AVALIAÇÃO E CARACTERIZAÇÃO DE APLICAÇÕES DO BENCHMARK POLYBENCH/GPU NO CONTEXTO DE SISTEMAS HETEROGÊNEOS

DOUGLAS WAHAST DA COSTA[1]; MATEUS SANTOS DE MELO[2]; CARLOS MICHEL BETEMPS[2]; BRUNO ZATT[2]

[1]Universidade Federal de Pelotas – dwdcosta@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas – {msdmelo, cm.betemps, zatt}@inf.ufpel.edu.br

1. INTRODUÇÃO

Aplicações computacionais possuem características específicas nas quais demandam processamento ímpar, de forma que podem ser executadas em hardware especializado para se obter melhor desempenho em termos de, por exemplo, tempo de execução e energia consumida (SINGH et al. 2017). Sistemas Embarcados Heterogêneos (SEH) proporcionam uma maior flexibilidade na utilização de seus diferentes elementos de processamento (EPs) - por exemplo: CPU, GPU, FPGA, entre outros - na execução de suas aplicações. MPSoCs (*Multi-Processor Systems-on-Chips*) se enquadram como sistemas heterogêneos, tipicamente contendo núcleos de processamento como CPU e GPU (SINGH et al. 2017). Um exemplo de MPSoC é a plataforma Odroid-XU3 (HardKernel, 2018).

A utilização de Benchmarks é essencial para que se possa ter um referencial de comparação entre diferentes trabalhos. Normalmente, benchmarks incluem diferentes tipos de aplicações, cada uma com suas características específicas, e que permitem observar, dada sua execução, os diferentes desempenhos de acordo com o conjunto de EP's utilizado. O benchmark Polybench GPU (PolyBench/GPU, 2012; GRAUER-GRAY et al., 2012) inclui 15 aplicações para as EPs CPU e GPU.

Simulação em nível de sistema utilizam modelos de mais grossa granularidade para representar arquitetura e aplicações (GRIES, 2004), permitindo um reduzido tempo de simulação e, possivelmente, uma ampla exploração do espaço de projeto. Alguns simuladores utilizam grafo de tarefas (MIELE, 2015) para a representação da carga de trabalho (aplicações). Portanto, a caracterização das aplicações PolyBench/GPU e respectivas representações por grafo de tarefas amplia as possibilidades de exploração das configurações possíveis para um SEH por meio do uso de simuladores de nível de sistema.

Este trabalho apresenta um estudo inicial sobre as características de algumas aplicações do PolyBench/GPU no contexto de execução na plataforma Odroid-XU3 e considerando, como métricas de análise, dados de latência (tempo) das tarefas das aplicações e potência dissipada pelos núcleos de processamento (CPU e GPU).

2. METODOLOGIA

Tendo como instrumento inicial de estudo o benchmark Polybench/GPU (PolyBench/GPU, 2012; GRAUER-GRAY et al., 2012), foi analisado o código-fonte das aplicações implementado na linguagem C com a utilização da biblioteca, de código aberto, OpenCL (KAELI et al., 2015). As aplicações PolyBench/GPU (GRAUER-GRAY et al., 2012) utilizadas nos experimentos estão descritas na Tab. 1. Cada aplicação do referido benchmark foi modelada e caracterizada obtendo-se um modelo mais abstrato - um grafo de tarefas, semelhante ao apresentado em (MIELE, 2015). Cada tarefa das aplicações foi identificada, modelada e, a partir de sua

execução na plataforma Odroid XU3 (HardKernel, 2018), caracterizada de acordo com sua latência e frequência de operação do EP utilizado na execução.

Tabela 1: Aplicações PolyBench/GPU Utilizadas nos Experimentos

Aplicação	**Descrição**	**Categoria**	**Dimensões do(s) *Array(s)***
3DConvolution	Convolução 3D	Convolução	256
3MM	3 Multiplicação de Matrizes	Álg. Linear	512
GRAMMSCHMIDT	Processo de Gram-Schmidt	Álg. Linear	2048
SYRK	Operações Simétricas de rank-k	Álg. Linear	1024
COVAR	Computação de Covariância	*Datamining*	2048
FDTD-2D	Domínio do Tempo da Diferença Finita 2D	*Stencils*	2048 (500 passos de tempo)

Conforme parágrafo anterior, após a modelagem das aplicações, o código das mesmas foi alterado com intuito de gerar (na forma de *logs*) os tempos (latências) de cada tarefa constituinte da aplicação. Para a instrumentalização do código das aplicações foram utilizadas funções referentes ao tempo de execução, existentes na biblioteca “time.h”, assim como funções específicas do código OpenCL (da biblioteca “cl.h”), respetivamente: A função “rtclock()” e o codigo “clGetEventProfilingInfo()”.

A partir da execução de aplicações do benchmark Polybench/GPU também foram caracterizados os elementos de processamento da plataforma Odroid-XU3 quanto à dissipação de potência. Em razão dos testes de potência é que foi escolhida a placa Odroid-XU3 (HardKernel, 2018). Esta plataforma contém sensores que possibilitam essa análise e, desta forma, foi escolhida como base de processamento das aplicações. Para a medição de energia, foi preparado um *script* de execução para o retorno dos valores acumulados de energia consumida por cada aplicação executada, especificamente a energia consumida pelos seus núcleos de processamento. A plataforma Odroid-XU3 contém um *cluster* de processadores de alto desempenho (A15) e um para tarefas mais leves, de menor desempenho e menor consumo de energia (A7), ambos *clusters* contém 4 núcleos de processamento. Esta placa ainda contém uma GPU de modelo Mali-T628. Nos testes realizados foram considerados dados de potência e de tempo referentes ao *cluster* A15 e a GPU Mali-T628, executados em 2GHz e 0,6GHz, respectivamente. Cada aplicação foi executada 30 vezes na obtenção das latências e potências.

3. RESULTADOS E DISCUSSÃO

A partir dos modelos gerados para as aplicações, foi possível perceber um certo padrão estrutural nas aplicações, principalmente no que tange os trechos de inicialização de memória, preparação para execução dos kernels e liberação de memória. As diferenças mais perceptíveis estão nos arquivos que implementam os Kernels (*.cl) das aplicações, no quais se encontram as partes do código em que o processamento inerente de cada aplicação ocorre, considerando a resolução de suas fórmulas, algoritmos e iterações. Estes são trechos de execução crítica de cada aplicação, podendo variar quanto ao número de Kernels empregado, o número de threads disparadas em paralelo, e a existência ou não de laços de repetição na execução do(s) kernel(s). A Fig. 1 apresenta um grafo de tarefas que descreve genericamente a estrutura das aplicações. Observa-se que há um laço de repetição (cujo número de repetições é dado pela variável N) para a execução iterativa do(s) kernel(s). As aplicações 3MM, SYRK e COVAR não utilizam o referido laço, portanto o valor de N nestes casos é zero (0). Por outro lado, para a aplicação 3DConvolution

o valor de N é igual a (256 - 3), na aplicação GRAMMSCHMIDT N é (2048 - 1), e para FDTD-2D o valor de N é (500 - 1). As tarefas *t6* e *t8* são estruturais e somente indicam o início e a finalização do(s) kernel(s), respectivamente.

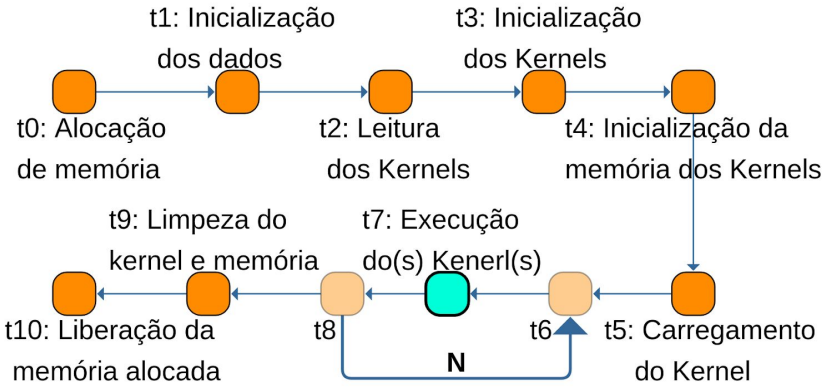

Figura 1: Grafo de Tarefas que Descreve Genericamente as Aplicações Analisadas.

Tabela 2: Latências (segundos) de cada Tarefa e Kernels usando CPU como *Device*.

Tarefa	3DCONV	3MM	COVAR	FDTD-2D	GRAMM	SYRK
t0	0,00010	0,00015	0,00013	0,00013	0,00008	0,00010
t1	0,05071	0,00302	0,01208	0,03599	0,06526	0,00622
t2	0,00439	0,00010	0,00014	0,00024	0,00278	0,00011
t3	0,00005	0,00005	0,00005	0,00006	0,00006	0,00005
t4	0,36477	0,01530	0,07550	0,11918	0,08337	0,01844
t5	0,25040	0,22546	0,22373	0,22986	0,27637	0,24020
t7	0,42932	1,42377	55,28151	37,96279	417,95588	0,67374
t9	0,02343	0,00226	0,00606	0,00958	0,01747	0,00188
t10	0,02678	0,00139	0,00645	0,01484	0,01165	0,00262
Total	1,14994	1,67150	55,60567	38,37266	418,41293	0,94335

Tabela 3: Relação entre as Latências de cada Tarefa e Kernels das Aplicações usando *Device* GPU em relação às Latências apresentadas na Tab. 2.

Tarefa	3DCONV	3MM	COVAR	FDTD-2D	GRAMM	SYRK
t1	3,09	3,49	3,29	3,44	1,81	3,35
t2	0,02	0,94	0,78	0,52	0,07	1,78
t3	808,22	964,16	818,91	777,95	1051,81	885,84
t4	1,29	1,55	1,29	1,11	1,88	1,25
t5	0,21	0,34	0,37	0,28	0,38	0,19
t7	0,55	0,52	4,19	0,67	0,10	5,03
t9	0,02	0,21	0,09	0,05	0,03	0,17
Total	0,86	0,54	4,18	0,68	0,10	3,74

A Tab. 2 apresenta os valores de latência para cada tarefa identificada na Fig. 1 para cada uma das aplicações experimentadas considerando como *device* para execução dos kernels a CPU (A15). A Tab. 3 apresenta as relações entre os valores de latência das tarefas considerando como *device* a GPU (Mali-T628) - latência com *device* GPU dividido por latência com *device* CPU (tarefas *t0* e *t10* são desconsideradas neste caso pois não sofrem influência do *device* adotado). Em

relação ao desempenho, observa-se que para as aplicações 3DCONV, 3MM, FDTD-2D e GRAMM o uso de GPU é vantajoso em relação às latências obtidas. Para COVAR e SYRK é melhor utilizar como *device* a CPU.

Quanto a potência dissipada por cada EP (A15 e GPU), as Tab. 4 e 5 apresentam os valores de potência (em Watts) considerando como *device* CPU e GPU, respectivamente. Ainda, mostra valores totais de potência considerando também a memória do sistema. Pode ser verificado que o uso de GPU proporciona menores valores para potência em todas as aplicações.

Tabela 4: Potência dissipada (Watts) em cada PE/Aplicação usando *Device* CPU.

	3DCONV	3MM	COVAR	FDTD-2D	GRAMM	SYRK
A15	4,6233	5,2389	3,9190	3,8596	3,6609	8,6595
GPU	0,0756	0,0652	0,0618	0,0628	0,0620	0,0960
Total (Mem.)	4,8040	5,3466	4,0953	4,1693	4,0399	8,8922

Tabela 5: Potência dissipada (Watts) em cada PE/Aplicação usando *Device* GPU.

	3DCONV	3MM	COVAR	FDTD-2D	GRAMM	SYRK
A15	2,5797	0,7882	0,7330	0,7872	0,7898	0,9455
GPU	0,8967	1,5387	0,9727	1,5802	1,0510	2,1486
Total (Mem.)	3,6859	2,4150	2,1963	2,6826	2,1272	3,2832

4. CONCLUSÕES

Este trabalho apresentou uma avaliação inicial da execução de aplicações do benchmark PolyBench numa plataforma heterogênea (Odroid-XU3). Foi verificado que as aplicações obtêm desempenhos diferentes de acordo com o EP empregado para a execução de seus *kernels*. A potência dissipada considerando os *devices* CPUs e GPUs para execução dos *Kernels* deve ser avaliada de acordo com as restrições definidas no projeto, visto que o uso de GPU geralmente propicia menores valores nesta métrica, mas nem sempre os valores de melhor desempenho.

5. REFERÊNCIAS BIBLIOGRÁFICAS

GRAUER-GRAY, Scott et al. Auto-tuning a high-level language targeted to GPU codes. In: **Innovative Parallel Computing** (InPar). IEEE. p. 1-10, 2012.

GRIES, Matthias. Methods for evaluating and covering the design space during early design development. **Integration**, the VLSI journal, v. 38, n. 2, p. 131-183, 2004.

HardKernel. **Odroid-XU3**. Hardkernel co., 2018. Acessado em 05 set. 2018. Online. URL:http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127

KAELI, David R. et al. **Heterogeneous computing with OpenCL 2.0**. Morgan Kaufmann, 2015.

MIELE, Antonio et al. A System-Level Simulation Framework for Evaluating Resource Management Policies for Heterogeneous System Architectures. In: **Digital System Design** (DSD), 2015 Euromicro Conference on. IEEE. p. 637-644, 2015.

PolyBench/GPU. **Implementation of PolyBench codes for GPU processing**. *University of Delaware*, 2012. Acessado em 05 set. 2018. Online. Disponível em: http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/GPU/index.html

SINGH, Amit K. et al. Energy-Efficient Run-Time Mapping and Thread Partitioning of Concurrent OpenCL Applications on CPU-GPU MPSoCs. **ACM Transactions on Embedded Computing Systems** (TECS), v. 16, n. 5s, p. 147, 2017.