Benchmark Paramétrico para o Domínio do Paralelismo de *Stream:* Um Estudo de Caso com o Ferret da Suíte PARSEC

Carlos A. F. Maron[1,2], Dalvan Griebler[1,2], Luiz Gustavo Fernandes[1]

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

[2]Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC), Faculdade Três de Maio (SETREM), Três de Maio – RS – Brasil

{carlos.maron, dalvan.griebler}@acad.pucrs.br, luiz.fernandes@pucrs.br

Resumo. *Benchmarks são aplicações sintéticas que servem para avaliar e comparar o desempenho de sistemas computacionais. Torná-los parametrizáveis pode gerar condições diferenciadas de execuções. Porém, a técnica é pouco explorada nos tradicionais e atuais benchmarks. Portanto, esse trabalho avalia o impacto da parametrização de características do domínio de stream no Ferret.*

1. Introdução

A fim de entregar uma aplicação escalável e eficiente, é importante para o projetista de software paralelo entender os impactos causados no desempenho de uma aplicação. Porém, ter o entendimento de todas as características que impactam no desempenho pode ser complexo e exigir muito esforço. Inicialmente, porque aplicações têm diferentes propósitos, executam diferentes instruções, têm inúmeras variáveis de código, processam diferentes tipos e tamanho de dados, *etc*. Além disso, a arquitetura pode influenciar no desempenho, uma vez que diferentes tecnologias garantem que a execução seja mais rápida ou lenta. Desse modo, os *benchmarks* (programas sintéticos que representam aplicações reais) são ferramentas utilizadas para avaliar e comparar o desempenho de sistemas ou componentes [Arnold et al. 2015].

Independentemente da arquitetura, as funções que formam um *benchmark* têm comportamentos diferenciados durante a execução, sendo capaz de causarem impactos no desempenho das aplicações. Algumas pesquisas [Blumenthal et al. 2009] sugerem a parametrização como uma técnica para oferecer informações mais precisas sobre seus comportamentos. Eigenbench [Hong et al. 2010], por exemplo, é um *microbenchmark* que permite que características de aplicações de memórias transacionais sejam parametrizadas, gerando comportamentos e condições diferenciadas.

A técnica de parametrização ainda é pouco explorada nos tradicionais *benchmarks*, principalmente nos atuais para o domínio de processamento de *stream*. Em sistemas computacionais e na comunidade científica, é cada vez maior o destaque do domínio de processamento de *stream*, formado por aplicações de processamento de sinais, banco de dados, imagens, *Internet of Things* (IoT), e outras [Griebler et al. 2017, Hirzel et al. 2014]. Em particular, as aplicações desse domínio possuem algumas das seguintes características: janela deslizante, canais de comunicação, *buffers*, fluxo de *stream*, processamento infinito, e outras [Hirzel et al. 2014]. Um *benchmark* desse domínio de

processamento deve considerar tais características e, assim, conseguir representar uma aplicação real, tanto paralela quanto sequencial.

Embora Eigenbench seja um *microbenchmark* paramétrico, não é possível avaliar características específicas do domínio de *stream*. Por outro lado, o ambiente do StreamIt (linguagem de programação, compilador e *benchmarks*) [Thies 2010] oferece configurações parametrizáveis para o domínio de processamento de *stream*. No entanto, seus *benchmarks* não são portáveis e representam apenas o cenário de *data stream* e *dataflow*.

Portanto, o desafio desse trabalho é permitir a parametrização das características de aplicações do domínio de processamento paralelo de *stream* (*stream parallelism* em inglês). Para isso, foi escolhida a aplicação Ferret da suíte PARSEC [Bienia 2011], pois representa esse domínio de processamento. O presente trabalho expande alguns dos resultados encontrados em Maron [Maron et al. 2018]. As contribuições dessa pesquisa são:

- **Identificação das características do paralelismo de *stream* que influenciam nos comportamentos de *benchmarks* da suíte PARSEC.**
- **Suporte à parametrização das características do paralelismo de *stream* em *benchmarks* da suíte PARSEC.**

2. Resultados

Entender com clareza o paralelismo de *stream* foi determinante para definir as características parametrizáveis. Para isso, foi utilizado o Ferret da suíte PARSEC que faz parte desse domínio. Apesar de que *data flow*, *data stream* e sistemas reativos possuem características semelhantes ao paralelismo de *stream*, eles não são abordados nesse trabalho.

Ferret é uma aplicação de busca de similaridade de imagens. Sua versão original é paralelizada com POSIX Threads em um modelo *pipeline* de seis estágios. O primeiro (*Load*) e o último estágio são sequenciais, pois realizam a entrada e a saída dos elementos do *stream* (imagens nesse caso). O segundo estágio (*Seg*) realiza a segmentação, processo que organiza em conjuntos as regiões de cada imagem. O terceiro estágio (*Extract*) extraí 14 características das regiões segmentadas de cada imagem. O quarto estágio (*Vec*) aplica um método de indexação, que seleciona possíveis imagens semelhantes. O quinto estágio (*Rank*), realiza a busca refinada das imagens selecionadas no quarto estágio, classificando as imagens mais semelhantes.

O contexto da parametrização no Ferret foi desenvolvido com base nas características das aplicações de *stream*. Assim, as características parametrizáveis são: filas, tamanho e tipo do elemento do *stream*. Para entender o efeito da parametrização no Ferret, foram implementadas métricas de desempenho compatíveis com o domínio de processamento de *stream* [Hirzel et al. 2014]. As métricas de *throughput*, latência e *service time* são o reflexo do comportamento da aplicação. Uma vez que o PARSEC oferece somente o tempo de execução, essa métrica não é o suficiente para as aplicações de *stream*, dado que uma das características nesse domínio é o processamento infinito (nunca acaba).

Na aplicação de *stream*, o *throughput* é a quantidade máxima de itens processados em um intervalo de tempo. Latência é o tempo de comunicação entre os estágios, indicando o tempo gasto por um elemento ao trafegar de um estágio ao outro. *Service Time*

é o tempo total do elemento na aplicação, desde o momento que entrou para ser computado até a sua saída. Essa métrica contabiliza toda a computação aplicada ao elemento, incluindo a comunicação entre os estágios, processamento, operações de escrita, entre outros.

Para avaliar o Ferret, novos conjuntos de entradas foram criados para tornar mais realístico o cenário de testes. Esses conjuntos são dividido em quatro classes, que são chamadas de *Test, Light Duty Class, Heavy Duty Class* e *Free Class*. As classes *Light Duty* e *Heavy Duty* são divididas em outras três sub-classes, identificadas como L1 e H1, L2 e H2, LS e HS (*shuffled*). A numeração das sub-classes serve como referencia para o tamanho das imagens, onde *um* são imagens com resoluções variando entre 800x600, 678x435, 128x96 e cada arquivo não ultrapassam 1 MB; *dois* são imagens com resoluções entre 14400x7200, 3000x2000, 5472x3648 e cada arquivo não ultrapassa 14 MB. *Shuffled* é um conjunto aleatório de imagens criado com o proposito de avaliar a variação de entrada do *stream* no Ferret. Para isso, são utilizadas imagens de diferentes tamanhos, pois no Ferret o elemento do *stream* é representado por cada imagem. Todas as imagens estão no formato de arquivo JPEG.

Os testes foram executados em um servidor com dois processadores Intel Xeon E5-2620 v3 2.40 GHz, 12 núcleos e 24 *threads* (*Hyper-Threading*), 32 GB RAM, Ubuntu Server 64 bits (kernel 4.4.0-121-generic).Os *benchmarks* foram compilados com o GCC 5.4.0 utilizando a *flag* -Os. Para avaliar o efeito da parametrização, o cenário padrão indica os valores encontrados na versão original do Ferret no PARSEC. O cenário parametrizado utilizou os seguintes parâmetros: filas com tamanho 5; algoritmo de segmentação[1] com valores do *Q value* 128 e o *threshold* 0.005. A quantidade de imagens similares classificadas foi 50.

Ferret é uma aplicação dependente de latência [Bienia 2011] e a parametrização causou efeito positivo para a aplicação, como é demonstrado na Figura 1 da caracterização do desempenho nos cenários parametrizado (gráfico destacado) e padrão. Por limitações de espaço, somente a sub-classe H1 será demonstrada.

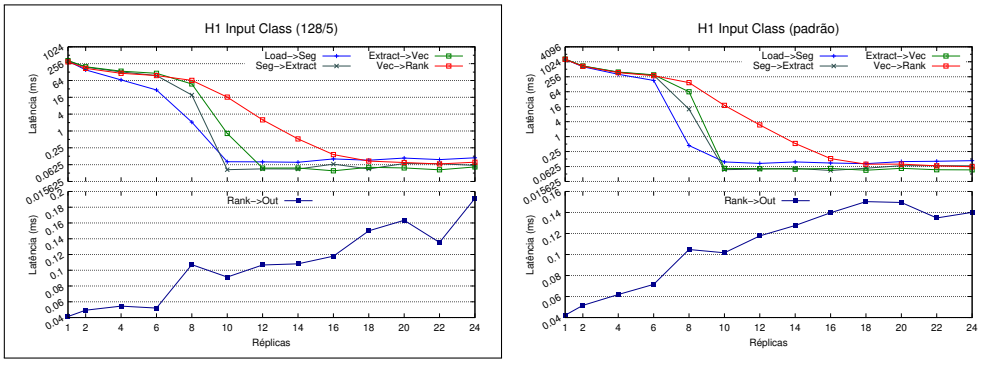

Figura 1. Latência do Ferret com a sub-classe H1 (plotado em escala logarítmica).

Com a caracterização do desempenho é possível ver o problema de escalabilidade existente no Ferret. A latência entre o estágio quatro (*vec*) e cinco (*rank*), indica que o estágio cinco realiza a maior parte da computação no Ferret e o desempenho da aplicação

[1] A segmentação da imagem define o tamanho do elemento do *stream*.

fica limitado à este estágio. O semelhante formato das curvas dos gráficos em ambos os cenários ocorre devido a frequente comunicação entre os estágios do Ferret causado pela janela deslizante com valor 1. Essa alta comunicação inviabiliza os benefícios do modelo *pipeline*, pois existe mais comunicação que computação.

3. Conclusões

Esse trabalho expande parte dos resultados de Maron [Maron et al. 2018], onde a técnica de parametrização das características das aplicações de *stream* vem sendo estudada e avaliada nos *benchmarks* da suíte PARSEC. Tal estudo é realizado pois as aplicações paralelas do domínio de *stream* possuem características que são compartilhadas entre si [Maron et al. 2018]. O conhecimento adquirido com o análise do comportamento de uma aplicação pode ser compartilhado entre as outras aplicações do mesmo domínio. Desse modo, com a parametrização implementada no Ferret, foi possível perceber o impacto no comportamento da aplicação devido a parametrização das características do domínio de *stream*. A parametrização possibilitou a melhora da latência na aplicação.

Como trabalhos futuros, pretende-se avaliar outras características do domínio de *stream*, como: múltiplos canais de entrada e saída do *stream*, janela deslizante. Além disso, avaliar as versões paramétricas dos *benchmarks* utilizando técnicas que adaptam o grau de paralelismo [Vogel et al. 2018].

Referências

[Arnold et al. 2015] Arnold, J. A., Huppler, K., Lange, K.-D., Henning, J. L., Cao, P., et al. (2015). How to Build a Benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, pages 333–336.

[Bienia 2011] Bienia, C. (2011). *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University.

[Blumenthal et al. 2009] Blumenthal, A., Luedde, M., Manzke, T., Mielenhausen, B., and Swanepoel, C. E. (2009). Measuring Software System Performance Using Benchmarks. US Patent 7,546,598.

[Griebler et al. 2017] 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.

[Hirzel et al. 2014] Hirzel, M., Soulé, R., Schneider, S., Gedik, B., and Grimm, R. (2014). A Catalog of Stream Processing Optimizations. *ACM Computing Surveys*, 46(4):1–46.

[Hong et al. 2010] Hong, S., Oguntebi, T., Casper, J., Bronson, N., Kozyrakis, C., and Olukotun, K. (2010). Eigenbench: A Simple Exploration Tool for Orthogonal TM Characteristics. In *Proceedings of the IEEE International Symposium on Workload Characterization*, pages 1–11.

[Maron et al. 2018] Maron, C. A. F., Vogel, A., Griebler, D., and Fernandes, L. G. (2018). Should PARSEC Benchmarks be More Parametric? A Case Study with Dedup. In *Proceedings of the 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. In press.

[Thies 2010] Thies, William; Amarasinghe, S. (2010). An Empirical Characterization of Stream Programs and Its Implications for Language and Compiler Design. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, pages 365–376.

[Vogel et al. 2018] Vogel, A., Griebler, D., Sensi, D. D., Danelutto, M., and Fernandes, L. G. (2018). Autonomic and Latency-Aware Degree of Parallelism Management in SPar. In *Euro-Par 2018: Parallel Processing Workshops*, page 12, Turin, Italy. Springer.