Uso de Métricas de Codificação para Avaliar a Programação Paralela nas Aplicações de Stream em Sistemas Multi-core

Gabriella Andrade[1], Dalvan Griebler[1], Rodrigo Santos[2], Luiz G. Fernandes[1]

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

[2]Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Rio de Janeiro, Brasil

gabriella.andrade@edu.pucrs.br

Resumo. *Neste trabalho, sete métricas de codificação são avaliadas considerando quatro aplicações do mundo real implementadas com FastFlow, Pthreads, SPar e TBB. Nossos resultados mostram que SPar apresenta os melhores indicadores de acordo com as métricas utilizadas.*

1. Introdução

O desenvolvimento de aplicações paralelas é mais difícil quando comparado com a programação sequencial. Os programadores devem se preocupar com vários aspectos de paralelismo ao longo de todo o processo de desenvolvimento de aplicações paralelas. Por exemplo, os programadores devem implementar a sincronização dos dados, dividir o problema de computação entre as *threads*, explorar a concorrência, e fornecer otimização de hardware de baixo nível [McCool et al. 2012]. Para tornar este desenvolvimento mais fácil, foram criadas novas interfaces de programação paralela (IPP). A maioria das IPPs propostas se concentra na avaliação do desempenho de uma aplicação sem considerar o esforço de desenvolvimento, tornando difícil determinar qual IPP permite melhor usabilidade. Outros trabalhos na área usaram algumas métricas de codificação para avaliar a usabilidade, como Halstead, complexidade ciclomática (CCN) e linhas de código (LOC) [Fernàndez et al. 2019]. Estas métricas são normalmente utilizadas na Engenharia de Software de sistemas tradicionais, embora sejam também usadas no contexto de programação paralela [Griebler et al. 2014]. Logo, o objetivo deste estudo é avaliar a aplicabilidade destas métricas na programação paralela.

2. Resultados e Conclusões

Neste estudo, avaliamos as métricas número de LOC, número de caracteres (NOC), número de tokens (TOC), CCN, complexidade do fluxo de informações (IFC), Halstead e COCOMO II aplicadas a programação paralela. Para avaliar essas métricas, nós realizamos um experimento com o objetivo de verificar a usabilidade das IPPs FastFlow[1], Pthreads, SPar[2] [Griebler et al. 2017] e TBB[3] no desenvolvimento de quatro aplicações de processamento de stream (Bzip2, Dedup, Person Recognition e Lane Detection). Para comparar o tempo de desenvolvimento estimado pelas métricas Halstead e COCOMO II, convertemos ambos resultados para dias necessários para desenvolver a aplicação. A Tabela 1 apresenta os resultados obtidos por meio do experimento realizado. Para todas as aplicações, SPar obteve o menor número de LOC, NOC e TOC. Isso ocorre devido ao seu modelo de programação, no qual apenas anotações são introduzidas no código fonte

[1]https://github.com/fastflow/fastflow
[2]https://gmap.pucrs.br/spar-wiki/
[3]https://github.com/oneapi-src/oneTBB

sem a necessidade de alterá-lo. FastFlow e TBB apresentaram resultados semelhantes para o valor de LOC, NOC e TOC, pois para todas as aplicações foram criadas estruturas (`class` ou `struct`) para cada estágio do *pipeline*. Pthreads apresentou o pior resultado, pois usa várias sub-rotinas para gestão de *threads*, sincronização e *mutex*.

SPar apresenta a menor complexidade CCN para todas as aplicações com excessão do Dedup. Isto ocorre porque a métrica CCN considera as anotações de SPar como estruturas condicionais. Dessa forma, a mesma região do código da aplicação Dedup tem CCN igual a 47 para SPar e 29 para TBB. Diferente de CCN, a métrica IFC apresentou resultados de complexidade iguais a zero. Isso ocorre em aplicações como Lane Detection e Person Recognition, em que a aplicação sequencial tem apenas a função principal. Ao paralelizar essas aplicações, sua estrutura é mantida, pois não é necessário criar qualquer estrutura de dados para paralelizar a aplicação com SPar. Se o programa contiver apenas a função principal no código, a complexidade é igual a zero, mesmo que haja uma complexidade na adição das diretivas paralelas.

Tabela 1. Resultado das Métricas de Codificação.

Aplicação	IPP	LOC	NOC	TOC	CCN	IFC	COCOMO II (dias)	Halstead (dias)
Bzip2	FastFlow	1991	54071	14184	431	5341	178,09	61,36
	Pthreads	2312	61223	16199	473	6289	185,20	74,69
	SPar	1796	49270	13148	392	5118	173,34	52,94
	TBB	1868	51136	13732	404	7234	175,14	58,33
Dedup	FastFlow	1027	32688	6833	192	3726	149,73	21,14
	Pthreads	1052	35243	7322	196	1164	150,67	20,54
	SPar	602	20261	4313	119	3088	130,17	8,90
	TBB	629	21121	4356	115	24	131,68	10,43
Lane Detection	FastFlow	166	4384	1256	22	49	92,88	1,17
	Pthreads	360	9639	2187	41	888	113,76	3,31
	SPar	121	3496	1024	13	0	85,49	0,81
	TBB	168	4648	1331	22	49	93,17	1,47
Person Recognition	FastFlow	194	6586	1557	28	49	96,75	1,78
	Pthreads	393	11250	2476	46	888	116,41	3,78
	SPar	145	5687	1239	18	0	89,64	1,18
	TBB	193	6808	1560	25	49	96,62	1,86

Halstead e COCOMO II apresentaram resultados diferentes porque o modelo Halstead não considera aspectos essenciais do desenvolvimento de software, como o perfil dos programadores. O COCOMO II parece ser um modelo mais completo em relação ao Halstead. Porém, não considera a experiência dos programadores no desenvolvimento de aplicações paralelas. Além disso, COCOMO II considera que a aplicação será implementada do zero, sem considerar a inserção de diretivas paralelas no código. Portanto, concluímos que, embora as métricas sejam promissoras para a avaliação das aplicações paralelas, não consideram características particulares de tais aplicações. Em trabalhos futuros iremos focar no desenvolvimento de uma métrica que considere em sua avaliação os fatores que impactam no desenvolvimento de aplicações paralelas.

Referências

Fernàndez, J. F., Escribano, A. G., and Llanos, D. R. (2019). Simplifying the multi-gpu programming of a hyperspectral image registration algorithm. In *2019 International Conference on High Performance Computing & Simulation*, pages 11–18. IEEE.

Griebler, D., Adornes, D., and Fernandes, L. G. (2014). Performance and Usability Evaluation of a Pattern-Oriented Parallel Programming Interface for Multi-Core Architectures. In *The 26th International Conference on Software Engineering & Knowledge Engineering*, pages 25–30, Vancouver, Canada. KSIGS.

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.

McCool, M., Reinders, J., and Robison, A. (2012). *Structured parallel programming: patterns for efficient computation*. Elsevier.