Full-System GPU Design Space Exploration

Full-System GPU Design Space Exploration

Kuba Kaszyk, Björn Franke, University of Edinburgh

ABSTRACT – The prevalence of machine learning in recent times has had a dramatic impact on the way we design and use mobile and IoT systems. With growing concerns for privacy and quality-of-service, much of the computation that has been traditionally done in the cloud is now moving into edge devices, with far fewer resources available. This in turn increases pressure on the design and development phases, with significant improvements expected in each hardware iteration. Short design cycles require architects to rapidly explore the design space, but by definition, cycle-accurate simulators, which are the primary tool, contradict this requirement. Furthermore, GPUs operate as accelerators linked to a CPU and a tightly coupled software stack, however existing, detailed, GPU simulators are not fast or flexible enough to execute complete multi-kernel applications with heavy CPU-GPU interaction. We argue that for early design space exploration, cycle-accurate simulation is unnecessary, and can be replaced with models exhibiting strong correlation. Instead of implementing a cycle-level model, we propose a multi-phased approach to simulation, with a fast, full-system, functional simulation backed by an offline trace-based approach. By limiting the detail of both the functional and trace based models, we are able to make performance predictions that correlate strongly with real results, and are characterized by near-perfect rank correlation - both valuable metrics for early design space exploration.

THE MODEL – Our full-system approach enables us to model a real System-on-Chip (SoC) platform, with an unmodified, vendor provided software stack. Our trace-based simulator can be used to predict performance for compute kernels executing on a GPU, and we validate our efforts on the ARM Mali-G71 MP8 GPU, integrated in the HiSilicon Hikey-960 platform. However, the approach is retargetable to any GPU and platform combination.

TARGET APPLICATION – We target GPU compute, which includes any and all GPU kernels which make use of the compute pipeline. As the GPU is supported at the hardware level and implicitly supports any native software stack, this enables us to model not only OpenCL, but also higher-level frameworks that build on the hardware interface, or any future software stacks. GPU compute implicitly includes *machine learning* and *computer vision* workloads, which heavily rely on the GPU parallel compute capabilities.

MODELING TECHNIQUES – We take a *functional-first trace-based* approach. Functional execution traces are collected using an our full-system GPU simulator [1]. This GPU simulator executes with a complete software stack, ensuring that the simulation is indistinguishable from executing on a physical hardware platform. We instrument this GPU simulator with trace-generation routines, allowing us to observe memory accesses, non-memory instructions, and barriers in correct order.

The trace processor is a configurable, offline GPU model. The configurable components include architectural features, such as core count, as well as micro-architectural features, such as arithmetic and memory latencies, a configurable cache hierarchy, and functional unit organization. Cache implementation is approximated using a reuse distance model.

We validate our model against the Arm Mali G71 GPU. As the micro-architectural details for this (and most other GPUs) are not publicly available, we *learn* its micro-architectural parameters. In the spirit of [2], we learn and validate the micro-architectural parameters against an available hardware platform. To accelerate the learning, an expert provides an expected range for each parameter, limiting the search space. Afterwards the parameters can be modified for design space exploration.

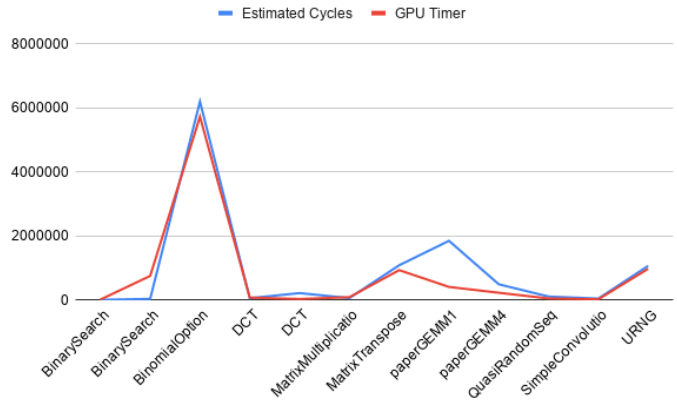

Fig. 1. Our predictions show strong correlation with measure GPU cycle counts.

Mean Abs. Pct. Err.	Correlation	Rank Correlation
44.981197	0.942226	0.820582

TABLE I
ERROR AND CORRELATION RESULTS FROM OUR TRAINED MODEL.

NOVELTY – Trace based simulation is a well-known approach, previously used for GPUs, e.g. [3]. However, existing approaches do not use realistic traces or model the GPU at an intermediate representation, are limited by both speed and framework to single kernels, and attempt, but fall short of achieving cycle-accuracy. In contrast, our full-system driven approach enables trace-based design space exploration across complex real-world applications, where the CPU and GPU continuously interact with each other.

PRELIMINARY RESULTS – We evaluated our system on a set of 62 benchmarks taken from benchmark suites including Polybench, Parboil, Rodinia, the Arm Compute Library, and the AMD APP SDK. Table I shows that across these benchmarks, our system exhibits 44.98% error, however it also shows strong correlation between predicted and actual execution times, with a correlation of 0.94, and a rank correlation of 0.82. Figure 1 provides a visual representation for a individual benchmarks. These correlations are strong enough to be able to confidently identify an optimal hardware configuration or the fastest executing kernel implementation from a selection of experiments. It demonstrates that fast, functional-first, full-system simulation is highly effective for large scale early design space exploration. Table II shows the latencies learned by the machine learning model. These learned latencies align with independent expert knowledge.

Arith.	Store	L1 Hit	L1 Miss	L2 Miss	TLB Hit	TLB Miss
1	1	1	10	200	1	100

TABLE II
MICRO-ARCHITECTURAL LATENCIES LEARNED BY OUR MODEL.

REFERENCES

[1] K. Kaszyk *et al.*, "Full-system simulation of mobile cpu/gpu platforms," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 68–78.

[2] A. Adileh *et al.*, "Racing to hardware-validated simulation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 58–67.

[3] A. Kerr *et al.*, "A characterization and analysis of PTX kernels," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, ser. IISWC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 3–12. [Online]. Available: https://doi.org/10.1109/IISWC.2009.5306801