GPU Mangrove
Execution Time and Power Prediction

Lorenz Braun, Sotirios Nikas, Chen Song, Vincent Heuveline, Holger Fröning

{lorenz.braun, holger.froening}@ziti.uni-heidelberg.de, {sotirios.nikas, vincent.heuveline}@uni-heidelberg.de, chen.song@iwr.uni-heidelberg.de

Ruprecht-Karls University of Heidelberg, Germany

Motivation

Predicting compute kernel execution behavior on GPUs is a difficult task. Scheduling GPU kernels can profit tremendously from modelling performance and power, improving the computation of workloads. Previous work has taken on this task using analytical models and machine learning. While good results were achieved, it is not known how well these approaches work for different GPU architectures (lack of portability). Also, to our knowledge, there are no public models for performance and power predictions (lack of availability).

Hypothesis: *well-structured, locality-optimized and latency-tolerant GPU kernels should, in average, behave agnostic of hardware-related dynamic effects in terms of time and power consumption and depend much more on properties of the code that is being executed.*

With GPU Mangrove we propose a model for execution time and power prediction which is simple, portable and fast.

Portable Code Features

To achieve portability, the code features must not depend on the GPU used, leaving a choice of possible features which are mainly covered by instruction counter and kernel launch configuration (e.g. CUDA grid size).

Features collected on one GPU can be used as model input for prediction on another GPU. This combined with compile time analysis coupled with a runtime system for prediction would allow to compute the code features before kernel execution and without using samples produced on other GPUs.

Profiling Info.	Input Feature
PTX instructions	total instructions
	global memory vol
	shared memory vol
	special operations
	logic operations
	sync operations
kernel launch configuration	threads per CTA
	number of CTAs
	dynamic shared memory

Table 1: CUDA Flux Feature Overview

We use CUDA Flux [2] for instrumenting and executing the GPU kernels and gather the code features. Table 1 gives a rough outline which information is gathered with CUDA Flux.

Prediction Model

Data for our prediction model is generated using CUDA Flux with four major benchmark suites: Rodinia[3], Parboil[4], SHOC[5] and Polybench-GPU[6]. From these four benchmark suites 189 kernels could be used for execution time prediction and 169 for power prediction. Profiling and measurement of kernel execution time and power are done in separate steps (Figure 1). For a given set of benchmarks the profiling step is only performed once and the measurement of time and power is done for each individual GPU.

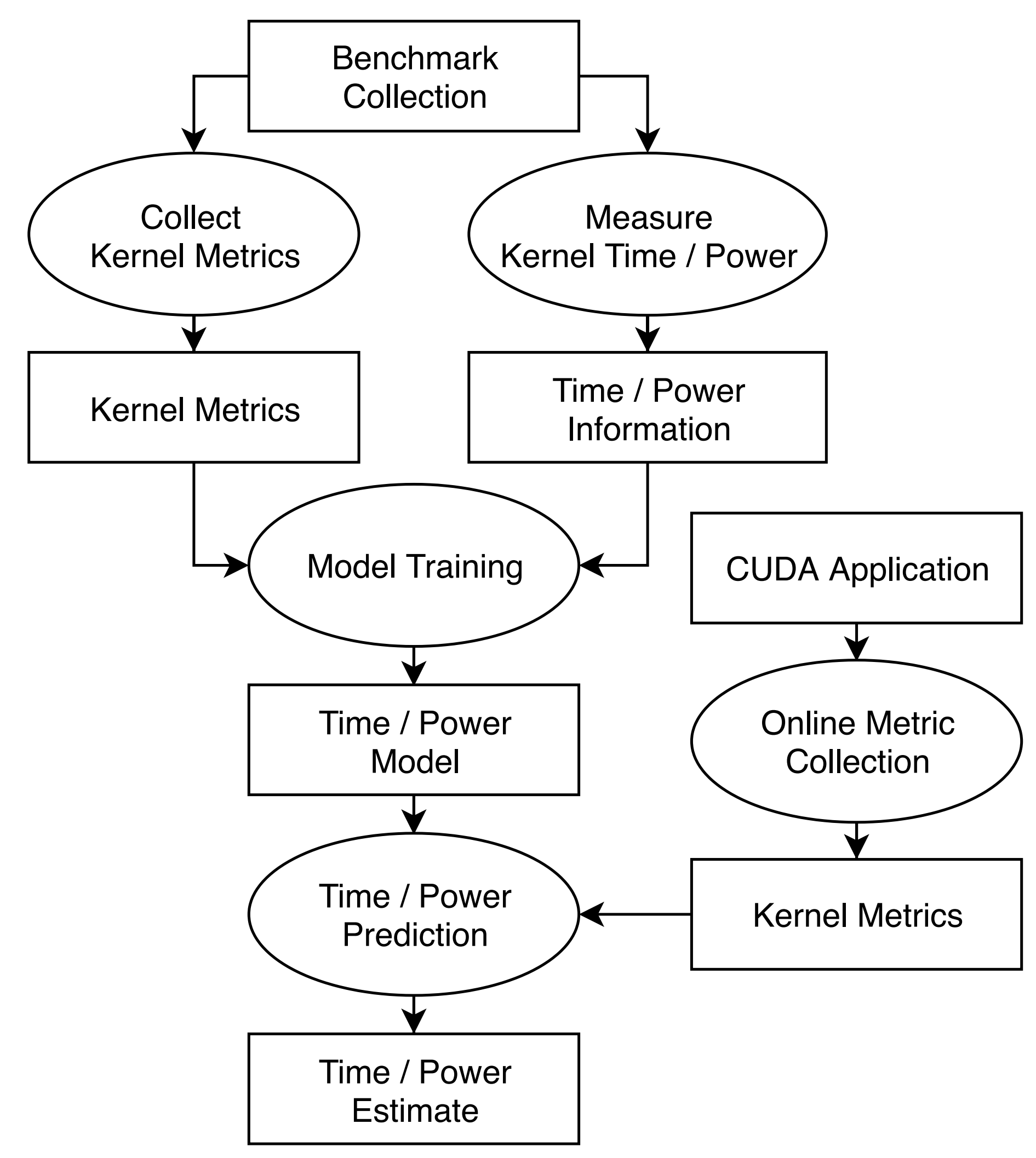

Figure 1: Workflow for execution time and power prediction

The data is used to construct a model for each GPU. We used the Extremely Randomized Trees Regression Algorithm to train our model due to rather fast training time and the limited amount of samples at hand. Nested cross-validation was used to find reasonably good hyperparameters and to ensure good generalization. As the range of time measurements was very large we choose to train the models on the mean absolute percentage error (MAPE).

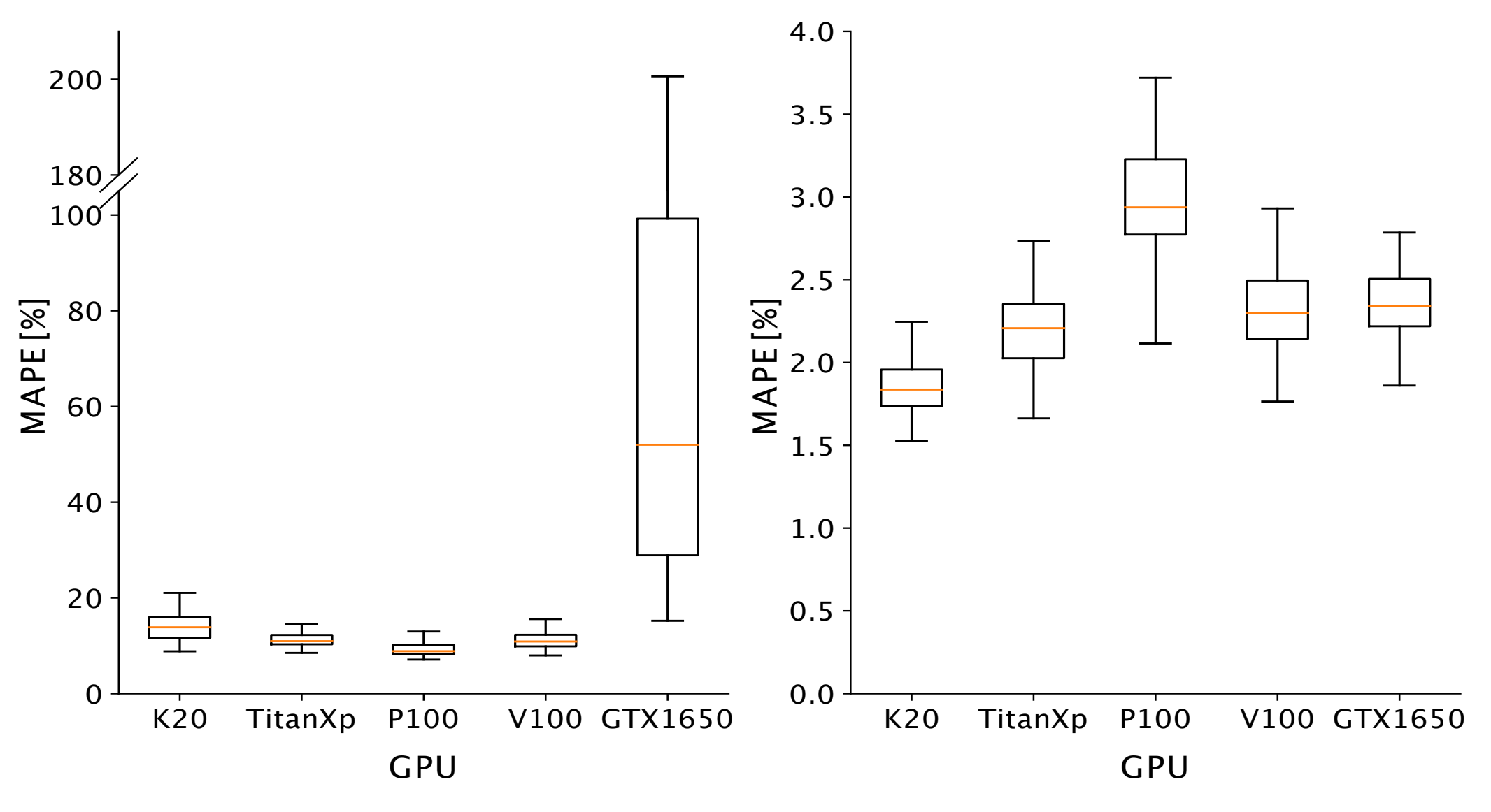

Figure 2: GPU Mangrove prediction results on execution time (left) and power (right) on five GPUs

Results

The results of nested cross-validation across all five GPUs of time and power prediction are summarized in Figure 2. MAPE scores for all iterations with random splits of nested cross-validation with median, first and third quartile are shown. The prediction latency of the unoptimized models varies between 15 and 110 ms.

Using a random forest algorithm allows to evaluate the relative importance of each feature. Figure 3 shows the importances for a NVIDIA K20 GPU. The most important features are mostly in line with our expectations. Surprisingly, parameter memory volume seems to be important for both models. This could indicate that computationally complex kernels have more parameters.

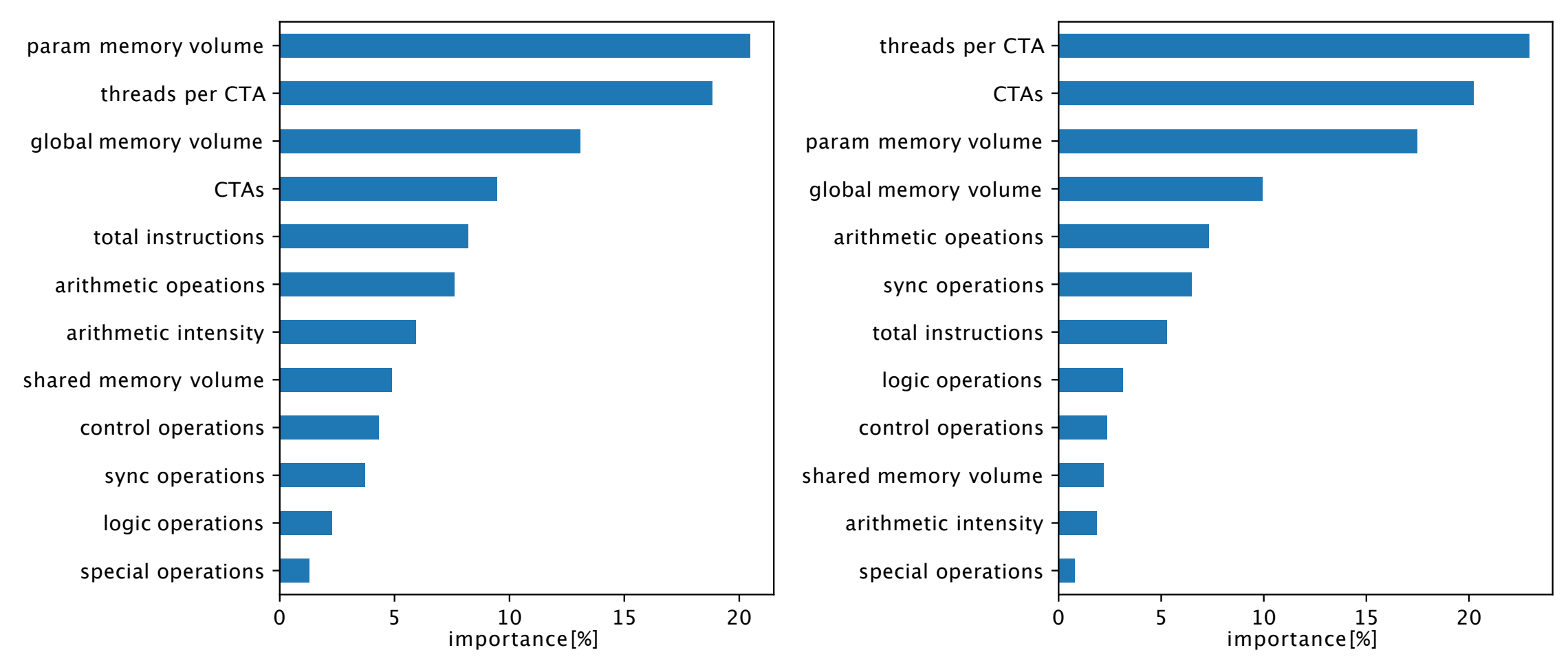

Figure 3: Feature Importance of execution time model (left) and power model (right) for a NVIDIA K20 GPU

Summary

We conclude that in the scope of our work the results support our hypothesis; fast prediction of execution time and power consumption based solely on hardware-independent features is feasible.

However, there is still room for improvement: some effects like caching between kernel launches, bank conflicts and branch divergence are not covered by our features. Also, a larger training dataset including more samples, specifically longer and more computational intensive kernels, would be helpful to improve the prediction accuracy.

More details here:

- https://arxiv.org/abs/2001.07104
- https://github.com/UniHD-CEG/gpu-mangrove

References

[1] Braun, Lorenz, et al. "A Simple Model for Portable and Fast Prediction of Execution Time and Power Consumption of GPU Kernels." arXiv preprint arXiv:2001.07104 (2020).

[2] Braun, Lorenz, and Holger Fröning. "CUDA Flux: A Lightweight Instruction Profiler for CUDA Applications." PMBS Workshop, collocated with International Conference for High Performance Computing, Networking, Storage and Analysis (SC2019). 2019.

[3] Che, Shuai, et al. "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads." IEEE International Symposium on Workload Characterization (IISWC'10). IEEE, 2010.

[4] Stratton, John A., et al. "Parboil: A revised benchmark suite for scientific and commercial throughput computing." Center for Reliable and High-Performance Computing 127 (2012).

[5] Danalis, Anthony, et al. "The scalable heterogeneous computing (SHOC) benchmark suite." Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. 2010.

[6] Grauer-Gray, Scott, et al. "Auto-tuning a high-level language targeted to GPU codes." 2012 Innovative Parallel Computing (InPar). Ieee, 2012