Avaliação do Esforço de Programação em GPU: Estudo Piloto

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 – RS – Brasil

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

Resumo. *O desenvolvimento de aplicações para GPU não é uma tarefa fácil, pois exige um maior conhecimento da arquitetura. Neste trabalho realizamos um estudo piloto para avaliar o esforço de programadores não-especialistas ao desenvolver aplicações para GPU. Os resultados revelaram que a GSParLib requer menos esforço em relação as demais interfaces de programação paralela. Entretanto, mais investigações são necessárias a fim de complementar o estudo.*

1. Introdução

A programação para GPU é mais desafiadora quando comparada à programação paralela para ambientes multi-core. Isso ocorre devido à hierarquia das *threads*. Na GPU algumas *threads* compõem um *warp*, alguns *warps* compõem um bloco de *threads*, e alguns blocos de *threads* compõem um *grid* [Wu and Shen 2017]. Nesse sentido, alguns esforços têm sido realizados visando avaliar a produtividade de codificação para esse tipo de arquitetura [Miller and Arenaz 2019, Fabeiro et al. 2019, Wu et al. 2019].

Este trabalho apresenta um estudo piloto, visando avaliar a produtividade de interfaces para programação em GPU: CUDA, GSParLib [Rockenbach 2020], OpenACC, e OpenCL. Esse estudo foi realizado com alguns participantes de modo a avaliar o plano de experimentação. Logo, o estudo pode ser refinado antes ser realizado em maior escala.

Os participantes deste estudo piloto foram 4 estudantes de mestrado em Ciência da Computação da PUCRS. Eles possuem certa experiência com o desenvolvimento de aplicações paralelas para ambiente multi-core e *cluster*. Entretanto, apenas um participante possui experiência no desenvolvimento de aplicações para GPU. Os estudantes receberam a tarefa de pararalelizar o problema do *Animal Rescue* com *drones*, cujo objetivo é encontrar o ambiente mais adequado para reintroduzir animais resgatados na natureza. Para comparar o esforço de programação avaliamos o tempo gasto na paralelização (em horas) e o tamanho do código (linhas de código - SLOC) [Fabeiro et al. 2019, Wu et al. 2019]. Além disso, consideramos o tempo de execução da versão paralela (em segundos).

2. Resultados e Conclusões

A Tabela 1 apresenta os resultados obtidos, os quais representam a média dos resultados individuais de cada um dos 4 participantes para as métricas SLOC, tempo de desenvolvimento e tempo de execução. Conforme pode ser observado, OpenACC obteve a menor média de SLOC. Entretanto, os participantes conseguiram paralelizar a aplicação com menos esforço utilizando a GSParLib. Logo, o menor número de SLOC não indica o menor esforço de programação. Além disso, o menor esforço de desenvolvimento não garante o melhor desempenho. Entretanto, esse estudo não visa a avaliação do desempenho, logo métricas como *speedup* e eficiência não foram consideradas. Pois, neste estudo o melhor *speedup* não indica necessariamente o desempenho ideal.

Embora GSParLib forneça o menor tempo médio de desenvolvimento, apenas o valor das médias não pode determinar qual interface provê a melhor produtividade, logo

é necessário realizar um teste de hipóteses para verificar se existe ou não uma diferença significativa entre as médias. Considerando o nível de significância convencional (α) de 0,05, foi realizado o teste de Shapiro-Wilk para verificar se as amostras apresentam uma distribuição normal. A Tabela 1 mostra que para todas as interfaces o valor de p é maior que o valor de α, logo todas as amostras apresentam uma distribuição normal. Dessa forma um teste de hipótese paramétrico deve ser realizado.

Tabela 1. Resultados de CUDA, GSParLib, OpenACC, e OpenCL.

	SLOC	Tempo de Desen. (hr)	Tempo de Execução (s)	Shapiro-Wilk - p-value
CUDA	999,75	18,33	**4,35**	**0,5041**
GSParLib	1005,25	**6,17**	5,67	**0,0657**
OpenACC	**803,00**	9,25	146,06	**0,4162**
OpenCL	998,50	9,25	4,75	**0,2041**

Para verificar se existe uma diferença significativa entre os tempos médios de desenvolvimento para cada interface, foi realizado o teste *t* de *Student* pareado. A Tabela 2 mostra que conforme esperado, não existe uma diferença significativa entre as médias do OpenACC e OpenCL. Além disso, embora a média do GSParLib seja menor que a do OpenACC, não há uma diferença significativa entre essas médias. Entretanto, GSParLib provê maior desempenho em relação ao OpenACC. Para todos os outros casos, o teste de hipóteses confirma que existe uma diferença significativa entre as médias.

Tabela 2. Resultados dos Testes *t* de *Student*.

	GSParLib x OpenACC	GSParLib x OpenCL	GSParLib x CUDA	OpenACC x OpenCL	OpenACC x CUDA	OpenCL x CUDA
p-value	**0,2992**	0,0076	0,0052	1	0,04439	0,0099

Os resultados preliminares indicam que a GSParLib exige um menor esforço de desenvolvimento em relação às outras interfaces. Além disso, é mais custoso desenvolver aplicações com CUDA. Os resultados podem ter sido impactados pelo efeito de aprendizagem, pois os participantes iniciaram suas atividades com CUDA. Logo esse estudo apresenta limitações. Em trabalhos futuros pretendemos realizar um estudo com mais participantes a fim de responder essas questões. Além disso será realizada uma análise qualitativa para avaliar a satisfação dos participantes em relação as interfaces utilizadas.

Referências

Fabeiro, J. F., Escribano, A. G., and Llanos, D. R. (2019). Simplifying the multi-gpu programming of a hyperspectral image registration algorithm. In *HPCS 2019*, pages 11–18. IEEE.

Miller, J. and Arenaz, M. (2019). Measuring the impact of hpc training. In *EduHPC 2019*, pages 58–67. IEEE.

Rockenbach, D. A. (2020). High-Level Programming Abstractions for Stream Parallelism on GPUs. Master's thesis, PPGCC - PUCRS, Porto Alegre, Brazil.

Wu, B. and Shen, X. (2017). Software-level task scheduling on gpus. In *Advances in GPU Research and Practice*, Emerging Trends in Computer Science and Applied Computing, chapter 4, pages 83–103.

Wu, S., Dong, X., Wang, Y., and Chen, W. (2019). Language constructs and semantics for runtime-independent parallelism expression on heterogeneous systems. In *ICCC 2019*, pages 1269–1275. IEEE.