

Artifact Description Appendix

ABSTRACT

This appendix provides information of our experimental implementations, including the computing platform, software environment and experimental setup. We also open-source our project that can reproduce our experimental results.

1 HARDWARE CONFIGURATIONS

We implement our experiments on the machine of which the configurations are listed in Table 1. On our tested NVIDIA GTX 980, we cover both core frequency and memory frequency at a 2× range of scaling from 500 MHz to 1,000 MHz with a step size of 100 MHz.

Table 1: target gpu frequency configurations

Device	GTX 980
Compute capability	5.2
SMs * cores per SM	16 * 128
Global memory bus width	256-bit
Global memory size	4 GB
Core frequency scaling	[500, 600, 700, 800, 900, 1,000] MHz
Memory scaling	[500, 600, 700, 800, 900, 1,000] MHz
CPU	Intel Core i7-3770 4-core 3.4 GHz
Host Memory	16 GB DDR3

2 SOFTWARE CONFIGURATIONS

With the help of the NVIDIA Inspector [5], we can fix the performance state and adjust the core frequency and memory frequency of the GPU within a certain range. Using this method we can obtain the execution time data with certain frequency combinations of GPU.

We use the NVIDIA Profiler [4] to extract the performance counters at the baseline core and memory frequency to drive our model. Note that we only need one time data collection, which makes our model run fast. We use NVML API [3] to record the power consumption every 20 milliseconds.

3 EXPERIMENTAL SETUP

We validate our model among 20 realistic GPU kernels from CUDA SDK 8.0 [2] and Rodinia [1], listed in Table 2. These benchmark applications cover a wide range of execution patterns, such as DRAM intensive, L2 cache intensive, shared memory intensive and computation intensive. They also have different kernel grid and block settings, which implies various GPU workloads and different magnitudes of kernel execution time. With these experimental settings, we have a total of 720 unique experiments. We repeat each experiment for 1,000 times and report the MAPES.

We develop a open-source project that is used to reproduce our experiments. The source code can be found in <https://github.com/blackjack201312/NV-DVFS-Benchmark>. One can use the project to collect the data of kernel performance counters and execution time under different core and memory frequency settings automatically. We also provide codes for performance modeling and error analysis.

Table 2: tested applications

Abbr.	Application Name	Grids/Blocks	Warps
BP	backprop	(1 4,096 1)/(16 16 1)	32,768
BS	BlackScholes	(32,768 1 1)/(128 1 1)	131,072
CG	conjugateGradient	(32,768 1 1)/(4 32 1)	131,072
FWT	fastWalshTransform	(4,096 1 1)/(512 1 1)	65,536
HSP	hotspot	(37 37 1)/(16 16 1)	10,952
Hist	histogram	(240 1 1)/(192 1 1)	1,440
MMG	matrixMul(Global)	(32 32 1)/(32 32 1)	32,768
MMS	matrixMul(Shared)	(32 32 1)/(32 32 1)	32,768
MS	MergeSort	(8,192 1 1)/(128 1 1)	32,768
NN	near neighbor	(168 1 1)/(256 1 1)	1,344
RD	reduction	(65,536 1 1)/(256 1 1)	524,288
SC	scan	(6,656 1 1)/(256 1 1)	53,248
SN	sortingNetworks	(2,048 1 1)/(256 1 1)	16,384
SP	scalarProd	(128 1 1)/(256 1 1)	1,024
SQR	SobolQRNG	(1 1,000 1)/(64 1 1)	2,000
TR	transpose	(512 512 1)/(16 16 1)	2097,152
VA	vector addition	(4,096 1 1)/(256 1 1)	32,768
convS	convolutionSeparable	(24 768 1)/(16 4 1)	36,864
convT	convolutionTexture	(192 128 1)/(16 12 1)	147,456
quasiG	quasirandomGenerator	(128 1 1)/(128 3 1)	1,536

REFERENCES

- [1] Shuai Che et al. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization (IISWC) 2009. IEEE International Symposium on*. IEEE, 44–54.
- [2] NVIDIA. 2016. GPU Computing SDK. [Online] <https://developer.nvidia.com/gpu-computing-sdk>. (2016).
- [3] NVIDIA. 2016. NVIDIA Management Library. [Online] <https://developer.nvidia.com/nvidia-management-library-nvml>. (2016).
- [4] NVIDIA. 2016. NVIDIA Profiler. [Online] <http://docs.nvidia.com/cuda/profiler-users-guide>. (2016).
- [5] Orbm2k. 2016. NVIDIA Inspector. [Online] <http://blog.orbm2k.de/tools/nvidia-inspector-tool>. (2016).