Proposta de Adaptação Dinâmica de Padrões Paralelos

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

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

adriano.vogel@edu.pucrs.br,{dalvan.griebler, luiz.fernandes}@pucrs.br

Resumo. *Este trabalho apresenta uma perspectiva para adaptar dinamicamente os padrões paralelos em tempo de execução, objetivando abstrair dos programadores a definição de qual padrão paralelo usar e aumentar a flexibilidade. Os resultados preliminares demonstram a eficácia da solução proposta.*

1. Introdução

Aplicações de processamento de *stream*s executam por longos e/ou infinitos períodos de tempo, e paralelismo é relevante para aumentar o desempenho. É possível modelar uma aplicação paralela usando padrões paralelos ou combinando-os, resultando em configurações que impactam no desempenho e no consumo de recursos [Vogel et al. 2021b].

O programador usualmente testa os padrões paralelos e encontra uma configuração com desempenho adequado. Porém, em aplicações de processamento de *stream*s que computam dados em tempo real, tal prática é limitada pois usualmente ocorrem variações (carga, recursos, dados) durante a execução. Portanto, otimizações são aplicadas dinamicamente (em tempo de execução), como adaptar o grau de paralelismo [Vogel et al. 2021a] que é efetivo em padrões *Farms* com estágios replicados.

A flexibilidade e o espaço de customização poderiam ser aumentados adaptando dinamicamente o padrão paralelo utilizado, sendo uma adaptação complexa pois envolve modificar a estrutura da aplicação [Vogel et al. 2021b]. Por outro lado, é uma adaptação mais flexível e poderosa para um número maior de aplicações, tendo trabalhos que propuseram compilação dinâmica que altera a topologia da aplicação. A perspectiva neste trabalho é gerar diferentes configurações de padrões paralelos e encontrar em tempo de execução a mais adequada, objetivando aumentar a flexibilidade e QoS para aplicações dinâmicas, oferecendo para programadores uma camada de abstração adicional.

2. Solução Proposta e Resultados

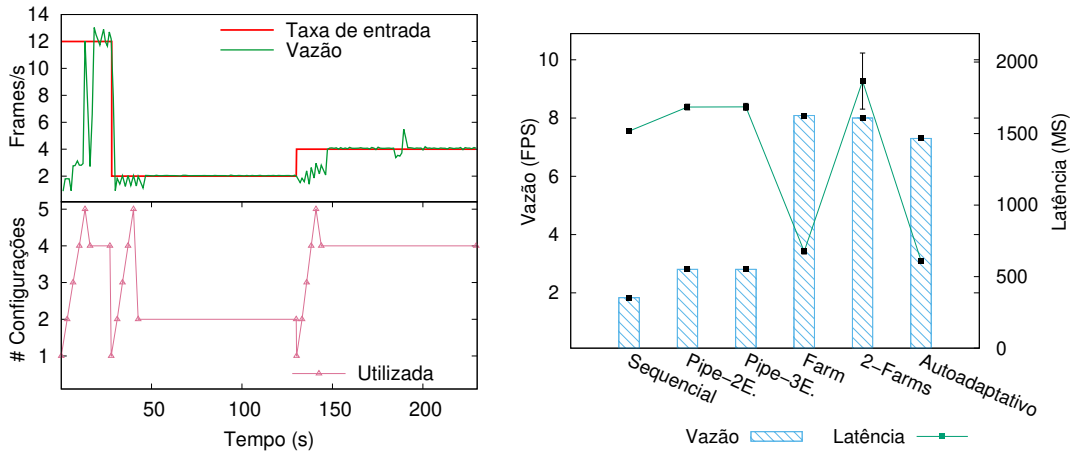
A configuração a ser usada é definida por uma estratégia de decisão proposta que possui uma etapa de treinamento das configurações criadas pelo programador ou sistema, ativando e suspendendo as configurações com transições entre elas. Após, é escolhida a configuração mais adequada conforme os objetivos não-funcionais definidos pelo programador da aplicação, como desempenho (vazão ou latência das tarefas) ou eficiência.

A estratégia proposta foi integrada no FastFlow [Aldinucci et al. 2017], que é um *framework* de programação paralela na linguagem C++. Abstraindo detalhes de implementação, a estratégia foi integrada com mecanismos para alterar as configurações (#) em tempo de execução. A solução foi avaliada em uma máquina multi-core com 2 Intel Xeon 2.40 GHz (12 cores- 24 *threads*) e 32 GB memória. A aplicação usada foi a *Person*

Recognition que detecta faces de pessoas em *streams* de vídeo. Foram criadas configurações como um *Pipeline* sequencial (#1), e outros *pipelines* com 2 e 3 estágios (#2 e #3) e configurações usando o padrão *Farm* que replica todas as funções da aplicação (#4) e outra com 2 estágios replicados (#5). Com a programação paralela estruturada, diversas configurações foram criadas sem duplicação de código. Porém, o quão adequadas são as configurações varia conforme as características da aplicação e dos objetivos.

A Figura 1(a) apresenta uma caracterização da solução autoadaptativa onde a taxa de entrada em *frames* por segundo oscila e o objetivo definido foi uma vazão não inferior a taxa de entrada e que consumisse a menor quantidade de recursos, sendo um exemplo para adaptar as configurações. Etapas de treinamento ocorrem a cada variação na taxa de entrada e as configurações criadas são testadas para encontrar a mais adequada.

A Figura 1(b) compara execuções estáticas (sem alterar a configuração) e autoadaptativa. A solução autoadaptativa aplicou a configuração mais adequada e a latência foi similar ao melhor caso estático. A média da vazão foi inferior que a melhor execução estática devido ao vídeo de entrada ser curto, aumentando o impacto da etapa de treinamento. Em execuções mais longas e estáveis o impacto da etapa de treinamento pode ser menor. Os resultados preliminares do estudo mostram que a solução funciona e está sendo aperfeiçoada, principalmente na etapa de treinamento da estratégia de decisão e para encontrar automaticamente alternativas de configurações adequadas para cada aplicação.



(a) Caracterização - vazão e configurações.

(b) Desempenho - taxa de entrada 8 FPS.

Figura 1. Aplicação *Person Recognition*.

Referências

Aldinucci, M., Danelutto, M., Kilpatrick, P., and Torquati, M. (2017). Fastflow: High-Level and Efficient Streaming on Multicore. *Programming multi-core and many-core computing systems, parallel and distributed computing*, pages 261–280.

Vogel, A., Griebler, D., and Fernandes, L. G. (2021a). Providing high-level self-adaptive abstractions for stream parallelism on multicores. *Softw Pract Exp*, pages 1–24.

Vogel, A., Mencagli, G., Griebler, D., Danelutto, M., and Fernandes, L. G. (2021b). Towards On-the-fly Self-Adaptation of Stream Parallel Patterns. In *Intern. Conf. on Parallel, Distributed and Network-Based Processing (PDP)*, page 5, Valladolid, Spain.