An Overview of Theoretical Performance Prediction Models for GPUs

Taabish Jeshani

Agenda

Introduction

Background

A Performance Prediction Model for the CUDA GPGPU Platform

An Adaptive Performance Modeling Tool for GPU Architectures

Input-Aware Auto-Tuning of Compute-Bound HPC Kernels

GPGPU Performance and Power Estimation Using Machine Learning

A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architecture

MWP-CWP

Future Work & Conclusion

Introduction

Problem

How it relates to my research?

Why?

Background

A Performance Prediction Model for the CUDA GPGPU Platform

Combination of the BSP model of Valiant, the PRAM model of Fortune and Wylie, and the QRQW model of Gibbons, Matias, and Ramachandran

The model separates memory accesses and computations and accounts for the cost of both separately.

Also takes into account the effects of scheduling, pipelining, and memory hierarchy on performance

The model can be used to analyze pseudo-code for a CUDA kernel and predict its performance by estimating the number of cycles required for computation and memory accesses

Kothapalli, Kishore & Mukherjee, Rishabh & Rehman, M. & Patidar, Suryakant & Narayanan, P. & Srinathan, Kannan. (2009). A performance prediction model for the CUDA GPGPU platform. 16th International Conference on High Performance Computing, HiPC 2009 - Proceedings. 463-472. 10.1109/HIPC.2009.5433179.

How it works?

The cost of computation is estimated by considering the cycle requirement of each operation and adding them up.

The cost of memory accesses is estimated by considering the deep memory hierarchy in the GPU and the large variation in access time for each level of the hierarchy.

Parameter	Definition
N_w	Number of warps per block
N_t	Number of threads per warp = 32
D	Pipeline depth of a core
N_c	Number of cores per SM
K_i	Kernel i on the GPU
$C_t(K)$	Max. number of cycles required by any thread in kernel K
R	Clock rate of GPU
$T(K)$	Time taken by a kernel K

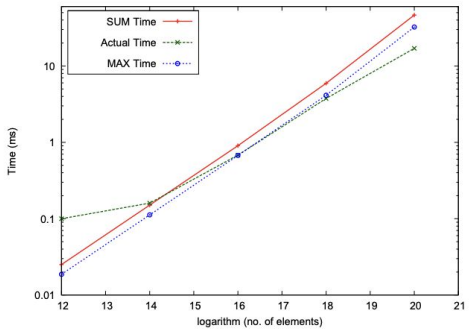
Table 1. List of Parameters in our Model

Kothapalli, Kishore & Mukherjee, Rishabh & Rehman, M. & Patidar, Suryakant & Narayanan, P. & Srinathan, Kannan. (2009). A performance prediction model for the CUDA GPGPU platform. 16th International Conference on High Performance Computing, HiPC 2009 - Proceedings. 463-472. 10.1109/HIPC.2009.5433179.

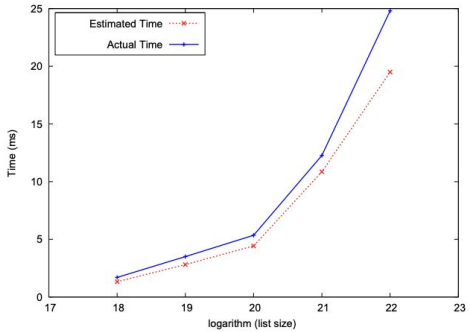
Results

The model assumes that scheduling is preemptive in nature, but the actual scheduling performed inside the GPU is not public knowledge.

The model does not take into account the effect of intra-kernel synchronization steps such as __syncthreads() on overall runtime.



(a) Matrix Muliplication



(b) List Ranking

Kothapalli, Kishore & Mukherjee, Rishabh & Rehman, M. & Patidar, Suryakant & Narayanan, P. & Srinathan, Kannan. (2009). A performance prediction model for the CUDA GPGPU platform. 16th International Conference on High Performance Computing, HiPC 2009 - Proceedings. 463-472. 10.1109/HIPC.2009.5433179.

An Adaptive Performance Modeling Tool for GPU Architectures

The model is designed to provide performance information to an auto-tuning compiler and assist it in narrowing down the search to the more promising implementations.

It can also be incorporated into a tool to help programmers better assess the performance bottlenecks in their code.

The authors analyze each GPU kernel and identify how the kernel exercises major GPU microarchitecture features.

They introduce an abstract interpretation of a GPU kernel, work flow graph, based on which they estimate the execution time of a GPU kernel.

Sara S. Baghsorkhi, Matthieu Delahaye, Sanjay J. Patel, William D. Gropp, and Wen-mei W. Hwu. 2010. An adaptive performance modeling tool for GPU architectures. SIGPLAN Not. 45, 5 (May 2010), 105–114. https://doi.org/10.1145/1837853.1693470

How it works?

The model takes into account factors such as warp-level parallelism (WLP), data-level parallelism (DLP), and instruction-level parallelism (ILP).

The model introduces the concept of a work flow graph (WFG) to represent the computation of a kernel and combine the effects of different performance factors.

The model estimates the execution latency of a kernel by combining the effects of available concurrency (WLP, DLP, ILP) and latencies of the SIMD pipeline and memory system.

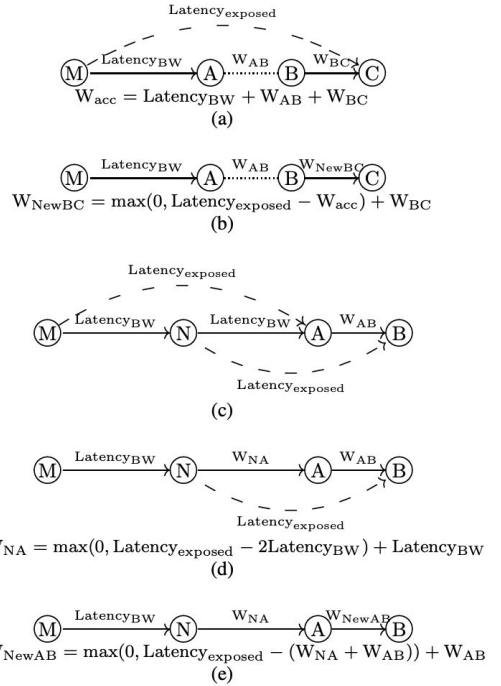


Figure 4. Reducing data-dependence arcs. (a) A sequential WFG subgraph with the long-latency data-dependence arc dashed. (b) Data-dependence arc removed after the untolerated memory latency is incorporated into W_{NewBC}. (c) Two interleaved long latency memory operations. (d) Data-dependence arcs are removed in the order that their corresponding uses appear in WFG. (e) W_{NA} that includes the untolerated memory latency of M is incorporated in covering the memory latency for N; the overlapped latency is counted once toward the warp execution latency.

Sara S. Baghsorkhi, Matthieu Delahaye, Sanjay J. Patel, William D. Gropp, and Wen-mei W. Hwu. 2010. An adaptive performance modeling tool for GPU architectures. SIGPLAN Not. 45, 5 (May 2010), 105–114. https://doi.org/10.1145/1837853.1693470

Results

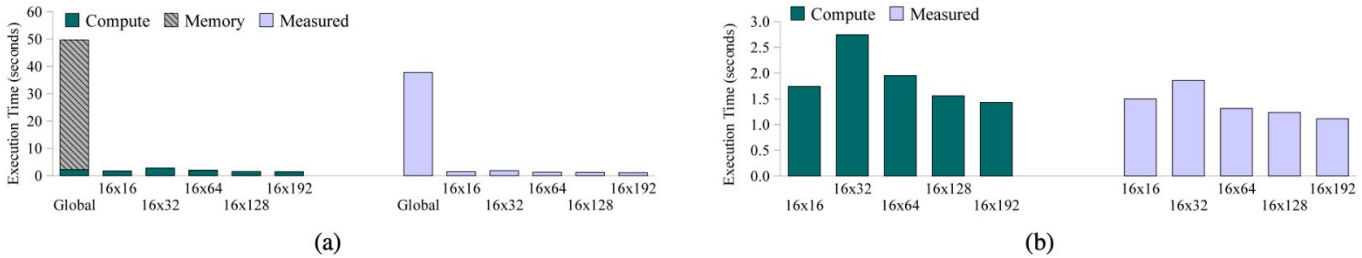


Figure 7. Matrix multiply kernels. (a) Initial and tiled kernels: a breakdown of the predicted time for the global version shows the portion of memory stalls versus compute time. (b) Zoomed for tiled kernels.

Sara S. Baghsorkhi, Matthieu Delahaye, Sanjay J. Patel, William D. Gropp, and Wen-mei W. Hwu. 2010. An adaptive performance modeling tool for GPU architectures. SIGPLAN Not. 45, 5 (May 2010), 105–114. https://doi.org/10.1145/1837853.1693470

Input-Aware Auto-Tuning of Compute-Bound HPC Kernels

The paper presents an input-aware auto-tuning framework for matrix multiplications and convolutions, called ISAAC.

ISAAC uses predictive modeling techniques to drive highly parameterized PTX code templates towards not only hardware-, but also application-specific kernels.

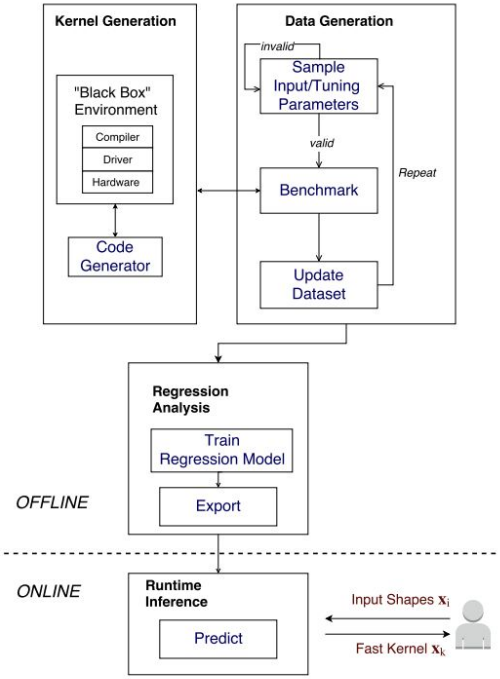


Figure 1: Overview of ISAAC

Philippe Tillet and David Cox. 2017. Input-aware auto-tuning of compute-bound HPC kernels. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17). Association for Computing Machinery, New York, NY, USA, Article 43, 1–12. https://doi.org/10.1145/3126908.3126939

How it works?

The system is composed of four major components:

1. code generation/parameterization techniques for matrix multiplication and convolution
2. a process for generating training data for the input-aware predictive model
3. a multi-layer perceptron used as the input-aware predictive model
4. and a method for using this model at runtime to quickly infer globally optimal kernels given any input configuration.

Philippe Tillet and David Cox. 2017. Input-aware auto-tuning of compute-bound HPC kernels. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17). Association for Computing Machinery, New York, NY, USA, Article 43, 1–12. https://doi.org/10.1145/3126908.3126939

Results

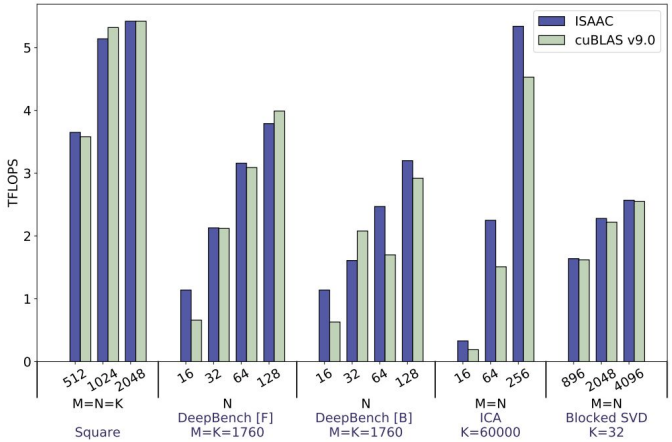


Figure 6: SGEMM performance on the GTX 980 TI

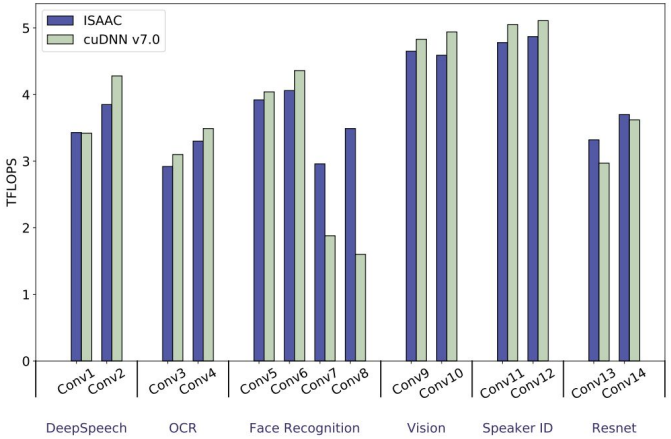


Figure 9: SCONV performance on the GTX 980 TI

Philippe Tillet and David Cox. 2017. Input-aware auto-tuning of compute-bound HPC kernels. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17). Association for Computing Machinery, New York, NY, USA, Article 43, 1–12. https://doi.org/10.1145/3126908.3126939

GPGPU Performance and Power Estimation Using Machine Learning

The paper presents a model for estimating the performance and power of GPUs using machine learning techniques.

The model is trained on measurements from real GPU hardware, allowing it to accurately predict how applications scale as the GPU's configuration is changed.

The model uses hardware performance counter values to predict which scaling curve from the training data best represents a new application, allowing it to estimate the performance and power of the new application at different GPU configurations.

G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena and D. Chiou, "GPGPU performance and power estimation using machine learning," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 2015, pp. 564-576, doi: 10.1109/HPCA.2015.7056063.

How it Works?

The model is trained on a collection of applications that are run at numerous different hardware configurations. From the measured performance and power data, the model learns how applications scale as the GPU's configuration is changed.

The model uses hardware performance counter values to predict which scaling curve from the training data best represents a new application. These dynamic counter values are fed into a neural network that predicts which training kernel is most like this new kernel.

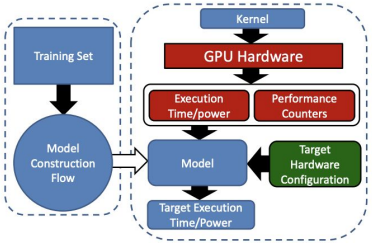


Fig. 2: The model construction and usage flow. Training is done on many configurations, while predictions require measurements from only one.

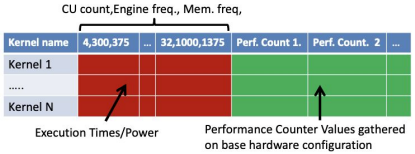


Fig. 3: The model's training set, which contains the performance or power of each training kernel for a range of hardware configurations.

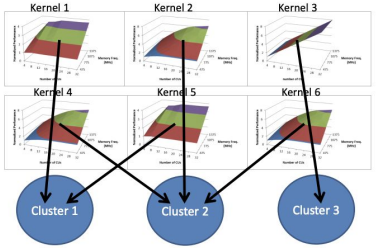


Fig. 4: Forming clusters of kernel scaling behaviors. Kernels that scale in similar ways are clustered together.

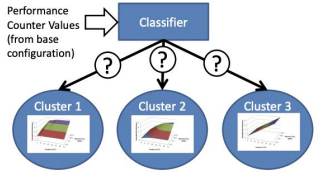
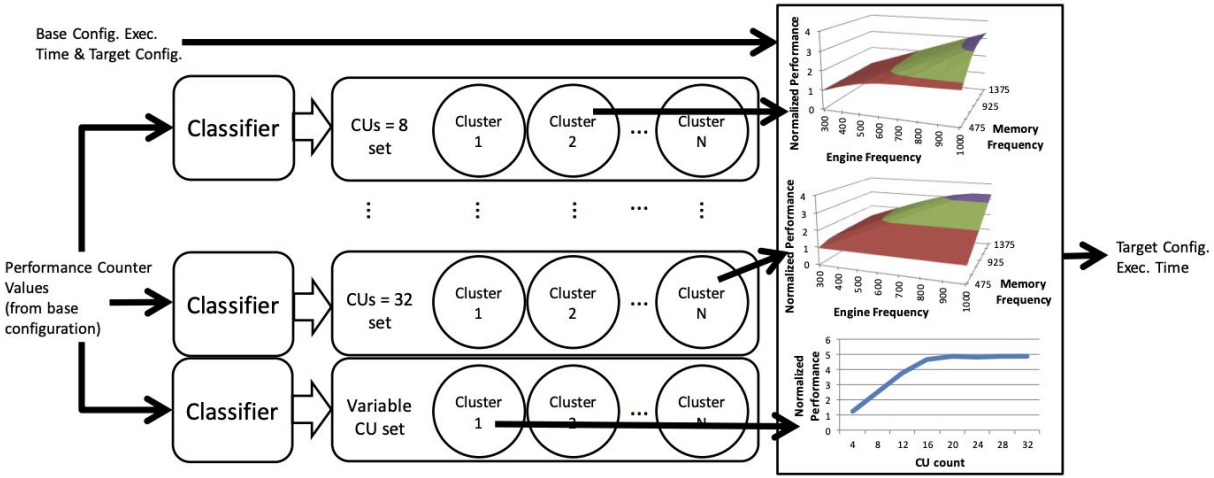


Fig. 5: Building a classifier to map from performance counter values to clusters.

G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena and D. Chiou, "GPGPU performance and power estimation using machine learning," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 2015, pp. 564-576, doi: 10.1109/HPCA.2015.7056063.

How it Works? Cont.



G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena and D. Chiou, "GPGPU performance and power estimation using machine learning," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 2015, pp. 564-576, doi: 10.1109/HPCA.2015.7056063.

Results

Memory Frequency (MHz) \ CU Count	4	8	12	16	20	24	28	32
475	20.4	18.2	20.5	20.7	23.5	25.9	26.5	31.6
625	20.3	15.5	14.4	13.5	16.7	21.1	20.2	21.2
775	24.7	15.6	11.9	13.1	13.3	17.0	17.3	19.4
925	14.5	13.7	11.3	13.5	14.2	12.9	13.4	17.2
1075	13.5	13.7	13.0	12.6	13.5	13.6	13.2	18.3
1225	15.8	16.3	12.2	10.6	9.0	13.5	11.8	14.2
1375	15.5	11.1	12.8	10.8	11.1	11.6	12.7	11.5

Legend
10.0
15.0
20.0
25.0
30.0

Fig. 10: Validation set error heat map at 1000 MHz core frequency. Each point represents the average error of estimating from that point's base configuration to all other configurations (including all other frequencies).

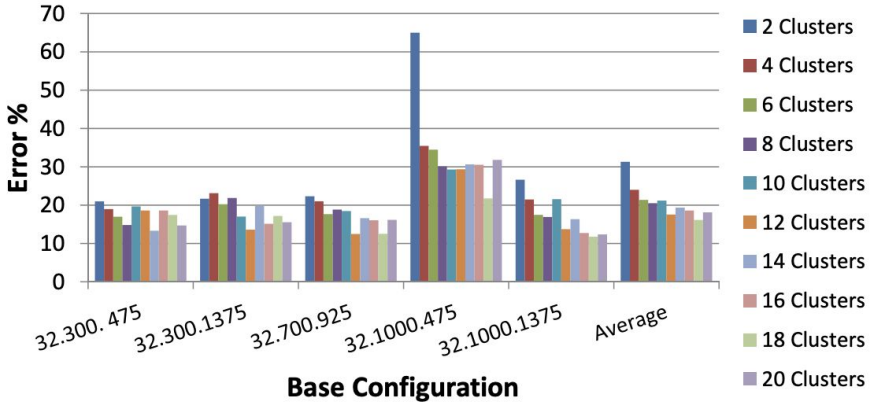


Fig. 13: Validation set error variation across cluster count.

G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena and D. Chiou, "GPGPU performance and power estimation using machine learning," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 2015, pp. 564-576, doi: 10.1109/HPCA.2015.7056063.

A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architecture

The paper proposes a model that combines hardware performance counter data with machine learning and advanced analytics to model power-performance efficiency for modern GPU-based systems.

The resulting model is accurate for predicting power and performance for application kernels on modern GPUs and can identify power-performance bottlenecks and their root causes for various complex computation and memory access patterns.

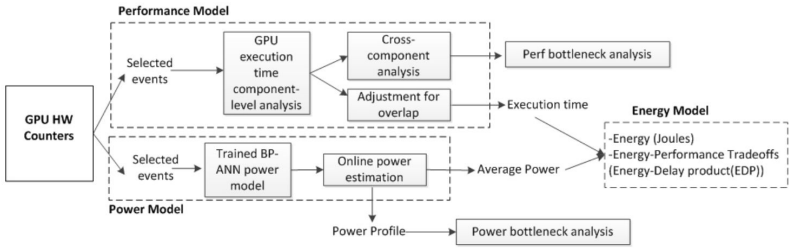


Fig. 3: Detailed view of our GPU performance counter based approach to estimate energy use on a real system.

S. Song, C. Su, B. Rountree and K. W. Cameron, "A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures," 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, Cambridge, MA, USA, 2013, pp. 673-686, doi: 10.1109/IPDPS.2013.73.

How it works?

TABLE II: Performance Events For Training

events	composition	category
G_{load}	gld_request+l1_global_load_hit +l1_global_load_miss	global memory
G_{store}	global_store_transaction	global memory
L_{local}	local_load	local memory
L_{store}	local_store	local memory
$exec\_inst$	inst_executed	instruction pipeline
$branch$	branch	instruction pipeline
$divergent\_branch$	divergent_branch	instruction pipeline
S_{Access}	Shared_load+l1_shared_bank_conflict	shared memory
T_{hits}	tex0_cache_sector_queries	texture memory
T_{miss}	tex0_cache_sector_misses	texture memory

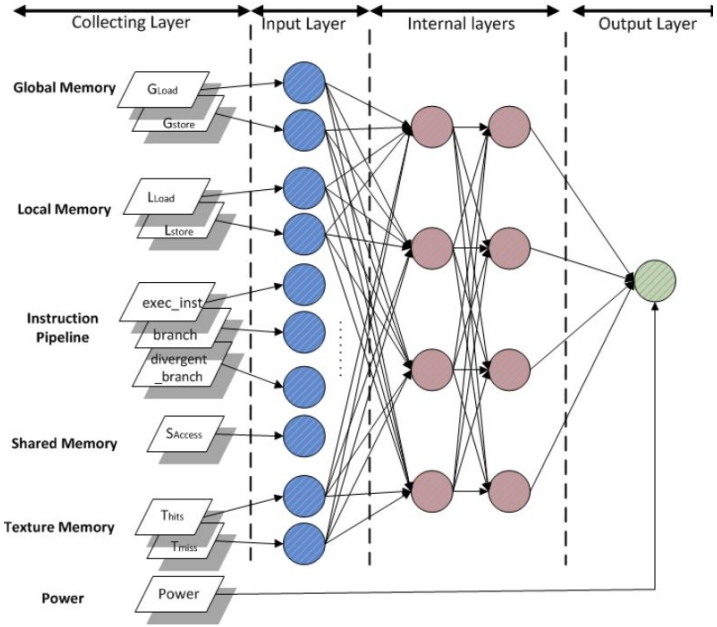


Fig. 6: Design diagram for the BP-ANN based GPU power model. Circles represent neurons.

S. Song, C. Su, B. Rountree and K. W. Cameron, "A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures," 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, Cambridge, MA, USA, 2013, pp. 673-686, doi: 10.1109/IPDPS.2013.73.

Results

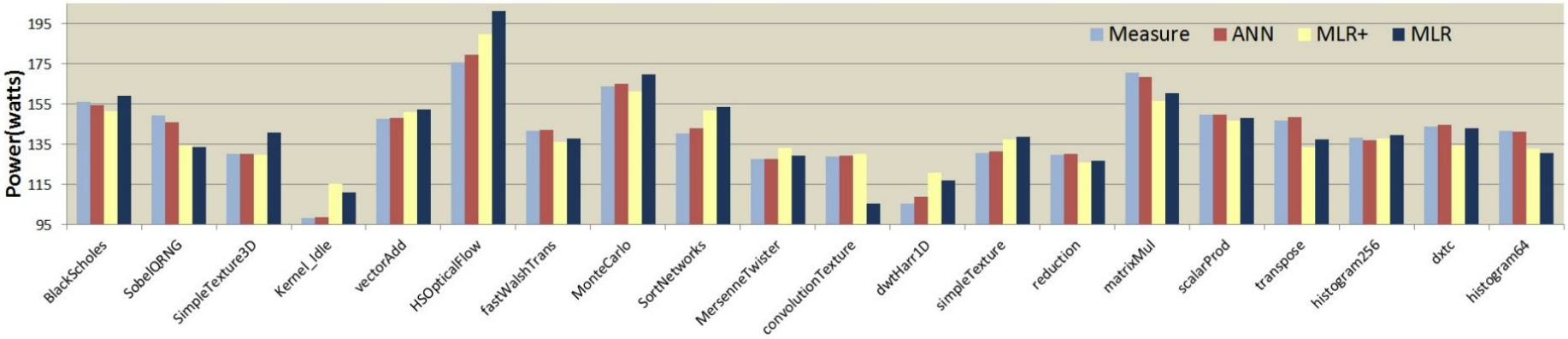


Fig. 9: Prediction accuracy comparisons of BP-ANN, MLR and MLR+ for 20 CUDA kernels.

S. Song, C. Su, B. Rountree and K. W. Cameron, "A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures," 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, Cambridge, MA, USA, 2013, pp. 673-686, doi: 10.1109/IPDPS.2013.73.

An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness

The model takes into account the degree of memory warp parallelism and computation warp parallelism to estimate the cost of memory requests and overall execution time of a program

The model can provide insights into the performance bottlenecks of parallel applications on GPU architectures and help programmers tune their applications for better performance.

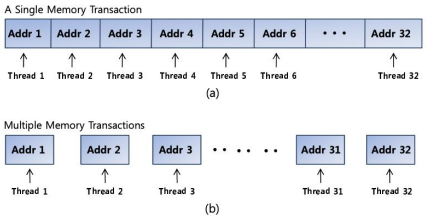


Figure 2: Memory requests from a single warp. (a) coalesced memory access (b) uncoalesced memory access

Sunpyo Hong and Hyesoon Kim. 2009. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In Proceedings of the 36th annual international symposium on Computer architecture (ISCA '09). Association for Computing Machinery, New York, NY, USA, 152–163. https://doi.org/10.1145/1555754.1555775

How it Works?

The model estimates the execution time of massively parallel programs on GPU architectures by taking into account the degree of memory warp parallelism (MWP) and computation warp parallelism (CWP).

MWP represents the maximum number of warps in each SM that can access memory simultaneously during one memory warp waiting period.

CWP represents how much computation can be done by other warps while one warp is waiting for memory values.

The model estimates the cost of memory requests based on MWP and CWP, and uses this information to estimate the overall execution time of a program.

The model takes into account factors such as the number of active warps per SM, memory access patterns (coalesced/uncoalesced), and synchronization effects to accurately estimate execution time.

Sunpyo Hong and Hyesoon Kim. 2009. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In Proceedings of the 36th annual international symposium on Computer architecture (ISCA '09). Association for Computing Machinery, New York, NY, USA, 152–163. https://doi.org/10.1145/1555754.1555775

Results

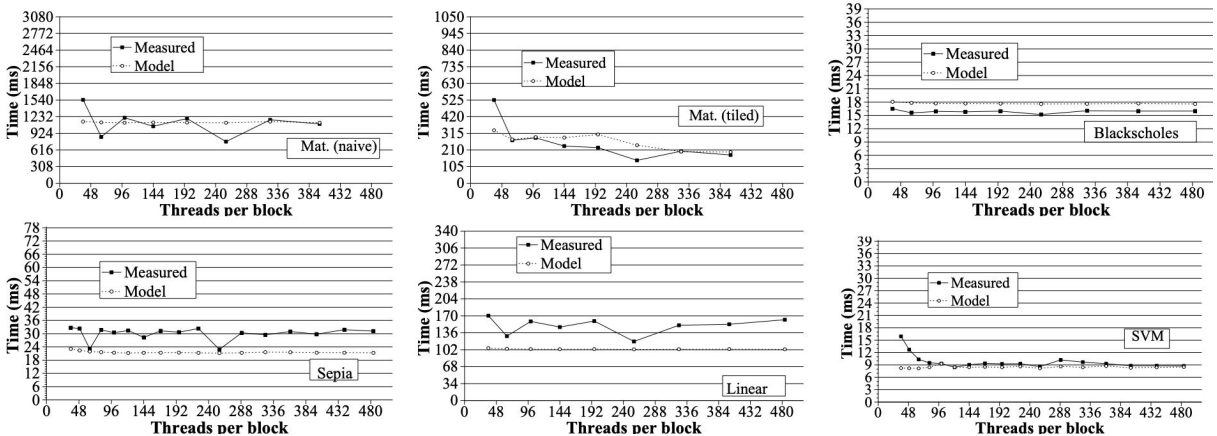


Figure 16: The total execution time of the Merge benchmarks on GTX280

Sunpyo Hong and Hyesoon Kim. 2009. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In Proceedings of the 36th annual international symposium on Computer architecture (ISCA '09). Association for Computing Machinery, New York, NY, USA, 152–163. https://doi.org/10.1145/1555754.1555775

Future Work & Conclusion

Machine learning based Models

More performance metrics

Dynamic adaptations to newer GPU architectures

KLARAPTOR