Avaliação da Usabilidade de Interfaces de Programação Paralela para Sistemas Multi-Core em Aplicação de Vídeo

Gabriella Andrade[1], Dalvan Griebler[1], Luiz Gustavo Fernandes[1]

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

{gabriella.andrade, dalvan.griebler}@edu.pucrs.br, luiz.fernandes@pucrs.br

Resumo. *Com a ampla variedade de interfaces para a programação paralela em ambientes multi-core é difícil determinar quais destas oferecem a melhor usabilidade. Esse trabalho realiza um experimento comparando a paralelização de uma aplicação de vídeo com as ferramentas FastFlow, SPar e TBB. Os resultados revelaram que a SPar requer menos esforço na paralelização de uma aplicação de vídeo do que as demais interfaces de programação paralela.*

1. Introdução

Com a popularização das arquiteturas multi-core, o desenvolvedor dispõe de uma variedade de linguagens e bibliotecas para programação paralela com abordagens de programação amplamente diferentes. Entretanto, a falta de resultados que caracterizem convincentemente a usabilidade dessas abordagens dificulta a escolha entre os desenvolvedores [Nanz et al. 2013]. Neste trabalho é realizado um experimento com o objetivo de comparar a usabilidade das ferramentas para programação paralela em C++ para ambientes multi-core: FastFlow[1], SPar (*Stream Parallelism*) e TBB (*Threading Building Blocks*)[2].

FastFlow e TBB são bibliotecas que abstraem a paralelização através da perspectiva de *skeletons*, que são esqueletos que estruturam o fluxo de um algoritmo. A SPar é uma DSL (*Domain-Specific Language*) que explora o paralelismo através da introdução de anotações no código fonte sem precisar reescrevê-lo, onde o seu compilador gera o código paralelo [Griebler et al. 2017].

2. Procedimento

Os participantes deste experimento foram 15 estudantes de mestrado em Ciência da Computação da PUCRS, matriculados na disciplina de Programação Paralela no 2º semestre de 2016. Eles possuem formação em diferentes cursos de graduação, como Ciência da Computação, Engenharia de Computação, Engenharia Elétrica, etc. Os estudantes receberam a tarefa de pararalelizar uma aplicação de processamento de vídeo OpenCV (*Open Source Computer Vision Library*), cujo o objetivo é extrair o canal verde do vídeo.

Os estudantes foram divididos em 3 grupos, variando a sequência das ferramentas usadas. Foi disponibilizado um manual para cada uma das ferramentas, sendo que apenas este manual podia ser consultado pelos participantes. A tarefa foi considerada completada somente quando: a versão paralela reduzisse o tempo de execução em 3x, a versão paralela produzisse o mesmo resultado da versão sequencial, e quando o formulário a respeito do

[1]Disponível para download em: https://github.com/fastflow/fastflow
[2]Disponível para download em: https://github.com/intel/tbb

experimento fosse devidamente preenchido. Além disso, o experimento foi monitorado por vídeo, permitindo a coleta de outras informações para uma análise futura. Assim como em [Nanz et al. 2013], foram utilizadas as seguintes métricas para avaliar o esforço de programação: Número de linhas de código (*Lines of Code* - LOC), tempo total gasto no desenvolvimento em minutos, tempo de execução da versão paralela em segundos.

3. Resultados e Conclusões

A Tabela 1 apresenta os resultados obtidos através do experimento realizado. Os resultados representam a média dos resultados individuais de cada um dos 15 participantes. Conforme pode ser observado, a SPar obteve a menor média de LOC. Isso ocorre devido ao modelo de programação, onde apenas anotações são introduzidas no código fonte sem a necessidade de reescrevê-lo. Com TBB e FastFlow é necessário modificar o código criando classes para cada estágio do pipeline. A SPar também obteve o menor tempo médio de desenvolvimento, o que também pode ser explicado pelo modelo de programação dessa linguagem.

Tabela 1. Resultados de FastFlow, SPar e TBB em uma Aplicação de Vídeo.

		LOC	Tempo de Desenv. (min.)	Tempo de Exec. (s)
FastFlow	Média	108,27	94,00	10,50
	Desv. Padrão	6,28	68,68	1,03
SPar	Média	79,87	56,53	11,03
	Desv. Padrão	1,45	22,77	1,03
TBB	Média	117,53	126,67	11,52
	Desv. Padrão	7,51	80,38	0,87

A versão com FastFlow obteve o menor tempo médio de execução em relação ao TBB e SPar, onde o número de *threads* que levou ao menor tempo de execução para cada participante variou entre 3, 4, 5, 6, 7 e 8. Outras métricas como speedup e eficiência não foram avaliadas neste trabalho, pois o objetivo é avaliar o esforço de programação. Neste caso, o menor tempo de execução paralelo não indica que a paralelização foi a mais eficiente. Pois um dos critérios da tarefa era o de reduzir o tempo de execução em relação a versão sequencial em 3x. Logo, nem todos participantes avançaram neste quesito. O que se pode concluir até o momento é que a SPar requer menos esforço no processo de paralelização de uma aplicação de vídeo.

Nesse experimento foi considerado o tempo total de desenvolvimento, incluindo o tempo de consulta do manual, o tempo gasto entendendo a aplicação sequencial, o tempo corrigindo erros no código, etc. Em trabalhos futuros, pretende-se analisar os dados das gravações com objetivo de verificar o impacto destes fatores no resultado. Além disso, pretende-se verificar outros fatores como o conhecimento prévio dos alunos, sua formação acadêmica, quais erros foram cometidos e as principais dificuldades encontradas.

Referências

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.

Nanz, S., West, S., Da Silveira, K. S., and Meyer, B. (2013). Benchmarking usability and performance of multicore languages. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 183–192. IEEE.