Title

Hardware-Assisted Cross-Generation Prediction of GPUs Under Design

Journal

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(6)

ISSN

0278-0070

Authors

O'Neal, Kenneth
Brisk, Philip
Shriver, Emily
et al.

Publication Date

2019-06-01

DOI

10.1109/tcad.2018.2834398

Peer reviewed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Hardware-Assisted Cross-Generation Prediction of GPUs Under Design

Kenneth O'Neal, Philip Brisk, *University of California Riverside*
|Emily Shriver, Michael Kishinevsky, *Intel Corporation*

***Abstract*— This paper introduces a predictive modeling framework for GPU performance. The key innovation underlying this approach is that performance statistics collected from representative workloads running on current generation GPUs can effectively predict the performance of next-generation GPUs. This is useful when simulators are available for the next-generation device, but simulation times are exorbitant, rendering early design space exploration of microarchitectural parameters and other features infeasible. When predicting performance across three Intel GPU generations (Haswell, Broadwell, Skylake), our models achieved impressively low out-of-sample-errors ranging from 7.45 % to 8.91 %, while running 29,481 to 44,214 times faster than cycle-accurate simulations. A detailed ranking of the most impactful features selected for these models provides an insight as to which microarchitectural subsystems have the greatest impact on performance from one generation to the next.**

***Index Terms*— graphics processors, hardware architecture, machine learning, modeling techniques, performance of systems.**

I. Introduction

CYCLE-ACCURATE simulation times for highly-threaded processors, such as GPUs, are rapidly becoming untenable. The situation is exacerbated in industry, where simulators serve a dual-purpose of performance simulation and RTL performance validation. Consequently, industrial simulators offer greater detail and higher accuracy at the cost of longer runtimes compared to their academic counterparts. Ever-increasing simulation times are prohibitive for early-stage GPU design space exploration when a considerable number of perturbations to the present design must be considered.

Hardware-assisted predictive statistical modeling can help to overcome this conundrum when designing subsequent GPU devices in a larger family. What is needed is a commercially available GPU, representing a current- or past-generation member of the family, a simulator representing a future-generation GPU family member under development, and a set of representative rendering workloads.

This paper extends *Hardware-Assisted Light Weight Performance Estimation (HALWPE) [1]*, a methodology that uses fabricated silicon (host) GPUs to predict the performance of future GPUs under development (target). Our experiments focus on GPUs and treat the 7.5th generation integrated HD4600 GPU of the Intel Core i7 4790 processor as a current-generation GPU (fabricated silicon) and predict the performance of three future-generation GPUs: two 8th generation Broadwell GPUs and one 9th generation Skylake GPU. The prediction is performed by configuring a cycle-accurate simulator (*CAS*) to model the newer generation GPUs, executing a set of 3D render workloads on both the host GPU, to collect performance counter and software metrics, and the CAS to collect target performance, cycles-per-frame— CPF, simultaneously. We then train an ensemble of regression models to predict the CPF of each frame using the host metrics; the models are then applied to new workloads. HALWPE has several contributions:

- HALWPE's novelty is accurately predicting GPU performance across three device generations, spanning micro-architectural, software, parallelism and process improvements. HALWPE achieves 7.45 %, 7.47 % and 8.91 % average *out-of-sample-error* respectively.
- HALWPE uses hardware assistance to run ~30,000-45,000x faster than a cycle-accurate GPU simulator.
- HALWPE predicts performance impacts from changes to vendor-provided drivers and APIs (Fig. 1) on current and future generations of GPU.
- HALWPE predicts performance impacts caused by increasing available GPU parallelism on the current and future generations of GPU.
- HALWPE ranks features to improve model inference and guide designers toward prime microarchitectural and software candidates that warrant additional study.

The remaining text is organized as follows: Section 2 details the modeling framework, model building and application, and workload execution. Section 3 describes the Intel GPU generations modeled and their relative differences. Section 4 details the regression model ensemble employed by HALWPE. Section 5 details simulation-based model results when modifying the driver and increasing available device parallelism. Section 6 details hardware-assisted, cross-generational model results. Section 7 presents feature ranking and discussion, and finally Sections 8 and 9 present related works and our concluding remarks, respectively.

Manuscript received April 28, 2017; revised July 24, 2017 and Sept 20, 2017; accepted Oct 12, 2017. Date of publication Oct 25, 2017. This work was supported in part by the National Science Foundation Award #1528181.

Kenneth O'Neal and Philip Brisk are with the University of California Riverside, Department of Computer Science and Engineering, Riverside, CA. (e-mail: konea001@ucr.edu, philip@cs.ucr.edu).

Emily Shriver and Michael Kishinevsky are with the Strategic CAD Lab of Intel Corporation, Hillsboro, Oregon (e-mail: emily.shriver@intel.com, michael.kishinevsky@intel.com).

Color versions of one or more of the figures in this paper are available at http://ieeexplore.ieee.org.

Digital Object Identifier XX.XXXX/TCAD.2017.XXXXXX

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

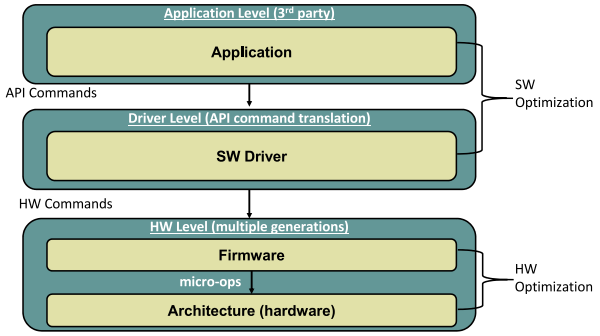


Fig. 1. GPU performance depends on the application, driver/API commands, and architecture.

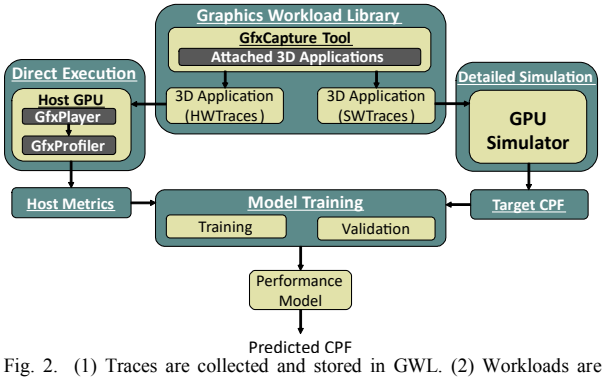


Fig. 2. (1) Traces are collected and stored in GWL. (2) Workloads are executed on the current-generation GPU, and next-generation simulator. (3) Performance counter readings from the host are used to train a model to predict the simulated performance (CPF) of the next-generation GPU.

II. Modeling Framework

We assume that at least one current-generation GPU is available in silicon, and that a high-accuracy next-generation GPU simulator is available, along with representative workloads. Fig. 2 illustrates the HALWPE model development flow. The *Graphics Workload Library (GWL)* refers to our collection of benchmarks, listed in Table 1.

The GWL contains rendering frames from 43 *DirectX* games and GPU benchmarking tools spanning the version 9, 10, and 11 APIs. We collected multiple frames per application and treat each as one workload. The one-frame-per-workload constraint is imposed by the GPU simulator's execution overhead, but longer traces can be executed as well. The GPU Simulator models the GPU microarchitecture, memory subsystems, and a representation of DRAM, all validated internally when configured to model post-silicon GPUs. No explicit memory model beyond the memory subsystem and DRAM already present on the integrated GPU, leading to few related counters.

The GWL applications are assembled into a single training set; we apply 10-fold cross validation to estimate out of sample error. We use three proprietary trace tools to collect single-frame traces in two formats (*GfxCapture*), replay isolated traces (*GfxPlayer*), and collect performance counters (*GfxProfiler*) [2], hardware queries [3], and DirectX program metrics [4-11] on the Haswell host GPU. The trace formats are *HWTraces* (DirectX commands) executed on our Haswell host GPU [2], and *SWTraces* (native GPU commands) executed on our GPU simulator. To reduce profiling overhead, we collect performance counters that can be read in one pass as detailed in the table on page 13 of Ref. 1. Table 2 summarizes the software tools required to implement the HALWPE framework.

TABLE I
GWL: 36 DirectX Applications, 364 rendered frames

Workload	Frame Count	DX Version
Diablo 3	2	9
Dragon Age: Origins	11	9
Dragon Age 2	10	9
Left For Dead 2	10	9
Portal 2	12	9
Skyrim	9	9
Starcraft 2	1	9
StarWars The Old Republic	10	9
Witcher2	10	9
3D Mark Vantage	30	10
Hawx	5	10
Cut The Rope	3	10
Fishie	2	10
Resident Evil 5	10	10
Winsat Alpha Blend	3	10
Winsat ALU Perf	3	10
Winsat Batch Perf	3	10
Winsat Constant Buffer	3	10
Winsat Geometry Perf1	3	10
Winsat Geometry Perf2	3	10
Winsat Texture Load Perf	3	10
3DMark 11 entry	19	11
3DMark 11 perf	20	11
3DMark Cloud Gate	9	11
3DMark Ice Storm	9	11
Assassins Creed 3	5	11
AVP: Evolution	7	11
Batman Arkham City	8	11
Battlefield 3	5	11
Bioshock Infinite	7	11
Crisis 3	10	11
Dirt2	10	11
F1 2012	15	11
Farcry 3	10	11
Heaven	10	11
Lost Planet 2	26	11
MaxPayne 3	5	11
Metro Last Light	10	11
Saints Row 3	8	11
Saints Row 4	7	11
Sleeping Dogs	6	11
Stone Giant	11	11
Tomb Raider	1	11

TABLE II
Software Tools and Libraries used

Tool	Acronym	Purpose
Graphics Workload Library	GWL	Collect frame traces
GPU Simulator	-	Collect golden reference CPF
Graphics Capture Tool	GfxCapture	GPU frame trace capture tool
GPU Hardware Trace	HWTrace	Capture traces executed on the host GPU
GPU Simulator Trace	SWTrace	Capture traces executed on the GPU simulator
HWTrace Replay Tool	GfxPlayer	Replay HWTraces on the host GPU
HWTrace Profiling Tool	GfxProfiler	Collect model features from execution on the host GPU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE III
GPU DEVICE LEGEND. FURTHER DETAILS FOUND IN REFS: [12-14]

GPU Product	Available As	# Slices	# EUs	Driver Gen.	# Executable Traces
Haswell GT2	Hardware	1	20	1	300
Broadwell GT2	Simulator	1	24	2	282
Broadwell GT3	Simulator	2	48	3	364
Skylake GT3	Simulator	2	48	3	364

The predictive models are programmed using R and other commercially available tools to estimate GPU *cycles per frame (CPF)*. Power per frame and other metrics can also be predicted given the target simulator provides a reference value, though feature rankings and selection may change. Model training time is not included in the runtime comparison of HALWPE to cycle-accurate simulation, as training time is amortized over repeated model usage. In practice, models are trained on one set of workloads, and deployed on a disjoint second set of workloads. Once a model has been trained it can be applied to any 3D rendering workload of any length. However, validation of that prediction is limited to workload lengths that can be reasonably executed on the CAS. Fig. 3 illustrates the model training and deployment (prediction) workflows.

III. INTEL GPU ARCHITECTURES

HALPWE is validated using three generations of Intel integrated GPUs (see Table 3). The host desktop PC has a 4-core, 8-thread Intel Core i7-4790k, 16 GB of DDR3 @ 1666 MHz, an Intel HD 4600 Haswell GT2 GPU running at 1155 MHz, and a 2TB 7200 RPM hard disk. The Broadwell GT2, Broadwell GT3 and Skylake GT3 are later versions of this GPU for which simulators are available. We include the performance differences between each generation of GPU to highlight that HALWPE's model suite can accurately generate cross-generation performance estimates, even when the relative performance difference of the two generations is large. In principle this method can be used on any GPU supporting the DirectX API to produce the same metrics, and the devices hardware counters. Further, GPUs leveraging other APIs, such as OpenGL can produce a set of metrics like the DirectX.

To create hardware-assisted model scenarios, we use simulator configurations that execute a driver reflective of the GPU generation: version 1 (Haswell GT2), version 2 (Broadwell GT2), and version 3 (Broadwell GT3, which we also use for Skylake GT3). In some situations, compatibility issues between the architecture and driver caused trace execution to fail on the GPU host and simulator. In Table 3, the Haswell GPU host can execute 300 of the available traces, while simulators for the Broadwell GT2, GT3 and Skylake GT3 can execute 282, 364, and 364 traces respectively.

While Skylake and Broadwell GPUs are commercially available, we validate accuracy using only CAS. Our goal is to mimic the GPU design process while employing commercially to maximal data can be disclosed publicly, e.g. model features. This ensure validated simulator configurations exist, while avoiding implementation work to target in-flight designs. For any host-target prediction scenario, the number of traces that we use to build and evaluate the model is the minimum number that both host and target have the capability to execute.

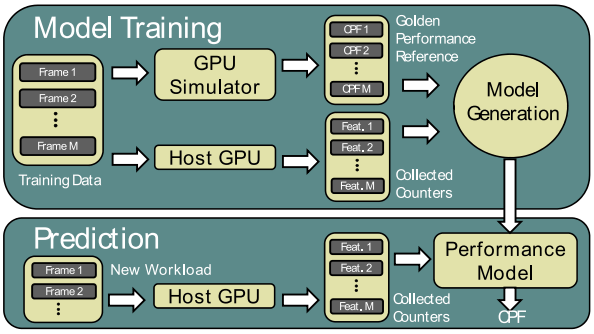


Fig. 3. (Top) Model training and usage (Bottom). Performance counters and DirectX program metrics collected from host GPU execution are used to train a model to a GPU simulator configuration's performance. Performance estimation of new frames uses host metrics as input to the performance model, which predicts the CPF that the GPU simulator would report for the same application, without further simulator execution.

A. GPU Generational Architecture Differences

HALWPE's novelty is accurately predicting GPU performance across multiple device generations spanning micro-architectural, software, parallelism and process improvements. Intel's GPU render pipeline is organized into two logical groups: (1) the Unslice and (2) the Slice. The *Slice* count of a GPU is a measure of available parallelism and contains three sub-elements. The slice common holds fixed function caches, global slice units, the sub-slice, and L3 cache. Sub-slices are organized into parallel groups each containing *Execution Unit (EU)* clusters and their supporting thread dispatch units, samplers, instruction cache, and peripherals. Section 7 provides relative performance comparisons between the host and target GPU representations. [12-14].

We utilize three generations of Intel integrated GPU device; a Haswell GT2 single-slice host (previous generation hardware), with 20 EUs per slice, two variants of Broadwell (single slice GT2, and dual-slice GT3), with 24 EUs per slice and a dual-slice Skylake GT3 containing 24 EUs per slice. The following section omits detailing individual units and their purpose, instead focusing only on the differences between generations. Readers interested in a more detailed description of the architectures should consult [12-14].

1) Haswell vs Broadwell Architecture

Broadwell generation GPUs optimize the microarchitecture. Below we highlight key areas where the Broadwell device has improved over the Haswell implementation.

Unslice: The CPU and GPU communication unit, the GT Interface (GTI) to lower level cache (LLC) bandwidth has improved, allowing 64-bit read and write rather than 32-bit as in Haswell. The Fixed Function (FF) render pipeline has also been optimized on a per-unit basis, resulting in improved pixel back end fill rate and improved Z/Hi-Z test performance.

Slice: Most notably, Broadwell has doubled 32-bit integer computational throughput, and has added 16-bit floating point support. An increase in computational throughput derives from more efficient global resource sharing (L3 cache) amongst slices, changing the total number of EUs per slice and changing the number of sub-slices.

Slice Common: The L1 cache has an increased in overall size by increasing allocation per slice, and the L3 cache has increased 33.33 % from 385 Kbytes to 576 Kbytes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

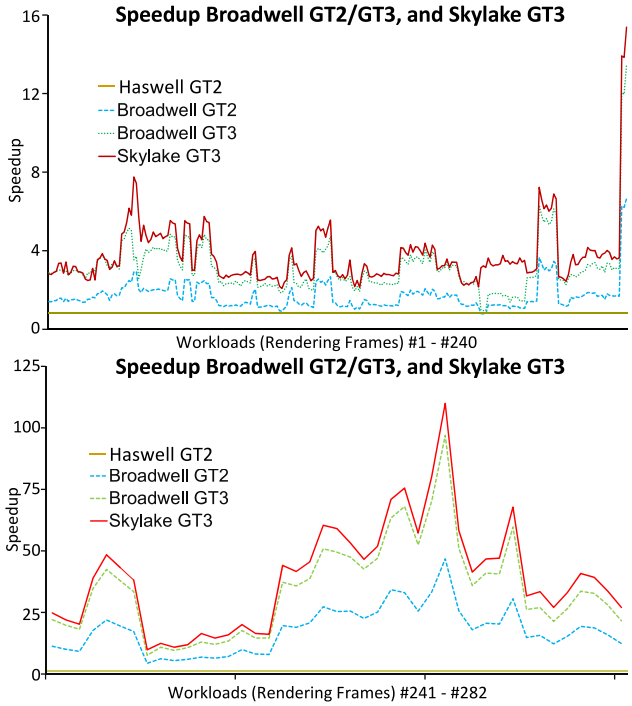


Fig. 4. The speedup computed from CPF obtained on the Broadwell GT2/GT3 and Skylake GT3 (normalized to the CPF attained by Haswell GT2) for 282 rendering frames. Frames 1-240 (top) and frames 241-282 (bottom) are shown on two separate graphs due to the stark difference in CPF ranges.

Sub-slice: By increasing the number of sub-slices to 3 and allocating 8 EUs per sub-slice in Broadwell rather than maintaining two subs-slices with 10 EUs each, two sources of additional throughput were added. First, the total number of EUs increased, and sampler contention has decreased. In total Broadwell contains 120 % more EUs, and 150 % more sampler throughput than a Haswell counterpart containing the same number of slices.

Comparing the performance of the Haswell and Broadwell generation devices we measure a median performance difference of 66.43 %, an average of 317.74 % and a maximum improvement >1000 %. The comparisons are between an actual GPU, the host Haswell generation device, and a near cycle accurate simulator, our Broadwell generation target, the same GPUs used in Scenarios$_{4\text{-}5}$.

2) Broadwell vs Skylake Architecture

The recently released Intel Skylake generation GPUs make significant architectural improvements to the Broadwell GPU architecture.

Unslice: At the platform level, the GTI latency has been reduced, and the command issue ring buffer has also been improved. Further, DDR speed has increased from 1868 MT/s to 2133 MT/s. In total platform compute has improved 50 % from 768 GFLOPS to 1152 GFLOPS. Notable improvements have also been made to the Fixed Function render pipeline, including to the Vertex Shader, the Geometry Shader, the Hull Shader, and the Domain Shader. Additional geometry features have also been added such as Auto Strip detection and their employment in the Tessellation stages of the FF pipeline. This serves to improve both bandwidth and cull rates by reducing the number of redundant computations performed.

Slice: pixel back end fill rate has been further increased between 33 % and 100 %, workload dependent. In addition, a new Multi-Sampling anti-aliasing (MSAA) mode has been added, allowing for 16x MSAA, along with performance improvement in the existing 2, 4 and 8x MSAA modes.

Slice Common: Cache has been increased to 768 Kbytes, an additional 25 %, a total of 200 % over the Broadwell size cache. In addition to the increased memory, memory management is optimized by performing render target compression, compressing memory before send to increase bandwidth at each cache line. This results in 11 % increased cache line bandwidth.

Sub-slice: has been improved by adding explicit 16-bit and 32-bit floating point support. EU/Sampler throughput and Z/Stencil and Pixel operation speed has increased 200 % by performing individual pixel hashes on different slices. Shared virtual memory and cache coherency have also been improved, resulting in better 3D computation. Atomic operations for 32-bit floats, min, max, compare and exchange have also been added. Thread dispatching has been further improved by allowing smaller thread groups, providing finer granularity pre-emption to increase 3D compute responsiveness. These architectural changes result in an additional increase of 24 % performance on average, and 14.3 % median when comparing the Broadwell and Skylake generations.

Fig. 4 reports the speedup of the three target GPU architectures we predict (Broadwell GT2/GT3, and Skylake GT3) normalized to the CPF attained by the Haswell GT2 host for 280 frames. The CPF difference between Skylake GT3 and Haswell GT2 varies from 3x to 112x. This large variation in CPF speedup (and at times a decrease in the Broadwell GT2/GT3 cases) as compared to the Haswell GT2 baseline cannot be captured by a constant multiplier to the baseline performance; more complex predictive models are required.

IV. Regression Models

HALWPE includes twelve linear and one non-linear regression models, which are presented in Fig. 5 We produce 10 least-squares variants; two the standard OLS and NNLS approach, the remaining 8 are the combinations of employing 4 feature selection variants with each. Feature selection is performed using a combination of forward stepwise selection, backward stepwise selection and evaluation of the selected features using the Akaike Information Criterion (AIC) [15] and the Bayesian Information Criterion (BIC) [16]. We also leverage the Lasso regularization model, a non-negative Lasso and the non-linear Random Forest Model. We choose the model that yields the smallest out-of-sample error (E_{out}) as the most accurate.

The choice to use multiple models is driven by the fact that the best model is data dependent, and that each scenario exhibits different relationships between host features and target CPF. The OLS and NNLS models are useful when the relationship is linear, and features are non-correlated; using the AIC and BIC criteria to remove features can simplify the model and help to avoid overfitting.

Linear regularization, shown in the middle, selects features during model construction, by moving their coefficient closer to zero, helping reduce variance and noise on the prediction curve and can improve model accuracy. Random Forest models

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

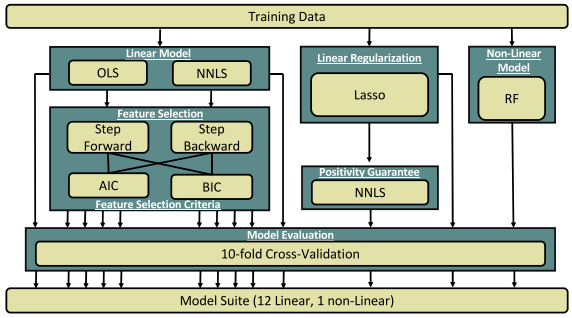


Fig. 5. The generated suite of 13 regression models.

non-linear behavior, prevalent as the generation gap between host and target grows. A larger gap may necessitate using new models such as Neural Networks to maintain accuracy.

A. Linear Regression Overview

Let M be the number of workloads and $X = [x_1, x_2, \ldots, x_N]$ be the set of features, i.e., the values of the performance counters that we measure for each workload. A *model* is a function f that computes a scalar predicted performance value, $\hat{y} = f(X)$. Under a linear model, f has the form:

$$f(X) = \sum_{j=j}^{N} x_j \beta_j + \beta_0, \quad (1)$$

where $\beta = \{\beta_1, \beta_2, \ldots, \beta_N\}$ is a coefficient vector that corresponds to the features, and β_0 is a bias term called the *intercept*, which serves as a model adjustment factor. The *error* associated with the i^{th} workload is $y_i - f(X_i)$, where y_i is the empirically obtained CPF, and $f(X_i)$ the predicted CPF. Given training data, the generation of a coefficient vector is formulated as a constrained optimization problem [17]. The model generation techniques employed by HALPWE differ in terms of the optimization problem formulation and how it is refined by post-processing steps (Fig. 5).

B. Ordinary Least Squares (OLS)

Given a coefficient vector β, the *aggregate error* of the training data set is the *Residual Sum of Squares (RSS)*:

$$RSS(\beta) = \sum_{i=1}^{M} (y_i - f(X_i))^2. \quad (2)$$

Ordinary Least Squares (OLS) computes the coefficient vector β and intercept β_0 that minimizes $RSS(\beta)$ [17].

C. Non-Negative Least Squares (NNLS)

OLS may produce models that estimate negative CPF values for certain data sets, which is physically impossible. *Non-Negative Least Squares (NNLS)* [18] can be applied to ensure that model estimates cannot be negative. NNLS implicitly removes certain features from model by setting negative-valued coefficients to zero and distributing their impact amongst the remaining positive values. NNLS may degrade model accuracy as it no longer minimizes $RSS(\beta)$.

D. Feature Selection and Ranking

OLS and NNLS are *full regression* models that may use all input features. *Feature selection*, which removes feature x_j from the model by setting coefficient β_j to zero, can improve *prediction accuracy* by sacrificing bias to reduce variance, as well as *interpretation*: identifying a subset of features that exhibits the strongest effect on model accuracy enhances understanding of the underlying mechanisms [19].

Forward Stepwise Selection greedily selects coefficient pairs that achieve the maximal incremental improvement to the model; the process terminates when adding more features is no longer beneficial to model prediction accuracy. *Backward Stepwise Selection* is similar but starts with a full regression model and iteratively removes one feature at a time. We apply AIC and BIC as feature ranking criteria during stepwise selection. This provides us with four feature selection methods: {Forward, Backward}×{AIC, BIC}, which can be applied to either OLS or NNLS models.

For each model, we *rank* the selected features via *p-value hypothesis testing* [17] using a threshold of 0.05 to quantify their impact on model accuracy (a smaller p-value indicates greater significance). We report p-values for models that perform feature selection, omitting full regression models. We do not rank features for NNLS models, because the NNLS process discards features that break the assumption that model residuals follow a normal distribution.

E. Linear Regularization Model: Lasso

Lasso [20] is a *linear regularization model* that constructs a model while simultaneously selecting features using an RSS penalty term [17]. Lasso penalizes features in a blanket fashion, unlike step-wise selection, which is iterative. Lasso selects features via shrinkage, which reduces "small enough" coefficients to zero, depending on the value of the regularization term coefficient. We produce two variants of a Lasso, with and without the NNLS criterion.

F. Model Evaluation

We use 10-fold cross validation [21] as a precursor to quantify an estimate of the usefulness of a trained model in practice. We report the out-of-sample error (E_{out}), the mean absolute percentage error averaged over all ten folds, as our primary measure for model accuracy; E_{out} reflects the ability of a model to accurately predict a response when applied to previously unseen data. We also evaluate models in terms of their inlier ratios. Given a percentage threshold T, trace X_i is an *inlier* if X_i's *absolute relative percentage error (APE)* is less than T. Given T, the *inlier ratio* (I_R) is the percentage of traces that are inliers. We report 10 % and 20 % inlier ratios for each model we produce and compare inlier ratios across varying thresholds for comparative analysis of the prediction scenarios.

G. Random Forest Regression

Random Forest (RF) regression is a non-linear supervised learning model where prediction is an aggregate of individual predictions made by a set of regression trees. Due to space limitations, we omit describing RF in detail; interested readers may consult Ref. [22] for detail. We construct our RF using *bootstrap aggregation (bagging)*, applying *feature bagging* to reduce correlation among trees. We compute E_{out} using 10-fold cross-validation, by averaging the *out-of-bag error* for each fold. Regression trees and forests include all features by design. Although feature ranking via RSS error is performed, we do not report any feature rankings produced by RF models.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

V. Simulator Based models

Although the focus of this work is hardware-assisted modeling, we first create simulation-based models. This allowed us to build confidence in the cross-generational modeling approach by evaluating the broader capability of the hardware-assisted technique.

Using simulation-based modeling is easier, as it removes the well-known difficulties inherent to hardware-assisted modeling, such as limited architectural visibility, run-to-run noise cycle count and performance counter variations. Simulation modeling also provides greater degree control of the degree of difference between the generations of devices configured as host and target during model building, allowing us to isolate and evaluate well known design time tests.

A. Framework Validation Scenarios

We created 3 simulation-based prediction scenarios which were designed to be easy, thereby enabling us to validate our modeling suite. We used the GPU simulator to collect performance counters *and* to model the prediction target CPF. The simulator eliminates sources of non-determinism that can affect hardware-assisted models (see Section 6).

Scenario$_1$ (364 traces) configures the simulator as a 2-slice Broadwell GT3 and builds a model to predict its own CPF.

Scenario$_2$ (364 traces) configures the simulator as a 2-slice Skylake GT3 and builds a model to predict its own CPF.

Scenario$_3$ (364 traces) configures the GPU simulator as a 2-slice Broadwell GT3 running a Broadwell-generation driver and builds a model to predict the performance a 2-slice Skylake GT3 running the same driver. Although both GPUs have 48 EUs, the evolution from Broadwell to Skylake does include microarchitecture changes not reported in Table 2.

We generated 13 models for each scenario. For each model, we report the out-of-sample error, 10 % and 20 % inlier ratio, the number of selected features, and the number of available features; we also report the APE for each workload.

Tables 4 and 5 respectively report the best-performing non-NNLS and NNLS models that minimized the out-of-sample error for each of three scenarios listed above; Figs. 6 and 7 depict the observed CPF, predicted CPF, and APE for each workload for the three models listed in Tables 4 and 5. For all three scenarios, the best non-NNLS models produced lower out-of-sample errors than the best NNLS models.

B. Non-NNLS Models

The three models reported in Table 4 exhibit very low out-of-sample errors; the RF model for Scenario$_3$ had a slightly higher out-of-sample error than the OLS/Backward/AIC models for Scenario$_1$ and Scenario$_2$, which is to be expected because it is a cross-generation prediction scenario, whereas Scenario$_1$ and Scenario$_2$ are same-generation. All three models obtained 10 % inlier ratios of more than 80 % and varied slightly in terms of the number of selected features.

In Fig. 6, it is near-impossible to discern the difference between predicted and observed CPF for most workloads with the naked eye, which reinforces the accuracy of these models. The highest APEs are observed for workloads with the smallest CPFs on the left-hand side of the graphs, which suggests that the three models are stable; slightly higher APEs for workloads with large CPFs are observed for Scenario$_3$'s RF model, which we attribute to the fact that Scenario$_3$ entails cross-generation prediction.

C. NNLS Models

Comparing the non-NNLS models of Table 4 and Fig. 6, the NNLS models reported in Table 5 and Fig. 7 have a higher out-of-sample error while selecting fewer counters as features; however, when looking at Fig. 7 in detail, virtually all the visible increase in per-workload APE occurs for the workloads with the smallest CPFs. The gap in predictive accuracy between NNLS and non-NNLS models may not be as pronounced as one might interpret by considering out-of-sample error in isolation.

Scenario$_3$'s NNLS/Backward/BIC model has lower APEs for large-CPF workloads than Scenario$_3$'s non-NNLS RF model, which has a lower overall out-of-sample error. Likewise, Scenario$_3$'s NNLS/Backward/BIC model has a lower 10 % inlier ratio than the non-NNLS RF model; however, the outliers are clustered among workloads with the smallest CPFs. Similar, observations hold for Scenario$_1$ and Scenario$_2$ as well.

D. Driver Scalability Scenario

Scenario$_{3D}$ (364 traces) re-runs Scenario$_3$, modifying the target simulator to produce new validation data obtained used the same Skylake GT3 device, only now running the applications with the Skylake GT3 driver instead of the Broadwell GT3 driver. Updating the driver to increase application performance is a common optimization made by GPU designers, and it is imperative that predictive models can accurately predict generational changes in both the hardware and software stack. Scenario$_{3D}$ shows that the HALWPE regression suite selects features and produces models that account for the performance difference caused by updating the target platforms driver, producing accurate E_{out} estimates.

Table 6's last row presents the best performing non-NNLS model for Scenario$_{3D}$ and the first-row repeats Scenario$_3$'s best model result from Table 4. Table 7's last row presents the best performing NNLS model, and the first row repeats the last row of Table 5. Most notably, the out of sample error of the best performing NNLS model in Table 7 has decreased from 8.95 % to 7.40 % using the same features without retraining. The increased accuracy is a byproduct of using a small number of traces, 32, for validation. The reduced frame count is unavoidable as both driver versions were not compatible with all frames. This study demonstrates HALWPE's robustness when modeling driver generation updates.

E. Slice Scalability Scenario

Scenario$_{3S}$ (364 traces) re-runs Scenario$_3$, modifying the training data to use a Broadwell GT2 device to predict the original Skylake GT3 device. Modifying the training data approximates a scenario in which the host platform is both a generation older and has half the available parallelism of its target. It is likely that as new generations of GPU are developed and tested, their slice count will continue to increase.

The model is applied without retraining to a validation set with 60 single frame workloads, demonstrating HALWPE's ability to identify features that accommodate CPF changes due to slice scaling, obtaining high cross-generation prediction when the host has 200 % less parallelism than the target.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE IV
HIGHEST ACCURACY NON-NNLS MODELS SCENARIOS$_{1\text{-}3}$.

Scenario	Best Performing Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_1$	OLS/Backward/AIC	1.75%	100%	97.26%	34	42
Scenario$_2$	OLS/Backward/AIC	1.89%	91.77%	82.79%	31	42
Scenario$_3$	Random Forest (RF)	3.20%	98.91%	87.95%	38	42

TABLE VI
HIGHEST ACCURACY NON-NNLS MODEL SCENARIO$_{3D}$.

Scenario	Best Performing Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_3$	OLS/Backward/AIC	1.75%	100%	97.26%	34	42
Scenario$_{3D}$	Lasso	6.41%	93.55%	74.19%	23	42

TABLE VIII
HIGHEST ACCURACY NNLS MODEL SCENARIO$_{3S}$.

Scenario	Best Performing Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_3$	Random Forest (RF)	3.20%	98.91%	87.95%	38	42
Scenario$_{3S}$	Random Forest (RF)	4.81%	96.67%	93.33%	38	42

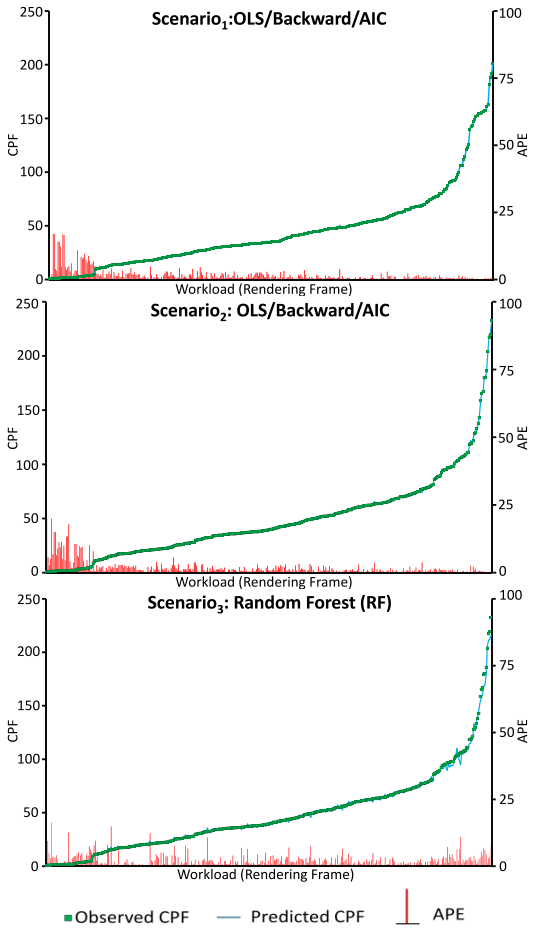


Fig. 6. The observed CPF, predicted CPF, and per-workload APE for the non-NNLS models of Scenario$_1$ (a), Scenario$_2$ (b), and Scenario$_3$ (c). Workloads are ordered from left-to-right in non-decreasing order of observed CPF.

Row 1 of Tables 8 and 9 re-report Scenario$_3$ results, and Row 2 reports the results of Scenario$_{3S}$'s slice scaling study. Comparing Scenario$_3$ and Scenario$_{3S}$, the error increases 2.3 %, resulting in an 11.5 % average error, but remains usable for pre-silicon performance estimation. The increase in error due to slice scaling is also present in our hardware-assisted scenarios.

TABLE V
HIGHEST ACCURACY NNLS MODELS SCENARIOS$_{1\text{-}3}$.

Scenario	Best Performing NNLS Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_1$	NNLS	2.22%	98.36%	97.27%	19	42
Scenario$_2$	NNLS	2.25%	97.54%	95.9%	15	42
Scenario$_3$	NNLS/Backward/BIC	8.95%	88.25%	74.04%	8	42

TABLE VII
HIGHEST ACCURACY NNLS MODEL SCENARIO$_{3S}$.

Scenario	Best Performing NNLS Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_3$	NNLS/Backward/BIC	8.95%	88.25%	74.04%	8	42
Scenario$_{3D}$	Lasso/NNLS	7.40%	93.55%	77.42%	11	42

TABLE IX
HIGHEST ACCURACY NNLS MODEL SCENARIO$_{3S}$.

Scenario	Best Performing NNLS Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_3$	NNLS/Backward/BIC	8.95%	88.25%	74.04%	8	42
Scenario$_{3S}$	Lasso/NNLS	9.75%	83.33%	66.67%	11	42

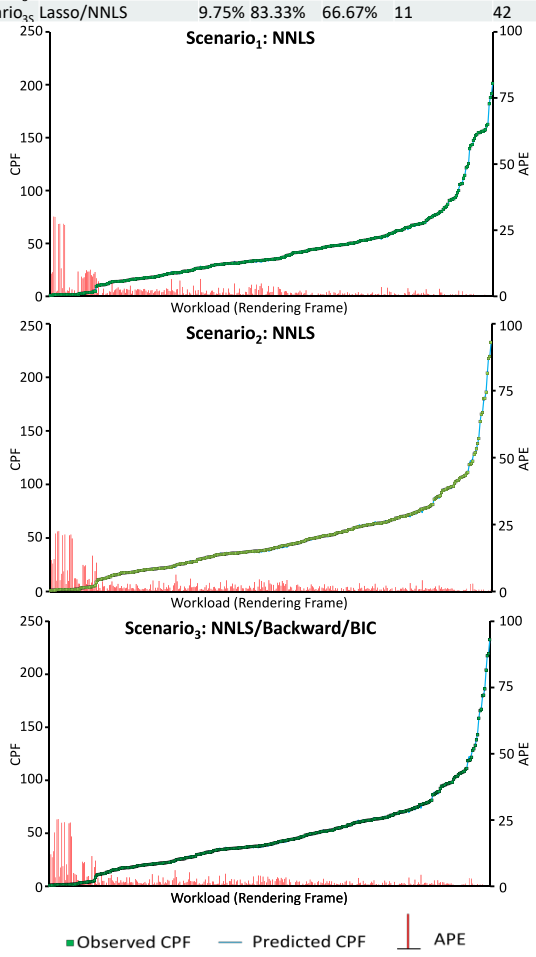


Fig. 7. The observed CPF, predicted CPF, and per-workload APE for the NNLS models generated for Scenario$_1$ (a), Scenario$_2$ (b), and Scenario$_3$ (c) in Table 5. Workloads are ordered from left-to-right in non-decreasing order of observed CPF.

VI. HARDWARE-ASSISTED MODELS

When profiling an application on commodity hardware, certain sources of non-determinism may arise that simulators either do not model or can suppress. We discuss strategies to mitigate these issues in detail before moving on to present the results of our hardware-assisted models.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

A. GPU Profiling and Mitigating Variation

Referring to Fig. 2, *HWTraces* refers to a stream of DirectX *GfxAPI* commands that we collect without firmware or driver modification using *GfxCapture*. *HWTraces* are repeatable, platform-independent, and allow instrumentation of the host GPU and API. We attach *GfxProfiler* directly to the *device context* [10], which is created along with its *device* when the GPU renders a frame. The device creates resources and queries the GPU's rendering capabilities, while the device context comprises the GPU's pipeline and resource states, which generate actual rendering commands.

GfxProfiler collects three classes of features: performance counter measurements (via HWTraces), profiled DirectX API commands (via HWTraces), and *hardware queries* (via the device context) which leverage exposed parts of the API. An exemplary hardware query is *PSInvocations*, the number of times the pixel shader invoked an EU while rendering.

Workload execution is performed using an unmodified operating system (OS; Windows 7) and driver. To reduce variability introduced by the OS, we suppress non-OS background processes and run traces in full-screen mode. By leaving the OS and driver unmodified, we eschew control of sleep states. By adjusting BIOS settings, we can disable deep sleep state RC6 and suppress dynamic frequency scaling and Turbo Boost. The sources of variation that remain are competing background tasks, which affect CPU-GPU communication latency, and access to shared resources, and the sleep states that we cannot control.

We perform outlier detection and elimination to mitigate variation. We apply the *Median Absolute Deviation (MAD)* test [23] to identify runs that exhibit abnormal behavior. We empirically determined a threshold of ±7 MADs using 10 representative frames, executing each frame 100 times.
During model construction and evaluation, we execute each frame 100 times on the host GPU using *GfxProfiler* to collect features. We remove outliers, i.e., all runs whose CPF values are outside of the ±7 MAD threshold. The CPF and feature values reported for the frame are averaged across the inliers.

Fig. 8 reports the CPF of 100 executions of Witcher 2 Frame 769 normalized to the smallest CPF that we observed. To avoid cold-start issues, we insert a generic "warmup" frame that is executed but not profiled. Most of executions are within the MAD window, although some non-negligible variation in CPF is clearly visible.

B. HALWPE Prediction Scenarios

We present three hardware-assisted predictive models based on performance counter measurements taken from a Haswell GT2 GPU, which provides 577 features. The results show that HALWPE can perform accurate cross-generation CPF prediction.

Scenario$_4$ (282 traces) uses a Haswell GT2 GPU host to predict the CPF of a simulated Broadwell GT2 GPU.
Scenario$_5$ (300 traces) uses a Haswell GT2 GPU host to predict the CPF of a simulated Broadwell GT3 GPU.
Scenario$_6$ (300 traces) uses a Haswell GT2 GPU host to predict the CPF of a simulated Skylake GT3 GPU.

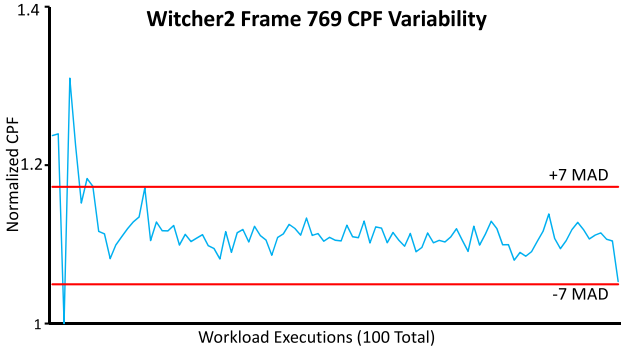


Fig. 8. CPF variability for a Witcher 2 Frame when executed 100 times; the first execution was removed due to cold start issues. Of the remaining 99 runs, 7 frames were identified as outliers and removed using the ±7 MAD approach.

Tables 10 and 11 respectively report the best-performing non-NNLS and NNLS models that minimized the out-of-sample error for each of three scenarios listed above; Figs. 9 and 10 depict the observed CPF, predicted CPF, and APE for each workload for the three models list in Tables 10 and 11. For Scenario$_4$ and Scenario$_5$, OLS/Forward/BIC produced the lowest out-of-sample errors; for Scenario$_6$, the NNLS produced the lowest out-of-sample error.

C. Non-NNLS Models

In Fig. 9, slight differences between predicted and observed CPF for the OLS/Forward/BIC model for Scenario$_4$ and Scenario$_5$ can be seen by the naked eye; the differences are more pronounced for Scenario$_6$'s RF model, especially for workloads with higher CPFs. The degradation in model quality is clear between scenarios.

The OLS/Forward/BIC models generated for Scenario$_4$ and Scenario$_5$, exhibited the largest APEs are at the low-CPF end up the spectrum; in contrast, the RF model generated for Scenario$_6$ has a more uniform distribution of high APEs across the CPF spectrum. This is similar to the distribution of APEs reported for the RF model in Fig. 6 for Scenario$_3$.

D. NNLS Models

The NNLS models produced for Scenario$_4$ and Scenario$_5$ in Table 11 nearly double the out-of-sample errors produced by the non-NNLS models in Table 10, with large reductions in the 10 % inlier ratios in both cases. In the case of Scenario$_6$, the NNLS model yielded an out-of-sample error of 8.91 %, which is slightly worse than the 7.45 % and 7.47 % produced by the best non-NNLS models for Scenario$_4$ and Scenario$_5$ in Table 10, but respectable given the challenges associated with CPF prediction across two GPU generations; it's 10 % inlier ratio was respectively 14.23 % and 12.77 % lower, which can be explained similarly.

TABLE X
HIGHEST ACCURACY NON-NNLS MODELS SCENARIOS$_{4-6}$.

Scenario	Best-Performing Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_4$	OLS/Forward/BIC	7.45%	92.76%	85.13%	40	577
Scenario$_5$	OLS/Forward/BIC	7.47%	91.33%	83.67%	30	577
Scenario$_6$	Random Forest (RF)	10.28%	95.85%	62.85%	577	577

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

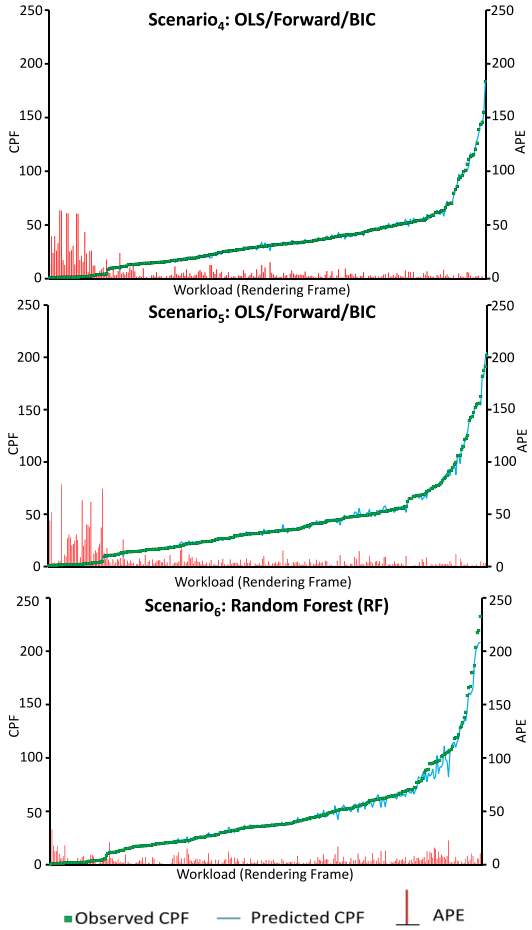


Fig. 9. The observed CPF, predicted CPF, and per-workload APE for the non-NNLS models summarized in Table 6. Workloads are ordered from left-to-right in non-decreasing order of observed CPF.

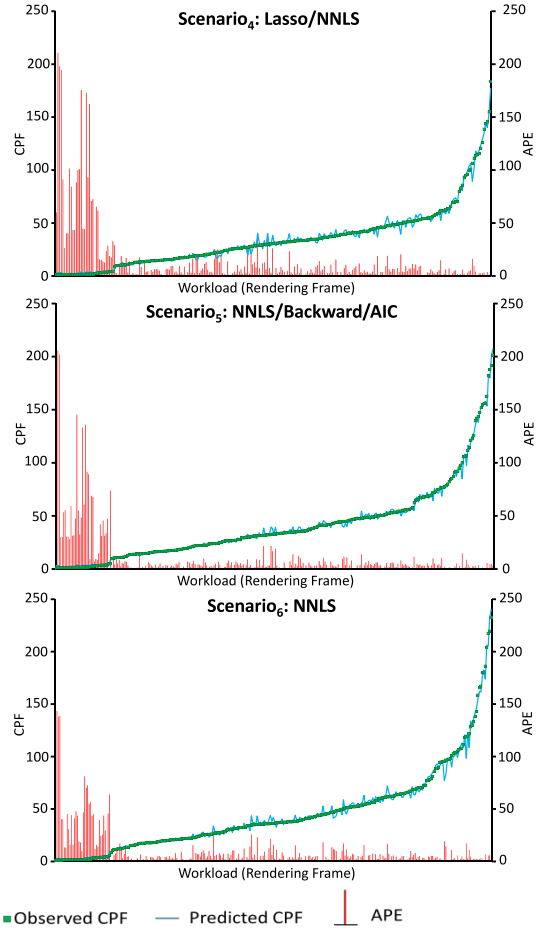


Fig. 10. The observed CPF, predicted CPF, and per-workload APE for the NNLS models summarized in Table 11. Workloads are ordered from left-to-right in non-decreasing order of observed CPF.

This level of accuracy should be sufficient for use in early-stage design space exploration; however, designers must understand that model accuracy will necessarily degrade as number of generations between the host and prediction target increases. Scenario$_6$ investigates this issue further.

E. Scenario$_6$ Model Comparison

Table 12 reports the accuracy of all 13 HALWPE models for Scenario$_6$. This study serves to justify the need for an ensemble of models, by assessing the differences in model accuracy in our most ambitious CPF prediction scenario, across two GPU generations. The out-of-sample error ranged from 8.91 % (NNLS) to more than 1000 % (four OLS variants); the four highly inaccurate OLS variants likely overfit the training data. Employing an ensemble of models increases the likelihood that at least one model does not overfit. Both RF (which is nonlinear) and Lasso (due to regularization) are less likely than OLS and NNLS variants to overfit the training data; including them in HALWPE's ensemble increases the likelihood that at least one model is accurate. RF, Lasso, and Lasso/NNLS did not overfit in any of our scenarios.

TABLE XI
HIGHEST ACCURACY NNLS MODELS SCENARIOS$_{4-6}$.

Scenario	Best-Performing NNLS Model	E_{out}	I_R (20%)	I_R (10%)	Features Selected	Available Features
Scenario$_4$	Lasso/NNLS	13.87%	75.06%	58.95%	23	577
Scenario$_5$	NNLS/Backward/AIC	13.68%	84.00%	73.67%	83	577
Scenario$_6$	NNLS	8.91%	92.63%	77.89%	59	577

Among the remaining nine models in the highest out-of-sample error was 19.69 % (NNLS/step-forward/BIC). As Tables 4-11 show, it is difficult to know a-priori which model (with or without the NNLS guarantee) will yield the highest overall accuracy, which justifies the ensemble approach. In Table 12, NNLS achieves the smallest out-of-sample error and is tied for third highest 10 % inlier ratio; however, it selects the most features of the remaining nine linear models.

F. Inlier Ratio vs. Out-of-sample Error

Scenarios exist in which a processor architect may prefer a predictive model that maintains a higher inlier ratio within a given threshold to a model that minimizes out-of-sample error. Treating the inlier ratio as a proxy for variance provides higher confidence in the fidelity of the model.

Fig. 11 depicts the inlier ratios at eight thresholds for five models generated for Scenarios$_{4-6}$. For each scenario, Fig. 11 includes HALWPE's OLS and NNLS variants that minimize

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

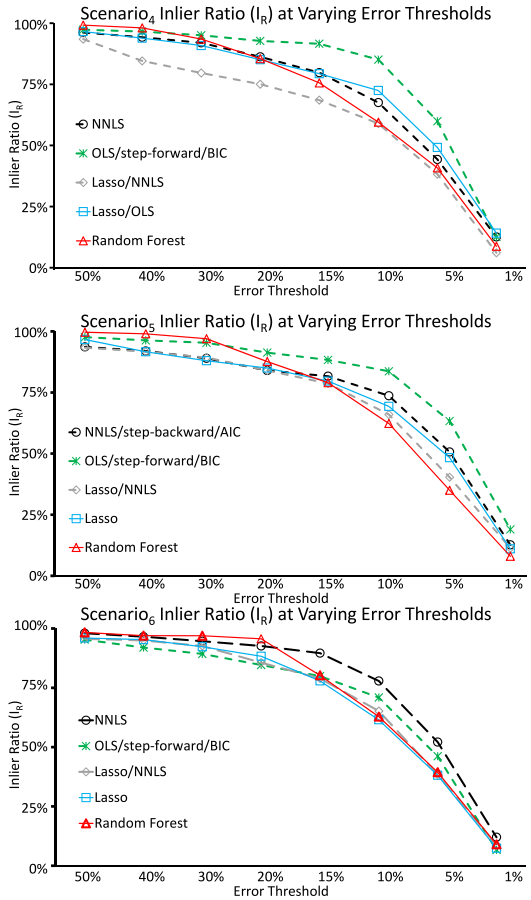


Fig. 11. Inlier ratios at various error threshold for predictive models generated for Scenario$_6$, Scenario$_7$, and Scenario$_8$. For each scenario, the best-performing OLS and NNLS-variants are shown, along with Lasso, Lasso/NNLS, and RF.

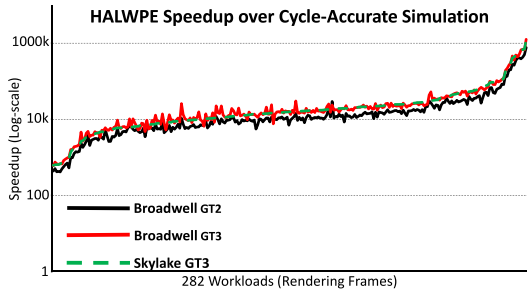


Fig. 12. HALWPE speedup over simulators for Broadwell GT2/GT3 and Skylake GT3 GPUs using 280 workloads common to all three simulators (Table 3). Frames are ordered by increasing Skylake GT3 speedup. Average speedups were 29481x for Broadwell GT2, 43643x for Broadwell GT3, and 44214x for Skylake GT3.

out-of-sample error, its two regularization models (Lasso, and Lasso/NNLS), and its non-linear model (RF).

For Scenario$_4$ and Scenario$_5$ OLS/Forward/BIC had the smallest out-of-sample error. For Scenario4 in Fig. 11, OLS/Forward/BIC has the highest inlier ratio at error thresholds of 30 % or lower, except for the 1 % threshold where NNLS and Lasso/OLS are marginally higher.

For Scenario5 it has the highest inlier ratio at error thresholds of 20 % or below; in Scenario$_6$, NNLS had the smallest out-of-sample error and the highest inlier ratios for thresholds of 15 % and lower. These observations reinforce the benefits of these three models: they exhibit low out-of-sample errors and high fidelity, so they can be used with high confidence.

G. *HALWPE Speedup*

Compared to cycle-accurate simulators that are properly tuned, predictive models sacrifice accuracy to provide computer architects with a rapid result.

TABLE XII
ACCURACY OF ALL HALWPE MODELS FOR SCENARIO$_6$.

Model	E_{out}	I_R (10%)	Features Selected	Available Features
OLS	>1000%	4.02%	299	577
NNLS	8.91%	70.90%	59	577
OLS/Forward/AIC	>1000%	9.43%	299	577
OLS/Forward/BIC	12.51%	77.89%	29	577
OLS/Backward/AIC	>1000%	4.02%	299	577
OLS/Backward/BIC	>1000%	4.02%	299	577
NNLS/Forward/AIC	14.32%	66.22%	53	577
NNLS/Forward/BIC	19.69%	44.15%	14	577
NNLS/Backward/AIC	12.78%	71.57%	59	577
NNLS/Backward/BIC	12.59%	70.90%	59	577
Lasso	12.24%	61.51%	12	577
Lasso/NNLS	13.34%	65.22%	11	577
RF	10.28%	62.85%	577	577

In the case of HALWPE, the execution time of the model on a given workload entails executing the workload trace on the Haswell host GPU and then applying the linear regression or RF model to the obtained features; in most cases, the latter is negligible. Fig. 12 compares the execution time of HALWPE to the simulator configured as a Broadwell GT2, Broadwell GT3, and Skylake GT3 GPU for the 282 common executable traces (Table 3). On average, HALWPE achieved a speedup of 29,481x over the Broadwell GT2 simulator, 43,643x over the Broadwell GT3 simulator, and 44,214x over the Skylake GT3 simulator. From workload to workload, the speedups reported in Fig. 12 vary considerably; in a few cases, the simulator executed a trace faster than the host GPU. Compared to cycle-accurate simulators, predictive models sacrifice accuracy to provide a rapid result.

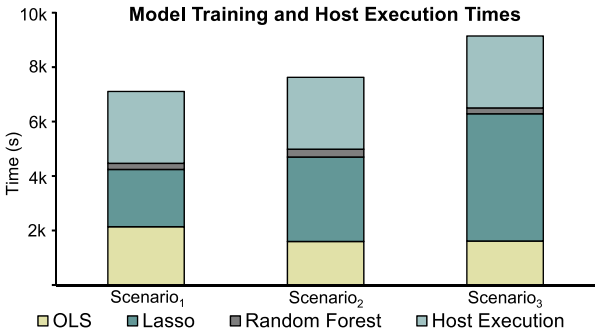


Fig. 13. Model training and host GPU execution time for Scenarios 4,5 and 6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2834398, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

The execution time of a model on a frame entails running the frame trace on the host GPU and then generating model using the obtained features; in most cases, the latter is negligible.

Fig. 13 reports the time to train all 13 HALWPE models -- excluding target simulation time and host GPU execution time to render each frame once. In addition to rendering, host GPU execution time includes overhead associated with loading the application, profiling, and streaming API commands from the trace player. The longest model training and host GPU execution time was ~2.5 hours for $Scenario_3$. We rendered each frame 100 times, thus execution time is dominated by the host GPU, not model training.

When comparing Haswell to Skylake the total performance increase is 820.79 % on average, and 242 % on median. HALWPE can provide accurate models, which utilize a Haswell generation GPU to predict the performance of a Skylake generation GPU, as demonstrated in $Scenario_6$.

VII. Feature Ranking

This section reports the 10 highest ranked features from the models generated for $Scenario_4$, $Scenario_5$, and $Scenario_6$. We rank the features using the OLS/Forward/BIC model, which was the best performing model for $Scenario_4$ and $Scenario_5$, and the third best for $Scenario_6$. In our system, 577 features are available (see Section 6). In the feature list tables presented in the next two subsections, performance counters are yellow rows, DirectX metrics are blue rows, and hardware queries are grey rows. All the features reported in these subsections have been publicly disclosed [2, 3-11].

A. Broadwell GT2/GT3

Tables 13 and 14 report the top-ten most influential features for the OLS/Forward/BIC models produced for $Scenario_4$ and $Scenario_5$, ranked by p-values (Section 4.D). These scenarios respectively target the Broadwell GT2 and GT3 GPU, where the latter has twice as many slices/EUs.

In Table 13, the top-5 highest ranked features provide information about EU activity relating to front-end render pipeline units. The top-2 features are EU active and stall times, which holds consistent with the observation that the limited parallelism in a single-slice Broadwell GT2 GPU can impede performance. The 3rd through 5th highest ranking features report compute and domain shader EU activity, which suggests that interactions between the two shaders and their EU occupations should be analyzed. Notably, the presence of A19_CSEUStallTime suggests that limiting the amount of time that the compute shader stalls the EU array could potentially improve overall GPU throughput.

In Table 14, EU busy/stall cycles no longer fall within the top 10 ranked counters, which reflect the increase in parallelism provided by GT3 GPUs. The two counters representing the domain shader are ranked 8th and 10th, suggesting that they still influence performance, but that other subsystems with higher-ranking features should be given priority for analysis. Again, A19_CSEUStallTime is within the top-5 ranked features, which suggests that the process by which the compute shader stalls the EU array still influences performance significantly.

TABLE XIII
Top-10 model features ranked by p-value for the OLS/Forward/BIC produced for $Scenario_4$.

Performance Counter Name, Category, and Description	P-value
A00_EUBusyTime (clocks; HW Counter) Number of cycles that all cores spent actively executing instructions.	1.78 E-125
A01_EUStallTime (clocks; HW Counter) Number of cycles on all cores where the EU was not idle or active.	2.35 E-83
A12_DSThreadBusyTime (clocks; HW Counter) Number of cycles the domain shader was active on all cores.	2.02 E-49
A19_CSEUStallTime (clocks; HW Counter) Number of cycles compute shader stalled on cores; all cores also stalled	2.75 E-49
A14_DSEUStallTime (clocks; HW Counter) Number of cycles domain shader stalled on cores; all cores also stalled	6.22 E-39
IDirect3DDevice9_DrawPrimitiveUP (count; DirectX Metric) Number of times that data specified by a user memory pointer as a sequence of primitives was rendered.	4.26 E-21
IDirect3DQuery9_OcclusionQuery_GetData (count; DirectX Metric) Number of all queries for occlusion data.	3.04 E-18
ID3D11DeviceContext_DrawInstanced (count; DirectX Metric) Number of times non-indexed, instanced primitives were drawn.	1.72 E-16
A39_OMZTestFail (count; HW Counter) Number of pixels/samples that failed the Z test after pixel shader execution.	9.81 E-16
ID3D11DeviceContext_ExecuteCommandList (count; DirectX Metric) Number of queue commands issued from a command list onto a device.	4.56 E-15

TABLE XIV
Top-10 model features ranked by p-value for the OLS/Forward/BIC produced for $Scenario_5$.

Performance Counter Name, Category, and Description	P-value
ID3D11DeviceContext_ClearUnorderedAccessViewFloat (clocks; DirectX Metric) Number of cycles spent accessing memory through a resource.	4.95 E-172
A41_GpuBusy (clocks; HW Counter) Number of cycles the render engine was not idle.	2.48 E-91
IDirect3DDevice9_DrawIndexedPrimitive (clocks, DirectX Metric) Number of cycles spent rendering a primitive into an array of vertices via indexing.	7.62 E-46
A17_CSThreadBusyTime (clocks; HW Counter) Number of cycles the compute shader was active on all cores.	1.25 E-31
A22_GSThreadBusyTime (clocks; HW Counter) Number of cycles the geometry shader was active on all cores.	2.68 E-29
IDirect3DDevice9_DrawPrimitiveUP (clocks; DirectX Metric) Number of cycles spent rendering data specified by a user memory pointer as primitives.	8.21 E-23
ID3D11DeviceContext_DrawInstanced (count; DirectX Metric) Number of times non-indexed, instanced primitives were drawn.	9.17 E-19
A14_DSEUStallTime (clocks; HW Counter) Number of cycles domain shader stalled on cores; all cores also stalled.	1.57 E-18
IDirect3DQuery9_OcclusionQuery_GetData (count; DirectX Metric) Number of queries for occlusion data.	1.79 E-18
A12_DSThreadBusyTime (clocks; HW Counter) Number of cycles the domain shader was active on all cores.	3.68 E-18

The geometry shader now appears in the top-10 highest ranked features with the inclusion of A22_GSThreadBusyTime, which suggests that as parallelism increases doubles Broadwell GT2 to GT3, additional front-end units start to influence performance.

B. Skylake GT3

Table 15 reports the 10 most influential features for the OLS/Forward/BIC model for $Scenario_6$ ranked by p-values. The top two features from $Scenario_5$ (Table 14) remain in the top ten for $Scenario_6$. The highest-ranking feature in Table 15, the number of cycles not idle, is the second feature in Table 14.

This suggests that the ability to provide the render engine with a steady supply of data remains a critical indicator of performance; potential optimizations ensure that the GPU can consume enough data to avoid idle states. In Fig. 4, we see that

Broadwell GT3 and Skylake GT3 had similar CPF profiles, and that the gap in favor of Skylake 3 was small. Table 15 suggests that Skylake GT3 CPF can be predicted foremost by back-end pixel-based metrics corresponding to pixel shader memory activity, pixel shader invocations, and the render engine's ability to pre-emptively avoid pixel processing by killing pixels (A36_RSKillPixelCount); A40, which counts render target writes also corresponds to the back-end. Compared to Table 14, the absence of geometry and compute shader metrics suggests that changes to the Skylake GT3 front end may have removed performance bottlenecks present in Broadwell GT3.

Table 15 also includes domain and hull shader metrics, which correspond to the first and last stages of the DirectX tessellation pipeline. This suggests that tessellation has emerged as a prime indicator for CPF for Skylake, and that it may be a suitable candidate for further optimization.

C. Discussion

This analysis shows how to interpret predictive models to obtain insights regarding which features have the greatest predictive impact on CPF. A highly-ranked feature may hint that a subsystem that could benefit from further architectural improvement; however, models are inexact, and, for a given scenario, feature rankings may vary from model to model. Thus, feature ranking can provide "hints" to understanding performance, not solutions These hints are symptoms, and should not be misconstrued as having diagnostic abilities to validate the existence of the bottlenecks or to identify root causes. Any hint provided by a model should be validated by further simulation before being accepted as fact. Architects should use predictive models judiciously and conservatively and should not misconstrue them as automated substitutes for existing methodologies or human intelligence.

.

TABLE XV
TOP-10 MODEL FEATURES RANKED BY P-VALUE FOR THE OLS/FORWARD/BIC PRODUCED FOR $SCENARIO_6$.

Performance Counter Name, Category, and Description	P-value
A41_GpuBusy (clocks; HW Counter) Number of cycles the render engine was not idle.	2.15E-84
ID3D11Device_CreateHullShader (count; DirectX Metric) Number of times a hull shader was created.	4.67E-33
A15_DSThreadsLoadedCount (count; HW Counter) Number of domain shader threads issued to EUs for execution.	3.82E-18
A36_RSKillPixelCount (count; HW Counter) Number of pixels/samples killed in the pixel shader.	6.29E-16
IDirect3DDevice10_PSSetConstantBuffers (count; DirectX Metric) Number of calls to allocate constant buffers used by the pixel shader pipeline stage.	3.44E-15
ID3D11DeviceContext_ClearUnorderedAccessViewFloat (clocks; DirectX Metric) Number of cycles spent accessing memory through a resource.	1.24E-14
PSInvocations (count; HW Query) Number of pixel shader invocations.	2.00E-12
IDirect3DDevice9_SetRenderTarget (count; DirectX Metric) Number of allocations of memory regions to declare color information for the render target.	5.87E-12
IDirect3DIndexBuffer9_Unlock (clocks, DirectX Metric) Number of cycles spent attempting to unlock index buffers used by the vertex shader and render engine.	5.13E-11
A40_OMSamplesWritten (count; HW Counter) Number of 3D render target writes.	2.04E-10

VIII. RELATED WORK

Cycle-accurate simulators such as GPGPU-Sim [24], Atilla [25], and Multi2Sim [26] run orders of magnitude slower than native execution [27, 28]. Techniques to reduce simulation times, such as representative sampling [29], multi-threading [30], and synthetic benchmark generation, such as for cache-coherent traffic [31] [32] are helpful but remain prohibitively slow. In response, we have turned to predictive modeling intended to speedup pre-silicon GPU design by avoiding cycle-accurate simulation in favor of estimating CPF via predictive models instead. HALWPE can leverage reduced simulation time to reduce model training time, with a reduction in speedup.

A. Predictive Models for CPUs

HALWPE is inspired by predictive models for CPUs performance and power using *linear regression* [33] and *artificial neural networks* (*ANNs*) [34]. Like HALWPE these models are built using performance counter readings and program metrics as features; however, the features that effectively predict CPU and GPU performance are different due to architectural dissimilarities. Cycle-accurate simulator internals can be more effective predictors than performance counters [35]; however, this requires simulation of each new workload before its performance can be predicted.

Cross-architecture models can predict the performance of an application from one CPU to another and can overcome ISA and microarchitectural differences with 90 % accuracy on computationally-intensive embedded kernels [36], and up to 97 % accuracy for both performance and power consumption when prediction is performed on the granularity of program phases [37]. Like HALWPE, the predictive features are performance counter measurements obtained from direct execution, which lends credence to our approach.

Approximate analytical models for out-of-order processors can perform high-level microarchitectural analysis [38]. This method requires a trace-driven off-line analysis of the model parameters to determine program locality behavior and miss rates as well as drain time after branch miss-prediction. The model does not generalize for workloads that have not been included in the off-line analysis.

B. Predictive Models for GPUs

Predictive models for GPUs based on linear regression [39] decision trees [40], random forests [41] and ANNs [42] can accurately predict performance and power consumption for the GPUs on which they were trained. HALPWE, in contrast, offers cross-generation prediction capabilities.

The work most closely related to HALWPE is an ANN-based model which can accurately predict the performance and power consumption of an application across a variety of GPU configurations comprising core frequency, available parallelism, memory bandwidth, etc. [28]. Like HALWPE, performance counter measurements are collected using a real GPU. The authors of the study modified the GPU firmware to control the number of active compute units, core frequencies, and memory frequencies, yielding 448 different configurations. They then trained an ensemble model that allows the user to

predict the performance and power consumption of an application (given collected performance counter measurements) on any of the 448 configurations.

HALWPE differs from the ANN predictor in two respects. First, HALWPE achieves cross-generation performance prediction, while the ANN predictor is limited to variants of the current-generation (host) GPU with three degrees of freedom. Second, HALWPE does not modify the firmware, enabling usage of production drivers. In its favor, the ANN predicts performance and power consumption, while HALWPE, presently, is limited to CPF prediction. Our evaluation of HALWPE focuses on graphics and gaming workloads, whereas the ANN predictor was evaluated using OpenCL workloads spanning several application domains.

A prior work on single-generation predictive performance and power models for ATI GPUs [43] shares many principle similarities with our work. They use Random Forest Regression for prediction and model feature ranking. We extend the feature ranking capability to all models generated using P-Value significance testing.

XAPP [44] uses single-threaded CPU performance counter measurements to predict performance when porting an application to a GPU. XAPP's objective is to determine which portions of a program will reap the greatest benefit from GPU acceleration, which is a challenging problem in performance tuning for heterogeneous platforms; in contrast, HALWPE focuses on early-stage GPU architectural design space exploration within an architecture family and early performance feedback for graphics software development.

Analytical GPU models can predict performance and power consumption [45] but require extensive tuning and require co-execution with functional simulators. In contrast, HALWPE automates model training and feature selection without co-simulation, thereby making it much easier to use in practice.

As performed in this work and explicitly in [46], it is important to compare the relative difference between generations of architecture when evaluating workload performance. Ref. [46] compares the performance of two dedicated graphics cards, a Maxwell and Pascal GPU across 6 different matrix factorization workloads. In this work, we evaluate 4 integrated GPU architectures on 36 distinct rendering workloads, to study the effectiveness of cross-generational performance prediction.

IX. CONCLUSION AND FUTURE WORK

HALWPE has established the feasibility of cross-generation GPU CPF prediction using performance counter readings, DirectX metrics, and hardware queries. HALWPE achieved an out-of-sample error rate of 8.91 % when predicting across two GPU architecture generations, which include extra parallelism, microarchitecture changes, and driver upgrades. HALWPE achieved a speedup of 44214x compared to a cycle-accurate simulator for this prediction scenario. Our results and analysis suggest that predictive modeling can aid early-stage microarchitecture design space exploration and may be able to help with identification of performance bottlenecks; however, predictive modeling must be applied with care, as the models themselves are inherently finicky.

Several open questions remain: (1) Can HALWPE predict energy consumption with equal effectiveness? (2) Can models be re-used to predict CPF of similar targets without retraining? Criteria for model retraining/reuse would be beneficial. (3) Can HALWPE generalize to other GPU vendors, rendering APIs, and to GPGPU workloads with greater workload diversity?

REFERENCES

[1] K. O'Neal, P. Brisk, E. Shriver, and M. Kishinevsky, "HALWPE: Hardware-Assisted Light Weight Performance Estimation for GPUs" in Proc. Of Annual Design Automation Conference (DAC), 2017.

[2] Intel Corporation, "Open Source Intel HD Graphics Programmer's Reference Manual" [Online]. pp. 13-15 Available: https://goo.gl/9KSJks

[3] Intel Corporation, "Intel GPA GPU Metrics", Available: https://goo.gl/3Ird2x

[4] Microsoft Corporation, "ID3D10Device Interface" [Online]. Available: https://goo.gl/4YNKpl

[5] Microsoft Corporation, "ID3D11DeviceContext Interface" [Online]. Available: https://goo.gl/4VnmIj

[6] Microsoft Corporation, "ID3D11Device Interface" [Online]. Available: https://goo.gl/gzeD0o

[7] Microsoft Corporation, "IDirect3DDevice9 Interface" [Online]. Available: https://goo.gl/HCSNTw

[8] Microsoft Corporation, "IDirect3DIndexBuffer9" [Online]. Available: https://goo.gl/flRm1A

[9] Microsoft Corporation, "IDirect3DQuery9 Interface" [Online]. Available: https://goo.gl/UZDyRD

[10] Microsoft Corporation, "Introduction to a Device in Direct3D 11" [Online]. Available: https://goo.gl/bi6USV

[11] Microsoft Corporation, "New Resource types in Direct3D 11" [Online]. Available: https://goo.gl/gX4MZz

[12] Intel Corporation, "The Compute Architecture of Intel Processor Graphics Gen7.5." [Online]. Available: https://goo.gl/5HZ54v

[13] Intel Corporation, "The Compute Architecture of Intel Processor Graphics Gen8." [Online]. Available: https://goo.gl/TnpAGc

[14] Intel Corporation, "The Compute Architecture of Intel Processor Graphics Gen9." [Online]. Available: https://goo.gl/RMmUc6

[15] H. Akaike, "A new look at the statistical model identification" in *IEEE Transaction on Automatic Control*, 1974, pp. 716-723.

[16] B.N. Petrov, and F. Csáki, "Information theory and an extension of the maximum likelihood principle", in 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, USSR, 1971, pp. 267-281.

[17] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. Springer series in statistics. Springer, New York, 2001.

[18] C. Lawson, and R. Hanson, Solving Least Squares Problems, Siam by Applied Mathematics (1995).

[19] R. R. Hocking, "The Analysis and Selection of Variables in Linear Regression", *Biometrics, Vol. 32, No. 1*, 1976, pp. 1-49.

[20] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso", in Journal of The Royal Statistical Society, Series B Vol. 58, No. 1, 1996, pp.367-288.

[21] R. Kohai, A study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection, IJCAI, 1995.

[22] L. Breiman, Random Forests, Machine Learning, Vol. 45, Issue 1, 2001. pp. 5-32

[23] C. Leys, K. Olivier, P. Bernard, and L. Laurent, "Detecting Outliers: Do not use standard deviation around the mean, use absolute deviation around the median" in Journal of Experimental Social Psychology, 2013.

[24] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," in Proc. of the Int'l Symp. on Performance Analysis of Systems and Software (ISPASS), 2009, pp. 163-174.

[25] V.M. del Barrio, C. Gonzalez, J. Roca, and A. Fernandez, R. Espasa, "ATILLA: a cycle-accurate execution drive simulator for modern GPU architectures" in International Symposium on Performance Analysis of Systems and Software, (ISPASS) March 2006 pp. 231-241.

[26] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2Sim: A Simulation Framework for CPU-GPU Computing," in Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT), 2012.

[27] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, "Sources of Error in Full-System Simulation," in Proc. of the Int'l Symp. on Performance Analysis of Systems and Software (ISPASS), 2014.

[28] G. Wu, J.L. Greathouse, A. Lyashevsky, N. Jayasena, and D.Chiou "GPGPU Performance and Power Estimation Using Machine Learning" in IEEE International Symposium on High Performance Computer Architecture, 2015, pp. 564-576.

[29] W. Jia, K. Shaw, and M. Martonosi, "Starchart: Hardware and Software Optimization Using Recursive Partitioning Regression Trees," in Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT), 2013.

[30] S. Lee and W. W. Ro, "Parallel GPU Architecture Simulation Framework Exploiting Work Allocation Unit Parallelism," in Proc. of the International Symposium on Performance Analysis of Systems and Software (ISPASS), 2013.

[31] M. Badr, N. E. Jerger, "SynFull: Synthetic traffic models capturing cache coherent behaviour", 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), pp. 109-120, June 2014.

[32] J. Yin, O. Kayiran, M. Poremba, N. E. Jerger, "Efficient synthetic traffic models for large, complex SoCs," 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 297--308, 12-16, March 2016.

[33] E. ¨Ipek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz, "Efficiently Exploring Architectural Design Spaces via Predictive Modeling," in Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2006.

[34] B. C. Lee and D. M. Brooks, "Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction," in Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2006.

[35] B. Ozisikyilmaz, G. Memik, and A. Choudhary, "Machine Learning Models to Predict Performance of Computer System Design Alternatives" in 37th International Conference on Parallel Processing (ICPP), 9-12 Sept 2008, pp. 495-502.

[36] X. Zheng, P. Ravikumar, L.K. John, and A. Gerstlauer, "Learning-based Anlaytical Cross-Platform Performance Prediction" in Embedded Computer Systems, Architectures, Modeling and Simulation (SAMOS), July 2015, pp. 52-59.

[37] X. Zheng, L. John, A. Gerstlauer, "Accurate Phase-Level Cross-Platform Power and Performance Estimation" in Design Automation Conference (DAC), June 05-09, 2016, pp. 4.

[38] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A Mechanistic Performance Model for Superscalar Out-of-Order Processors," ACM Transactions on Computer Systems, vol. 27, no. 2, pp. 3:1–3:37, May 2009.

[39] P. E. Bailey, D. K. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. R. de Supinski, "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," in Proc. of the Int'l Conf. on Parallel Processing (ICPP), 2014.

[40] X. Ma, M. Dong, L. Zhong, and Z. Deng, "Statistical Power Consumption Analysis and Modeling for GPU-Based Computing" Proceedings of the ACM SOSP Workshop on Power Aware Computing and Systems (HotPower), October 2009.

[41] J. Chen, B. Li, Y. Zhang, L. Peng, and J. Peir, "Tree structured analysis on GPU Power study" in IEEE 29th International Conference on Computer Design (ICCD), 2011, pp. 57-64.

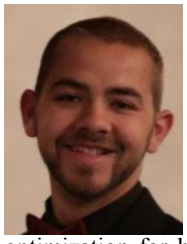
[42] S. Song, S. Chunyi, B, Rountree, and K.W. Cameron, "A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures", in International Symposium on Parallel & Distributed Processing (IPDPS), 2013. Pp. 673-686.

[43] Y. Zhang, Y. Hu, B. Li, and L. Peng, "Performance and power analysis of ATI GPU: a statistical approach," in Proc. Int. Conf. Networking, Architecture and Storage (NAS), 2011, pp. 149-158.

[44] N. Ardalani, C. Lestourgeon, K. Sangkaralingam, X. Zhu, "Cross-architecture performance prediction (XAPP) using CPU to predict GPU performance" in in Proc. of the 48th International Symposium on Microarchitecture(MICRO-48), 2015 pp 725-737.

[45] S. Hong, and K. Hyesoon, "An integrated GPU power and performance model", in Proc. of the 37th International Symposium on Computer Architecture (ISCA), 2010 pp. 280-289.

[46] X. Xie, W. Tan, L. Fong, and Y. Liang, "CuMF_SGD: Parallelized Stochastic Gradient Descent for Matrix Factorization on GPUS" in Proc. of the 26th International Symposium on High-Performance Parallel and Distributed Computing (HPDC), 2017. pp 79-92.



Kenneth O'Neal received the B.S. and M.S. degrees from University of California Riverside in 2012 and 2016 respectively. He is presently pursuing a Ph.D. degree in CS at UCR. He has also worked as an intern at Intel corporation in 2014, 2015, 2016, and 2017. His research interests include power and performance analysis and optimization for heterogeneous computing platforms with an emphasis on GPUs and FPGAs, and algorithmic design for high-level synthesis of Digital Microfluidic BioChips.

Philip Brisk received the B.S., M.S., and Ph.D. degrees from the University of California at Los Angeles, in 2002, 2003, and 2006, respectively, all in computer science. From 2006 to 2009, he was a Post-Doctoral Scholar with the Processor Architecture Laboratory, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL). He is an Associate Professor with the Department of Computer Science and Engineering, University of California at Riverside, Riverside. His current research interests include programmable microfluidics, FPGAs, compilers, and design automation and architecture for application specific processors.

Dr. Brisk was a recipient of the Best Paper Award at CASES 2007 and FPL 2009. He has been a Program Committee member for several conferences and workshops, including DAC, ASPDAC, DATE, VLSI-SoC, FPL, and FPT. He has been the General (Co-)Chair of the IEEE SIES 2009, the IEEE SASP 2010, and IWLS 2011, and the Program (Co-)Chair of the IEEE SASP 2011, IWLS 2012, ARC 2013, and FPL 2016.

Emily Shriver is a research scientist in system design and rchitecture at Strategic CAD Labs at Intel. She conducts research on power and performance modeling and simulation techniques across a broad spectrum of abstraction levels; circuits, RTL, emulation, architectural, and system level platform and software. She has coauthored over 20 published conference and journal papers, been an invited panelist at DAC, and served on the technical program committees of ISCA, ICCD, and DAC.

Michael Kishinevsky received the MS and PhD degrees in CS from the Electrotechnical University of St. Petersburg. He leads a research group in system design and architecture at Strategic CAD Labs of Intel. Prior to joining Intel in 1998, he has been a research fellow at the Russian Academy of Science, a senior researcher at a start-up in asynchronous design (TRASSA), a visiting associate professor at the Technical University of Denmark, and a professor at the University of Aizu, Japan. He coauthored three books in asynchronous design and has published more than 100 journal and conference papers. He received the Semiconductor Research Corporation outstanding mentor awards (2004 and 2010) and the best paper awards at DAC (2004) and ASD (2009).