Hierarchically Characterizing CUDA Program Behavior

Zhibin Yu, Hai Jin
Service Computing Technologies and System Lab/
Cluster and Grid Computing Lab,
Huazhong University of Science and Technology
Wuhan, China, 430074

Nilanjan Goswami, Tao Li
Intelligent Design of Efficient Architecture Lab,
University of Florida, Gainesville
Florida, USA

Lizy Kurian John
Laboratory for Computer Architecture,
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX78712 USA

Introduction

Over the last few years, the performance of Graphic Processing Unit (GPUs) has improved more rapidly than that of CPUs [1]. The key to harness the powerful computation power of GPGPUs is an easy and efficient programming model. To this end, NVIDIA created Compute Unified Device Architecture (CUDA) programming mode [2]. It is implemented by extending the standard ANSI C with keywords that designate data-parallel functions called kernels.

CUDA programming mode is very different from sequential programming modes. To characterize CUDA program behavior and understand why and where they can achieve significant speedup comparing to sequential programs, it is important to revisit the basic block level and instruction level properties besides those at the thread level. In this paper, we propose to characterize CUDA program behaviors hierarchically by quantitatively gleaning properties from thread, basic block, and instruction levels.

In addition, previous researchers have demonstrated that basic blocks vectors (BBVs) are one of the most accurate techniques for creating code signatures [3] for sequential programs. In this paper, we firstly employ basic block and basic block vectors to analyze the code signature of CUDA threads. We observed that basic block characteristics of CUDA kernels are very different from those of sequential programs. Based on the basic block vectors, we construct the similarity matrix of threads. We show that the similarity matrix can be a very powerful tool for performance tuning.

Methodology

Metrics
❖Number of instructions per thread
❖Thread performance
❖Number of basic blocks
❖Average basic block size
❖Program footprint
❖Instruction mix
❖Instruction-level parallelism
….

Similarity Matrix
❖Basic block vector per thread
❖Basic block vector per kernel
❖Synchronization vector
❖Similarity matrix based on basic block vectors

Benchmarks
❖CUDA SDK
❖Parboil
❖Rodinia
❖Other programs from recent papers
35 benchmarks in total

Platforms
❖Based on GPGPUsim
❖.Extends cuda-sim to support
- Measure instruction dependency distance
- Generate basic block vectors per thread and for the whole kernel
- Generate synchronization vectors
- Measure instruction mix
- Measure the instruction count per thread

❖Extends gpu-sim to support
- Measure the performance of each CUDA thread

Table 1 Hardware Configuration

Number of Shader Cores	28
Warp size	32
SIMD Pipeline Width	8
Number of Threads/Core	1024
Number of CTAs/Core	8
Number of Registers/Core	16384
Shared Memory /Core (KB)	16(16 banks, 1 access/cycle/bank)
Constant Cache Size / Core	8KB (2-way set assoc, 64 lines)
Texture Cache Size / Core	64KB (2-way set assoc, 64 lines)
Number of Memory Channels	8
L1 Cache	None
L2 Cache	None
Bandwidth Per Memory Module	8 (Bytes/Cycle)
DRAM Request Queue Capacity	32
Memory Controller	Out of order (FR-FCFS)
Branch Divergence Method	Immediate Post Dominator
Warp Schedule Policy	Round Robin among read warps

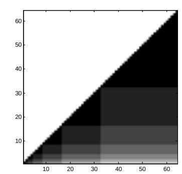
Table 2 Interconnect Configuration

Topology	Mesh
Routing Mechanism	Dimension Order
Routing delay	1
Virtual channels	2
Virtual channel buffers	4
Virtual channel allocator	iSLIP / PIM
Allocation iters	1
Virtual channel allocation delay	1
Input speedup	2
Flit size (Bytes)	16

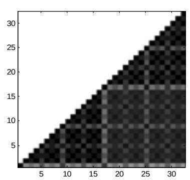
Results

Table 3 Basic Block Properties of CUDA Programs
NBB for x% means Number of Basic Blocks account for x% of program execution
***: The BS is a modified version of BlackScholes**

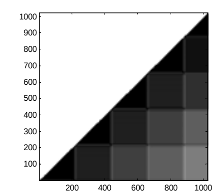
Benchmark	Number of Basic Blocks	NBB for 80%	NBB for 90%	Average Basic Block Size	Average Number of Successor Basic Blocks
64H-k2	11	5	8	5.67	1.45
BFS	8	3	3	7.56	1.625
BlackScholes	4	4	4	26.4	1.5
BN	24	4	6	7.32	1.5
BP-k1	10	4	6	9.63	1.5
BS(*)	4	4	4	26.4	1.5
CL	110	4	5	5.19	1.38
CP	6	5	5	10.43	1.33
CS-k1	7	4	5	33.75	1.29
FWT-k2	13	10	12	10	1.69
GS	7	4	5	12.25	1.43
HS	26	9	14	9.19	1.65
KM-k2	16	7	10	4.76	1.44
LIB-k1	49	11	13	8.72	1.6
LPS	30	12	14	7.61	1.57
LT-k3	13	4	4	7.5	1.55
LV	21	5	5	8	1.48
MC	1	-	-	10.5	1
MM	6	4	5	18.57	1.5
MRIF-k1	15	2	2	12	1.6
MT-k1	5	4	5	9.5	1.4
NE	21	12	14	5.68	1.33
NN	6	4	5	15.14	1.33
NQU	29	6	7	6.17	1.45
NW-k1	17	8	10	15.83	1.41
PF	16	4	4	8.53	1.44
PNS	103	59	100	8.47	1.48
PR-k1	10	7	8	7.36	1.4
RAY	79	-	-	10	1.46
RPES-k1	29	6	8	14.43	1.48
SAD	19	-	-	17.4	1.59
SLA-k1	13	6	9	6.79	1.54
SP	18	8	11	5.68	1.56
SRAD-k1	32	15	17	9.42	1.41
SS-k2	21	9	14	7.43	1.48
ST3D	8	3	4	22.78	1.5
STO	19	12	13	124.95	1.42
TPACF	64	15	24	8	1.55



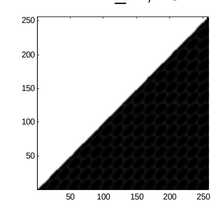
64H_k2,1-64



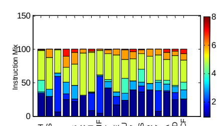
BP_k1,1-32



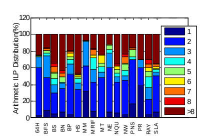
LT_k3,1-1024



NW_k1,1-256



The Instruction Mix of CUDA Benchmarks. The legends are: 1-INT, 2-FP, 3-CS, 4-LS, 5-DMC, 6-CF, 7-PSC, 8-MI



Arithmetic instruction dependency distances of CUDA benchmarks

INT ---- Integer arithmetic
FP ---- Floating point
CS ---- Comparison and selection
LS ---- Logic and shift
DMC ---- Data movement and conversion
CF ---- Control flow
PSC ---- Parallel Synchronization and Communication
MI ---- Miscellaneous

Conclusion

We present a hierarchical methodology to quantitatively characterize CUDA program behavior at thread, basic block and instruction level. We summarize the main findings here. First, the IPC of CUDA thread is only about 1/40~1/100 of the average IPC of CPUs. Second, the average number of basic blocks of CUDA programs is 1/11~1/25 of that of sequential programs. Finally, the data movement and conversion instructions (*mov, cvt*) of CUDA programs hold a high percentage (37.8%). There are also a lot of other findings such as ILP of CUDA kernels in the paper. To our best knowledge, we are the first to do such characterization for CUDA programs. The outcome of our work can be used to optimize GPGPU architectures and CUDA compilers.

The CUDA programming model derives from the more general Single-Program Multiple-Data (SPMD) model which is widely available other parallel processing systems. Therefore, the proposed hierarchical characterization methodology, especially the basic block vectors and similarity matrix, can also be used to characterize other SPMD parallel programs.

Acknowledgements

This work is supported by NSF China under Grant No. 60973036

References

[1] http://www.nvidia.com/
[2] NVIDIA CORPORATION. NVIDIA CUDA Programming Guide, version 3.0.
[3] T. Sherwood, E Perelman, G. Hamerly, and B. Calder, “Automatically Characterizing Large Scale Program Behavior”, Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM Press, October 5-9, 2002, San, Jose, CA, pp. 45-57