G. (2016). SPar: a DSL for High-Level and Productive Stream Parallelism. In *9th International Symposium on High-Level Parallel Programming and Applications*, HLPP'16, page 18, Munster, Germany. Springer.

[Schneider et al. 2009] Schneider, S., Andrade, H., Gedik, B., Biem, A., and Wu, K. L. (2009). Elastic scaling of data parallel operators in stream processing. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–12.

[Sensi et al. 2016] Sensi, D. D., Torquati, M., and Danelutto, M. (2016). A reconfiguration algorithm for power-aware parallel applications. *ACM Transactions on Architecture and Code Optimization (TACO)*, 13(4):43.

[Su et al. 2015] Su, Y., Shi, F., Talpur, S., Wang, Y., Hu, S., and Wei, J. (2015). Achieving self-aware parallelism in stream programs. *Cluster Computing*, 18(2):949–962.