Online Power Estimation of Graphics Processing Units

Vignesh Adhinarayanan, Balaji Subramaniam, Wu-chun Feng
Department of Computer Science, Virginia Tech
Email: avignesh@vt.edu, balaji@cs.vt.edu, wfeng@vt.edu

***Abstract*—Accurate power estimation at runtime is essential for the efficient functioning of a power management system. While years of research have yielded accurate power models for the online prediction of instantaneous power for CPUs, such power models for graphics processing units (GPUs) are lacking. GPUs rely on low-resolution power meters that only nominally support basic power management. To address this, we propose an instantaneous power model, and in turn, a power estimator, that uses performance counters in a novel way so as to deliver accurate power estimation at runtime.**

Our power estimator runs on two *real* NVIDIA GPUs to show that accurate runtime estimation is possible *without* the need for the high-fidelity details that are assumed on simulation-based power models. To construct our power model, we first use correlation analysis to identify a concise set of performance counters that work well despite GPU device limitations. Next, we explore several statistical regression techniques and identify the best one. Then, to improve the prediction accuracy, we propose a novel application-dependent modeling technique, where the model is constructed *online* at runtime, based on the readings from a low-resolution, built-in GPU power meter.

Our quantitative results show that a multi-linear model, which produces a mean absolute error of 6%, works the best in practice. An application-specific quadratic model reduces the error to nearly 1%. We show that this model can be constructed with low overhead and high accuracy at runtime. To the best of our knowledge, this is the first work attempting to model the *instantaneous* power of a *real* GPU system; earlier related work focused on average power.

I. Introduction

Today's high-performance computing (HPC) clusters are power-limited, and this trend is expected to continue for the foreseeable future. For instance, an exascale system projected to arrive in 2022 is expected to operate under a strict 20-megawatt power envelope [1], [2]. Realizing the above target will require a nearly 30-fold increase in performance with only a 1.2-fold increase in power consumption. As a consequence, the HPC community seeks to exploit both software and hardware innovations to meet the exascale goal.

Traditionally, on the software side, a runtime system manages the system power consumption [3], [4]. Power models play a vital role in the efficient functioning of such runtime systems by estimating the instantaneous power consumption of the system. While power *meters* provide the same benefit, power *models* go significantly beyond in that they enable a number of energy-management techniques, as explained in Section II.

On the hardware end, graphics processing units (GPUs) offer high performance and better energy efficiency than traditional CPU-only systems [5]. As a result, the HPC community has embraced the use of GPUs in their quest to build energy-efficient systems, as exemplified by the Green500 list (see Figure 1) [6]. However, the proliferation of GPUs in HPC systems has exposed the community to challenges in modeling the power consumption of GPUs.

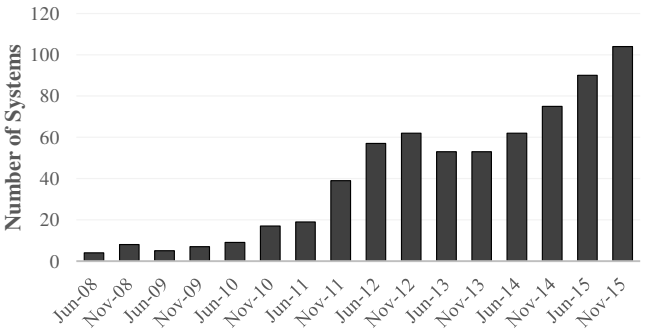


Fig. 1: Systems using accelerators in the Green500 lists

Though extensive CPU studies have led to some convergence on the power models used for CPU systems, the community is yet to understand the models required to estimate GPU power consumption. For example, linear models are widely accepted to be suitable for estimating CPU power consumption. Such acceptance is not established for GPUs. Some researchers have shown that linear regression-based techniques are suitable for GPU architectures [7] while others argue that such models may be insufficient to capture the complexities of a modern GPU architecture [8]. Moreover, the conclusions drawn from these studies are based on experiences gained in estimating the *average power* consumption. However, for these models to be useful in a runtime system, they should predict *instantaneous power* and should possess the following properties:

- **Accuracy:** The models have to estimate instantaneous power consumption accurately (less than 5% error rate) and track power-phase changes so that a runtime system can make correct decisions for power management.
- **Overhead:** Given the premium on performance in an HPC system, estimating the power consumption should not adversely affect the performance of the system under consideration. Therefore, monitoring system activity and predicting power should incur minimal overhead (not more than 1%).

Limitations of existing GPU power models: While a number of GPU power models have been explored recently, they all suffer from one of the following limitations.

1) They require multiple application runs and work only offline [7]–[11]. This renders the model unusable in a runtime system.
2) They work only in a simulator as the parameters used in these models cannot be measured on a real system [12], [13].

Contributions: Considering the above limitations in existing GPU power models and the previously described requirements for runtime power models, we make the following contributions in this paper:

- We present the *first* realization of an *instantaneous* power model for GPUs that is capable of providing live, runtime power estimates on real hardware.
  - Towards achieving the above, we perform a rigorous comparison of five types of statistical models.
  - To improve the model's accuracy, we introduce temperature-awareness to the model. While common in low-level models, modeling the temperature effects is generally lacking even in the well-studied higher-level CPU power models that are based on performance counters.
- To improve the accuracy of the instantaneous GPU power model, we introduce the notion of application-dependent models. To make this practical, we propose and evaluate the following two techniques:
  - We construct the power model from a smaller problem size to minimize the per-application data collection and modeling overhead.
  - Alternatively, we could construct these power models *online* at runtime using the GPU's built-in power sensors for feedback. Our evaluation of both these proposals show promising results with a mean absolute error rate of 1% and negligible overhead.
- Finally, we build and evaluate architecture-independent, portable models for estimating power across platforms.

Our major findings include the following:

- In the case of application-independent models, multiple linear regression produces the highest accuracy with a mean absolute error rate less than 6% for both microarchitecture generations of NVIDIA GPUs under consideration.
- Temperature plays a significant role in determining the power consumed by the GPU. For all models evaluated, introducing temperature awareness improved the prediction accuracy.
- Application-dependent models give significantly higher accuracy with a mean absolute error rate of nearly 1% for both GPUs using quadratic models. Our results also reveal that the penalty of using a quadratic model is higher when sufficient information is not available.
- Only a minimal number of samples (one hundred as determined by our experiments) is required to construct an accurate application-dependent model at runtime. This indicates that the model can be constructed at runtime with a low overhead and high accuracy.

The rest of the paper is organized as follows. We motivate our work in Section II. Section III discuses background related to the hardware platforms, benchmark applications, and statistical techniques used in this paper. Section IV describes our methodology. We present our evaluation in Section V. Section VI discusses the related work and Section VII concludes our paper.

II. Motivation

With the increasing prevalence of built-in power meters in today's high-end GPUs, it may seem that modeling instantaneous power is unnecessary. However, a performance counter-based power model offers a few advantages over meters. For instance, it is possible to estimate power consumption at a granularity of 5000 Hz with a power model. In comparison, the built-in meters available in the two platforms we study operate at 60 Hz and 1 Hz. The high resolution offered by a power model exposes more power- and energy-saving opportunities to the runtime system.

Furthermore, in today's power-constrained HPC clusters where a node should quickly start operating at a lower frequency when the allotted power budget is exceeded, these power models are valuable. Using power models constructed for different operating frequencies, the runtime system could quickly determine a safe frequency in which a system should operate. Thus, power models offer a *one-shot* reconfiguration opportunity whereas using a power meter would mean a time-consuming, iterative approach. The successful deployment of Intel's RAPL power models [14] and associated power management schemes for CPU clusters clearly emphasizes the model-based approach's advantages [15]–[17]. In this work, we focus only on estimating the GPU power for a given voltage and frequency setting. Managing the power by shifting to a different frequency is beyond the scope of this work.

III. Background

In this section, we describe the hardware platforms, statistical techniques, and the benchmark applications used.

A. Hardware Platforms

Our hardware platforms include two high-end NVIDIA GPUs from different generations: (i) Fermi C2075 and (ii) Kepler K20c. The relevant details of these platforms are presented in Table I. Both these platforms are equipped with built-in sensors to measure power and temperature and have hardware counters to profile performance events (i.e., system activity). We reiterate that while power sensors help in measuring instantaneous power, the power models go further in enabling a number of energy management techniques as explained in Section II.

Measuring Power and Temperature: Power and temperature values reported by the built-in sensors can be accessed through the NVIDIA management library (NVML) interface [18]. This interface provides a C-based thread-safe API to monitor and manage the GPU. The corresponding methods to measure instantaneous power consumption

TABLE I: Hardware Details

Parameters	Fermi C2075	Kepler K20c
# CUDA cores	448	2496
# SMs	14	13
Core frequency	1150 MHz	706 MHz
Memory size	6GB	5GB
Memory type	GDDR5	GDDR5
Memory frequency	1.5 GHz	2.6 GHz
Memory bandwidth	144 GB/s	208 GB/s
Peak DP performance	515 GFlops	1170 GFlops
Total board power	215W	225W

and temperature are `nvmlDeviceGetPowerUsage` and `nvmlDeviceGetTemperature`, respectively.

Profiling Performance Events: CUDA profiling tools interface (CUPTI) is used to profile performance events by configuring and querying the hardware performance counters available in the NVIDIA GPUs [19]. There are 74 and 140 native performance events in C2075 and K20c, respectively. However, only a small fraction of these events (between one and eight, depending on the chosen events) can be profiled simultaneously.

B. Statistical Methods

Regression techniques are normally used to establish the relationship between two variables, a dependent variable y (also known as the *response*) and an independent variable x (also known as the *predictor*), through an unknown parameter β. In our experiments, the response variable is the power consumed and the predictors are the performance counters. Regression modeling involves finding the best value for β, given a modeling function f. Traditionally, the method of least squares is used to find the value of β.

Linear Models: In statistics, if the modeling function f is linear in β, then the model is considered linear. Thus, even if the relationship between the dependent variable y and the independent variable x is non-linear, a model falls under the category of linear models as long as β is linear. The linear models explored in this paper are described below.

Simple Linear Model (SLR): In this model, the response variable y, depends on a single predictor x. The basis function can be written as $y_i = \beta_0 + \beta_1 x_i + \epsilon$, where ϵ is the error term that cannot be modeled empirically.

Multiple Linear Model (MLR): In a linear model, if the dependent variable is related to more than one independent variable, then it is called an MLR model.

Interaction Effects: Interaction terms must be included in a model if two independent variables play a combined role on the dependent variable and their effects cannot be separated. Interaction effects are expressed through a third variable which is the product of the two independent variables. In this paper, we consider only second-order interaction effects (i.e., interaction between two variables only).

Quadratic Relationships: The MLR model may also include higher-order variables which indicates a non-linear relationship between the predictors and the response. In this paper, we evaluate quadratic models in which the response depends on the square of the predictors.

The mathematical basis functions of the MLR models evaluated in this paper are presented below:

Basic MLR model without interaction (MLR)

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon$$

Basic MLR model with interaction (MLR+I)

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_{12} x_{1i} x_{2i} + \epsilon$$

Quadratic model without interaction (QMLR)

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_{11} x_{1i}^2 + \beta_2 x_{2i} + \beta_{22} x_{2i}^2 + \epsilon$$

Quadratic model with interaction (QMLR+I)

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_{11} x_{1i}^2 + \beta_2 x_{2i} + \beta_{22} x_{2i}^2 + \beta_{12} x_{1i} x_{2i} + \epsilon$$

Stepwise Regression: Sometimes, too many independent variables are considered for modeling. To eliminate the unnecessary ones, we use a statistical technique known as *stepwise regression*. This technique alternates between two steps: (i) *forward selection* and (ii) *backward elimination*. During the forward selection step, the variable with the smallest *p-value*[1] is added to the model if this value is below a certain threshold. Intuitively, this variable explains the most variation in the modeling data. In the backward elimination step, among all variables added to the model, the one with the largest *p-value* is dropped if the value is above a certain threshold. This indicates that the variable dropped does not significantly affect the accuracy of the model. The algorithm terminates when there are no more variables to be added or dropped.

C. Applications

Statistical modeling involves a *training phase* in which the data is collected and the model is constructed and a *testing phase* in which the prediction accuracy of the model is evaluated. For our training phase, we chose workloads that exhibit a variety of computational and communication pattern representing a spectrum of application behavior. The *Level 1* applications (basic computational primitives) from the SHOC benchmark suite [20] was appropriate for this task. For the testing phase, we use applications from various benchmark suites including *Level 2* applications (full-fledged applications) from SHOC, select CUDA SDK samples, LLNL ASC proxy apps [21], and CUDA-equivalent SPEC ACCEL benchmarks [22] from Rodinia [23] and Parboil [24]. A brief description of the training and the testing applications is presented in Tables II and III, respectively.

IV. Methodology

In this section, we describe our data collection methodology, event selection technique, and model construction.

A. Data Collection

We modified all the applications to include a *profiling CPU thread* that periodically measures instantaneous power and board temperature via NVML and system activities via

[1] *p-value* is a statistical metric that indicates whether a variable truly has an effect on the response.

TABLE II: Training Applications

Benchmark	Description	Problem Size
Stencil2D	Standard two-dimensional nine-point stencil	4096x4096; 1000 *iter*; 100 *pass*
SpMV	Sparse matrix-vector multiplication	12288x12288; 250*pass*
FFT	Multiple two-dimensional fast Fourier transform	512-pt; 256MB; 1000 *pass*
GEMM	Single and double precision matrix multiplication	16KB sq. matrix; 50 *pass*
MD	Pairwise calculation of Lennard-Jones potential	36864 atoms; 20000 *pass*
Reduction	Sum reduction operation	64MB vector; 1000 *pass*
Triad	Streaming vector dot product computation	16MB data; 1500 *pass*
Scan	Parallel prefix sum algorithm	64MB data; 350 *pass*
Sort	Radix sort algorithm	96MB data; 20000 *pass*
BFS	Breadth-first search on an undirected graph	1000000 vertices; 1000 *pass*

TABLE III: Testing Applications

Benchmark	Source	Description	Problem Size
LULESH	LLNL	Unstructured explicit shock hydrodynamics problem using Lagrangian methods	90 edges
S3D	SHOC	Computes rates of chemical reactions across regular 3D grid	48/40 edges; 2500*pass*
QTC	SHOC	Quality threshold clustering algorithm	26x1024
FWT	SDK	Computes product of a square data set and matrix of basis vectors	128 (Data), 32M (Kernel)
Eigen	SDK	Computes eigen values for a given matrix	32768x32768
NW	Rodinia	Needleman-Wunsch - dynamic programming technique for aligning DNA sequences	16384 *seq*; 20 *pass*
Hotspot	Rodinia	Estimates processor temperature based on architectural floorplan and power measurements	2048x2048, 4(ht) ; 60000 *iter*
Histo	Parboil	Accumulates number of occurrences of each value	256Wx8192H
MRI-Q	Parboil	MRI image reconstruction from sampled radio responses	128x128x128
TPACF	Parboil	Two-Point Angular Correlation Function; measures distribution of massive bodies in space	48589 pts; 100 *pass*

CUPTI. This thread periodically looks up dedicated GPU hardware registers for measurement and does not interfere with normal GPU execution. The profiling interval was set to 20 ms for the C2075 GPU and 1000 ms for the K20c GPU considering the capabilities of the power measurement infrastructure of these systems.[2] The problem sizes for the different applications were chosen to ensure that we collect enough data points for model construction and testing. The relevant details for each application is shown in Tables II and III.

B. Event Selection

Selecting the right predictors in a regression model plays an important role in determining the accuracy of the model. While all relevant events could be included to maximize the accuracy, current GPUs have only a limited number of hardware counters to profile events. Furthermore, several pairs of events cannot be simultaneously profiled. This severely limits the number of events that can be included in the model.

From several tens of available events, we select a concise set to model power consumption. The selection is done in two phases. In the first phase, we consider the events in isolation to identify those exhibiting high correlation with power. In the second phase, we consider the events in concert, identifying non-redundant events that can be simultaneously profiled to address device limitations.

The steps involved in the first phase are listed below:

1) We collect performance counters and power drawn for each application-event pair by running the applications multiple times.
2) We compute the Pearson's correlation coefficient between performance counters and power for each application individually, and for all the applications collectively.
3) We eliminate the events showing a low overall correlation less than δ. The value for δ was set as 0.65 for C2075 and 0.55 for K20c by manually performing sensitivity analysis to maximize accuracy.
4) From the remaining set of highly correlating events, we eliminate those events that do not consistently show high correlation across applications.

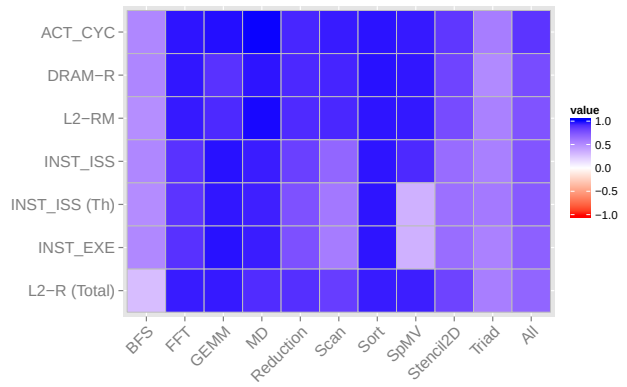
Events identified at the end of this phase are shown in the form of a correlation heatmap for the two GPUs under consideration in Fig. 2a and Fig. 2b.

Fig. 3 shows the algorithm used in phase two to determine events that are ultimately included in the power model. In this algorithm, the events are considered in decreasing order of overall correlation, with the highest correlating event selected first. For subsequent events, we check if they can be profiled simultaneously with the events already selected. If so, such events are included in the model only if they do not correlate with any of the events selected prior. As a result, we eliminate events that do not provide any new information.

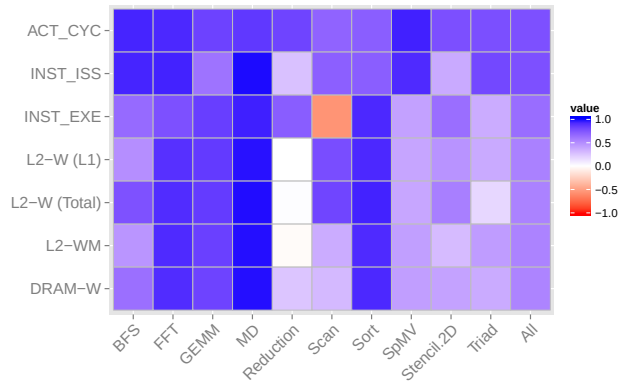
The performance counters identified at end of this phase for modeling power are described below:

- **ACT_CYC:** Number of cycles in which the GPU has at least one active warp.
- **DRAM-R:** Number of read requests sent to DRAM.
- **INST_ISS:** Number of instructions issued.
- **INST_EXE:** Number of instructions executed.

[2]We chose 20ms for C2075 versus 1000ms for K20c as the latter has issues with fine-grained power measurements as reported in [25].



(a) Correlation Heatmap for C2075



(b) Correlation Heatmap for K20c

Fig. 2: **Correlation Heatmap.** Performance counters showing the highest overall correlation with power (in decreasing order). Higher correlations are represented by darker shades and lower correlations are represented by lighter shades.

Input: E (Set of events showing high correlation)
Output: S (Set of events to be included in the model)
Algorithm
S ← Ø
for each event E_i (in decreasing order of correlation) in set E
  if E_i can be simultaneously profiled with events in Set S, **then**
    Calculate Pearson's correlation coefficient ρ_{ij} between E_i and all events S_j in Set S
    if $\rho_{ij} < \rho_{min}$ for all j, **then**
      S ← S ∪ E_i
    end if
  end if
end for

Fig. 3: Algorithm for event selection

- **L2-R:** Number of read requests sent to L2 cache.
- **L2-W(L1):** Number of write requests sent to L2 cache from L1 cache.

Among these, only `ACT_CYC`, `INST_ISS` and `INST_EXE` were selected for both the GPUs. The difference between the two GPUs is that reads (`DRAM−R`, `L2−R`) correlated with power on the C2075, whereas writes (`L2−W(L1)`) correlated with power on the K20. The predictors used for the two GPUs are shown in Table IV. We also consider portable models constructed using only those counters that show high correlation for both the GPUs under consideration. We consider these models to evaluate the scope of architecture-independent models.

TABLE IV: Predictors used in the models

Counter	C2075	K20c	Portable
ACT_CYC	+	+	+
DRAM-R	+		
INST_ISS	+	+	+
INST_EXE	+	+	+
L2-R	+		
L2-W (L1)		+	

C. Modeling

We construct five different models: simple linear regression (SLR), basic multiple linear regression (MLR), basic multiple linear regression with interaction (MLR+I), quadratic multiple linear regression (QMLR), and quadratic multiple linear regression with interaction (QMLR+I). We also explore the following approaches to construct the above five types of models.

Application-Independent Models: Two distinct sets of workloads are used for the training and the testing phase as described in Section III-C. From each workload used in the training phase, we collect 150 power and performance counter values to construct the model. This ensures adequate representation of each application in the model construction and avoids biasing the model towards longer running applications. We use the `lm` function available in the statistical software `R` to construct the model. The intercept values are fixed at 83000 mW and 42000 mW for the C2075 and K20c GPUs respectively. These values are the power consumed by the GPUs in their *active idle state*. The predictors of these models are the events identified in the *event selection* step. For the MLR models, we further refine the model by using stepwise regression to eliminate predictors that are not useful. To achieve this, we use the `stepAIC` function in `R`.

Application-Dependent Models: To increase the accuracy of the models, we explore application-dependent modeling. However, constructing such a model involves a one-time cost for each application. To reduce the cost, these models can be constructed using small-sized problems that run only for a few iterations.

Online Models: Our proposed technique for improving the accuracy while negating the drawbacks of application-dependent modeling involves the construction of a power model using the GPU's power sensors during the first few seconds of an application's execution. The power consumed by the rest of the application is predicted from the model thus constructed. The idea is to use a low-resolution power meter to construct a high-resolution power model for fine-grained power management. To test the feasibility of the approach, we study the sensitivity

(i.e., accuracy) of the model with respect to the number of samples (or data points), thereby determining the minimum number of samples required to construct these application-dependent models at runtime. A low number of samples indicates that the model can be constructed quickly within the first few seconds of an application's execution.

Temperature-Aware Models: An initial analysis of the predicted values on the workloads used in the training phase revealed that the accuracy steadily decreased with an application's execution time. This is because the long-running applications increased the GPU's temperature which in turn affected the leakage power. To study the relationship between temperature and power, we operated the GPU under different temperatures. We performed this study by measuring the *active idle* power *after* the execution of several stress workloads that increased the GPU's temperature. Fig. 4 shows this relationship for the C2075 GPU. As evident from this figure, idle power can be modeled as a linear function of temperature for a realistic operating range. Therefore, in order to accommodate the effect of the GPU temperature on power consumption, we add temperature as a linear predictor to the models under consideration.

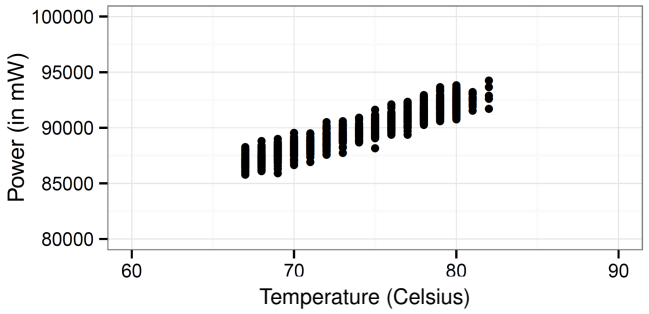


Fig. 4: Effect of temperature on idle power

V. Results

In this section, we present the accuracy of the various models in terms of mean absolute error percentage which is calculated as follows. For every time slice (20 ms for C2075 and 1000 ms for K20c), the absolute error percentage is calculated using the following equation:

$$\text{Error } \% = \frac{|\text{Estimated} - \text{Measured}|}{\text{Measured}} * 100$$

The mean absolute error for an application is then calculated by averaging the values obtained across time slices.

A. Application-Independent Models

Table V summarizes the results obtained for the various models on the target GPUs. The values presented in this table are the geometric mean error across all the test applications. We make two observations that hold true for both the GPUs: (i) *temperature-aware models consistently produce significantly higher accuracy compared to the basic models* and (ii) *linear models produce the highest overall accuracy with a mean error percentage of 4.49% on C2075 and 6.14% on K20c*.

TABLE V: Mean error % for application-independent models

Models	C2075		K20c	
	Basic	Temp-aware	Basic	Temp-aware
SLR	17.96	8.59	21.67	9.44
MLR	11.59	**4.49**	18.66	8.29
MLR+I	14.02	6.83	14.74	**6.14**
QMLR	14.83	6.42	15.46	7.82
QMLR+I	19.05	10.31	19.56	8.86

The coefficient terms for the best application-independent models (i.e., MLR for C2075 and MLR+I for K20c) are shown in Table VI. We observe that while modeling the interaction terms helped in improving the accuracy for K20c, their contribution towards overall power is small as indicated by their disproportionately smaller coefficients. According to these models, one degree Celsius increase in device temperature increased the power consumption by 0.4 W on the C2075 GPU and 0.58 W on the K20c GPU. This can be attributed to the difference in transistor sizes: 40 nm for C2075 and 28 nm for K20c. We note that the linear form of the equations and the parameters used are similar to the CPU power models explored in the past. This indicates the possibility of a generic power model for heterogeneous systems which is worth exploring in the future.

Next, we present the mean error percentage for the applications individually in Fig 5. We observe that even for the more accurate temperature-aware linear models, certain applications (e.g., *Eigen*, *TPACF*) exhibited high error. To understand the nature of this high error, we present the estimated and measured power profiles for *QTC* on C2075 and *Eigen* on K20c in Fig. 6 and Fig. 7, respectively. These applications were chosen for highlighting because they show both the positives and the negatives of the models simultaneously. In both the cases, we observe that the MLR model accurately estimates the phase shifts in power, but not the exact power values. If we subtract the estimated values by some constant offset, the error percentage drops dramatically, for example from 32% to 3% for Eigen on K20c. *Our results indicate that the application-independent models are robust predictors of power-phase shifts for the workloads under consideration.*

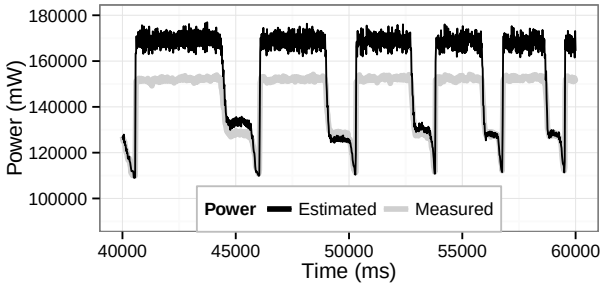


Fig. 6: **Application-Independent Models.** Estimated and measured power profiles for QTC on C2075.

TABLE VI: Coefficient values for the best application-independent models

Device	CYC	II	IE	DRAM-R	L2-R	L2-W	L2-W*IE
C2075	-3.62E-04	4.91E-04	1.54E-03	6.03E-03	1.22E-03	–	–
K20c	5.73E-05	-8.42E-05	2.70E-05	–	–	2.50E-05	9.54E-14

L2-W*II	L2-W*CYC	CYC*IE	CYC*II	II*IE	Constant	Temperature
–	–	–	–	–	8.30E+04	4.01E+02
8.23E-14	-1.41E-13	-3.99E-14	1.25E-13	-2.19E-15	4.20E+04	5.80E+02

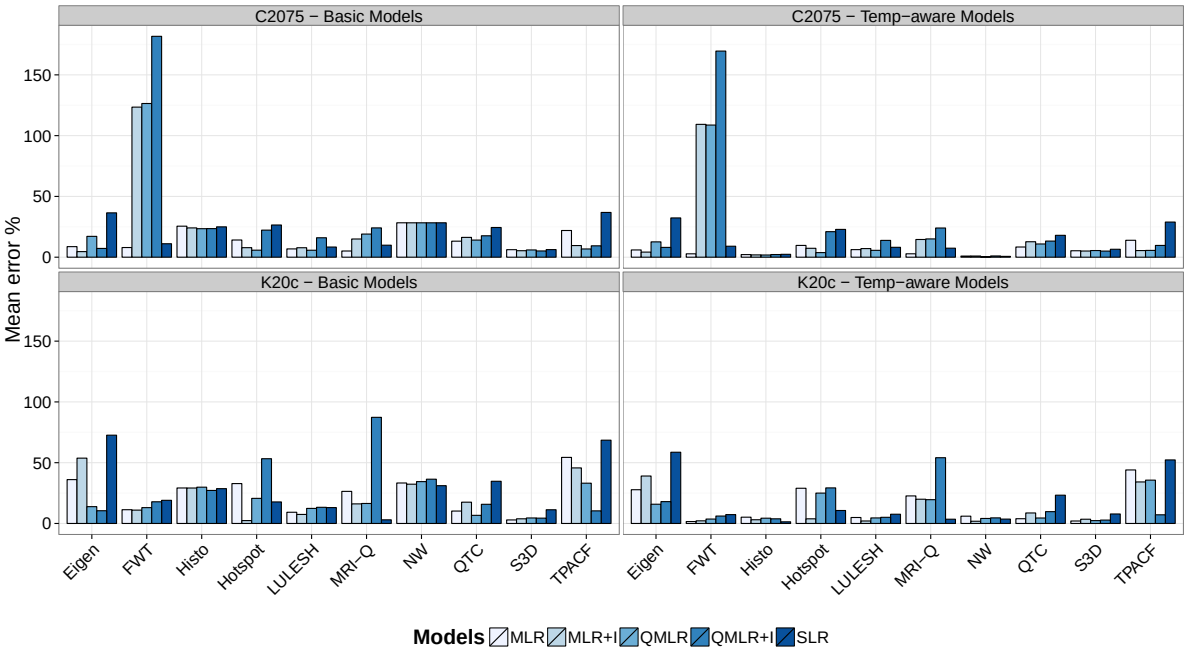


Fig. 5: **Application-Independent Models.** Mean error % for applications shown individually for all evaluated models.

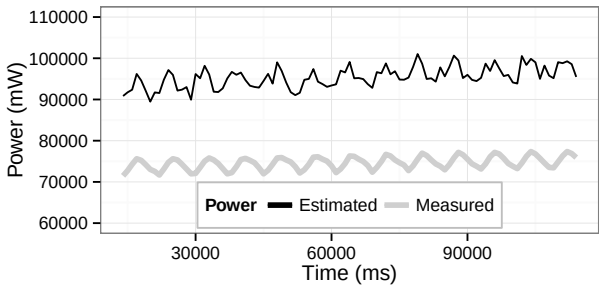


Fig. 7: **Application-Independent Models.** Estimated and measured power profiles for Eigen on K20c.

Cost of Portability: We evaluate portable models, by restricting the models to include only those events which have a high correlation with power consumption on both the GPUs under consideration. The error percentage achieved for these models is shown in Table VII. *We observe that the cost of portability is quite high: the error percentage for linear models increased from 4.49% to 8.22% on C2075 and from 6.14% to 9.40% on K20c*. This shows that even for successive generations of GPUs, portable models cannot be constructed without sacrificing accuracy.

TABLE VII: Mean error % for portable models

	C2075		K20c	
Models	**Basic**	**Temp-aware**	**Basic**	**Temp-aware**
SLR	17.96	8.59	21.67	9.44
MLR	18.55	8.22	23.36	8.48
MLR+I	18.26	11.15	22.84	9.40
QMLR	16.76	11.24	22.27	9.23
QMLR+I	18.36	11.63	20.87	8.79

Overhead: We observe an overhead of less than 0.1% when we profile up to 5 performance counters (the maximum used by our model) at a sampling frequency of 50 Hz. *Our experiments reveal that profiling the performance counters does not induce additional overheads on the GPU.*

B. Application-Dependent Models

In this section, we evaluate if there are benefits to using application-dependent models. *We observe that the application-dependent models exhibit higher accuracy than the application-independent models as shown in Table VIII and Table V; the mean error rate goes down from 4.49% to 1.02% for C2075 and from 6.14% to 0.88% for K20c*. Among the models evaluated, QMLR+I showed the highest accuracy. However, the difference in accuracy is insignificant when

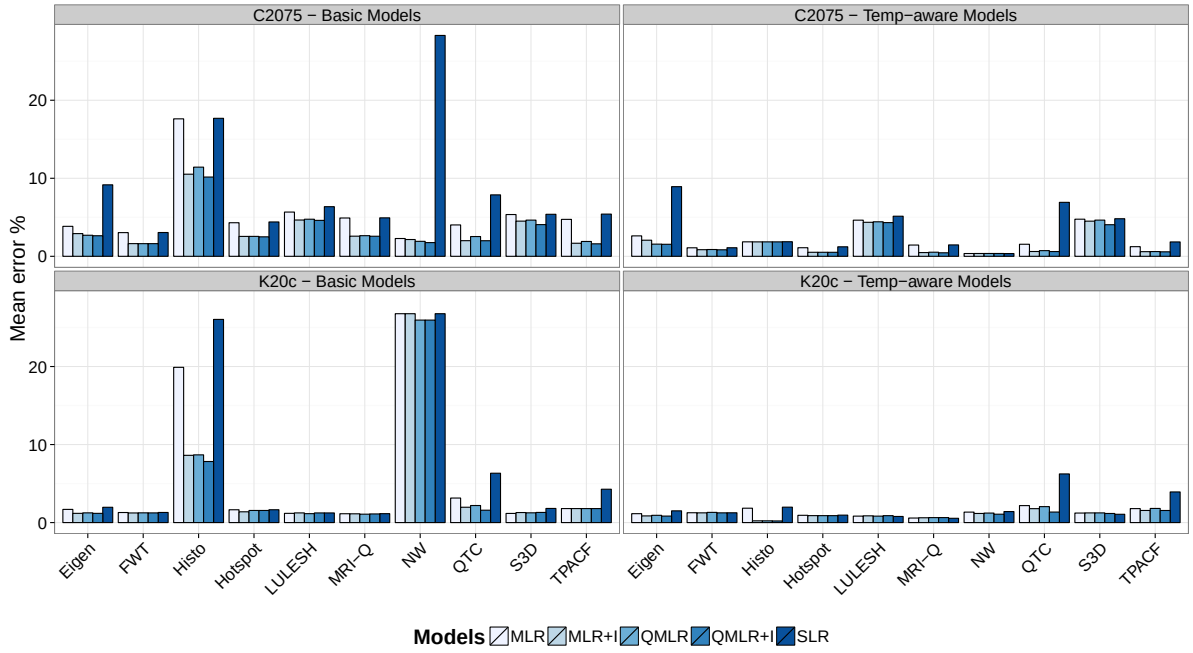


Fig. 8: **Application-Dependent Models.** Mean error % for applications shown individually for all evaluated models.

compared to the relatively simpler MLR+I model. Compared to application-independent models, these models are provided with only the most relevant information as their training data. This helps the more complex model in accurately estimating the power consumption and not merely phase shifts as shown in Fig. 9 and Fig. 10.

TABLE VIII: Mean error % for application-dependent models

Models	C2075		K20c	
	Basic	Temp-aware	Basic	Temp-aware
SLR	7.32	2.26	3.39	1.49
MLR	4.73	1.62	2.64	1.22
MLR+I	2.94	1.07	2.22	0.92
QMLR	3.04	1.08	2.24	0.96
QMLR+I	2.79	**1.02**	2.17	**0.88**

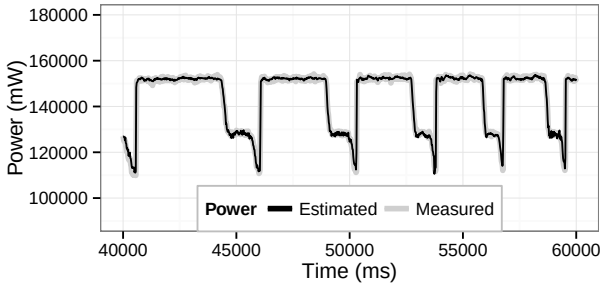


Fig. 9: **Application-Dependent Models.** Estimated and measured power profiles for QTC on C2075.

The mean error of the various power models is shown individually for each application in Fig. 8. In all the cases,

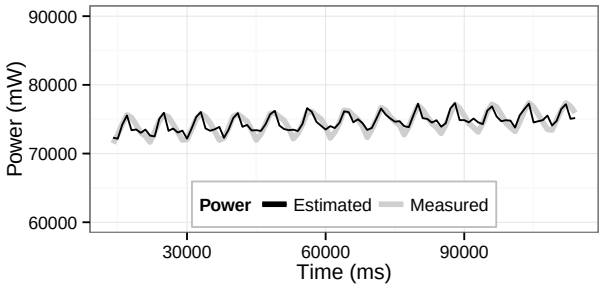


Fig. 10: **Application-Dependent Models.** Estimated and measured power profiles for Eigen on K20c.

the application-dependent models give significantly better accuracy compared to the application-independent models. However, applications such as LULESH and S3D show relatively poorer accuracy compared to the rest of the applications. This is because, these applications are composed of several computational kernels with distinct characteristics whereas the other applications are more homogeneous in nature. Such applications may benefit by modeling each computational kernel separately.

C. Constructing Power Models at Runtime

To achieve the high accuracy offered by the application-dependent models, we have to go through the process of constructing the model offline once for each application separately. This step can be avoided if we could construct the application-dependent models at runtime. However, for such models to be useful in a runtime system, model construction

should not introduce any significant overhead. This means only a few samples may be used to construct the model using simple techniques.

Sample-Size Sensitivity: We measured the sensitivity of MLR to the number of samples used for training. Fig. 11 shows the cumulative distribution function (CDF) plot for QTC on C2075. The x-axis represents error percentage and the y-axis represents the percentage of estimated values that falls below a given error percent during the testing phase. Models were constructed using the first 50, 100, 200, 400, and 800 samples and tested for the entire duration of the application (which consists of few thousand additional sample points). We observe that the accuracy of the MLR model improves only marginally when we use more than 100 samples.

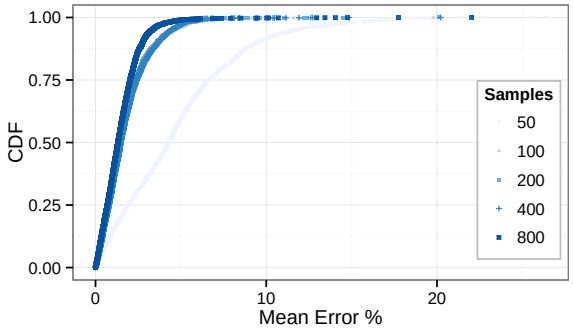


Fig. 11: Sample-size sensitivity for QTC

Fig. 12 shows sample-size sensitivity for the remaining applications. We observe that the different applications require different number of samples to accurately capture the characteristics of the application. In general, about 100 sample points are sufficient to construct an accurate model for scientific applications which produces no noticeable overhead. Therefore, *model construction is feasible at runtime as the overhead is minimal and the accuracy obtained is high*. However, for applications having heterogeneous characteristics, such models need to be adapted dynamically depending on the kernel being executed.

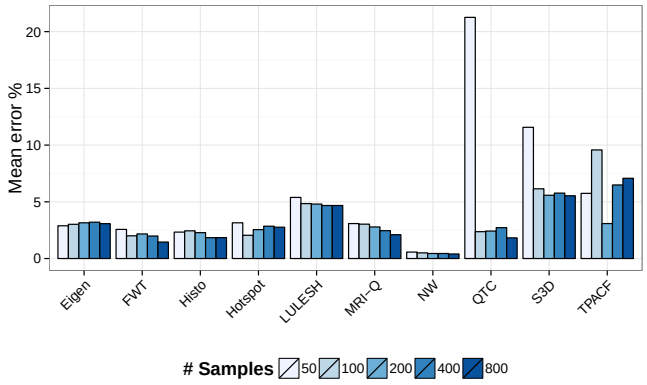


Fig. 12: Sample-size Sensitivity

VI. Related Work

In this section, we describe the related work in GPU power modeling and CPU power modeling that shaped our work.

GPU Power Modeling: Ma et al. were among the first to model power consumption of a graphics processor. They use support vector machines and achieve modest accuracy in predicting power [26]. However, these models are no longer applicable to general-purpose GPUs. Nagasaka et al. developed a linear regression-based model to accurately estimate the average power consumed by a GPU kernel on a GeForce 285 GTX GPU [7]. Abe et al. made the model architecture-agnostic but the accuracy was greatly diminished [11]. Song et al. present another model using artificial neural networks to estimate average power consumption [8]. Ghosh et al. have explored statistical techniques encapsulating non-linear relationship between power and performance event and have reported higher accuracy than the purely linear models [9]. Kasichayanula et al. estimate power of micro-architectural components on GPU using an empirical activity-based model [10]. However, all these works [7]–[11] use several counters requiring multiple application runs and can predict the power consumption only offline.

Hong and Kim present a detailed empirical model to estimate the power consumed by GPU [27]. Constructing such models require in-depth knowledge of the low-level micro-architecture, making it difficult to use in practice. GPUWattch [12] and PowerRed [13] are other GPU power modeling works, both of which require extensive architecture knowledge and low-level simulation. These power estimation techniques for simulators cannot be directly ported to real hardware that is available today due to the limitations of hardware counters.

CPU Power Modeling: Bellosa was the first to show the existence of a relationship between power consumption and performance events [28]. This discovery resulted in several works on instantaneous power prediction [29]–[31] and management [32], [33] for CPUs. In addition to adapting these techniques for GPUs appropriately, we also had to overcome limitations in GPU hardware profiling and software tools. Our proposals to address these limitations included application-specific and online modeling. We also systematically study several models similar to the work done by Rivoire et al. [34] and Davis et al. [35] for CPUs.

VII. Conclusion

In this work, we narrowed the knowledge gap between GPU power models and the existing literature on CPU power models. We found answers to previously unanswered questions regarding GPU power modeling. We identified system activities that correlate with power consumption of GPU systems. We found that apart from system activities, the device temperature plays a major role in determining the GPU's power consumption. We found that linear functions involving a few simple parameters are sufficient for highly accurate GPU power models. Then, we showed that application-dependent

models are highly accurate and useful. Finally, we showed that such models can be constructed at runtime with low overhead and high accuracy.

Acknowledgment

This work was supported in part by NSF I/UCRC IIP-0804155 and IIP-1266245 via the NSF Center for High-Performance Reconfigurable Computing.

References

[1] P. Kogge *et al.*, "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems," DARPA IPTO, Tech. Rep., Sep. 2008.

[2] J. Chen *et al.*, "Synergistic Challenges in Data-Intensive Science and Exascale Computing," DOE ASCAC, Tech. Rep., Mar. 2013.

[3] C. Hsu and W. Feng, "A Power-Aware Run-Time System for High-Performance Computing," in *Proceedings of the 2005 ACM/IEEE Conference on High Performance Networking, Computing, Storage and Analysis (SC)*, Nov. 2005, pp. 1–9.

[4] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: Making DVS Practical for Complex HPC Applications," in *Proceedings of the 23rd International Conference on Supercomputing (ICS)*. ACM, Jun. 2009, pp. 460–469.

[5] B. Subramaniam, W. Saunders, T. Scogland, and W. Feng, "Trends in Energy-Efficient Computing: A Perspective from the Green500," in *Proceedings of the 4th International Green Computing Conference (IGCC)*, Jun. 2013, pp. 1–8.

[6] "The Green500 List," http://www.green500.org/greenlists.

[7] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical Power Modeling of GPU Kernels Using Performance Counters," in *Proceedings of the 2010 International Green Computing Conference (IGCC)*. IEEE, Aug. 2010, pp. 115–122.

[8] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures," in *Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. IEEE, May 2013, pp. 673–686.

[9] S. Ghosh, S. Chandrasekaran, and B. Chapman, "Statistical Modeling of Power/Energy of Scientific Kernels on a Multi-GPU System," in *Proceedings of the 2013 International Green Computing Conference (IGCC)*. IEEE, Jun. 2013, pp. 1–6.

[10] K. Kasichayanula, D. Terpstra, P. Luszczek, S. Tomov, S. Moore, and G. D. Peterson, "Power Aware Computing on GPUs," in *Proceedings of the 2012 Symposium on Application Accelerators in High Performance Computing (SAAHPC)*. IEEE, Jul. 2012, pp. 64–73.

[11] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and Performance Characterization and Modeling of GPU-Accelerated Systems," in *28th IEEE International Parallel and Distributed Processing Symposium*, May 2014, pp. 113–122.

[12] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: Enabling Energy Optimizations in GPGPUs," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*. ACM, 2013, pp. 487–498.

[13] K. Ramani *et al.*, "PowerRed: A Flexible Modeling Framework for Power Efficiency Exploration in GPUs," in *Workshop on GPGPU*, 2007.

[14] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, Aug 2010, pp. 189–194.

[15] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing (ICS)*. ACM, 2013, pp. 173–182.

[16] P. Bailey, D. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. de Supinski, "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," in *43rd International Conference on Parallel Processing (ICPP)*, Sept 2014, pp. 371–380.

[17] T. Patki, D. K. Lowenthal, A. Sasidharan, M. Maiterth, B. L. Rountree, M. Schulz, and B. R. de Supinski, "Practical Resource Management in Power-Constrained, High Performance Computing," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. ACM, 2015, pp. 121–132.

[18] *NVML API Reference Manual*, Nvidia Corporation, Aug. 2013. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/5_5/rel/nvml/nvml.5.319.43.pdf

[19] *CUPTI: User's Guide*, Nvidia Corporation, Jul. 2013. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUPTI_Library.pdf

[20] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*. ACM, 2010, pp. 63–74.

[21] I. Karlin, J. Keasler, and R. Neely, "LULESH 2.0 Updates and Changes," Tech. Rep. LLNL-TR-641973, Aug. 2013.

[22] "SPEC ACCEL benchmark suite," http://www.spec.org/accel/.

[23] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–54.

[24] J. A. Stratton, C. Rodrigrues, I.-J. Sung, N. Obeid, L. Chang, G. Liu, and W.-M. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," University of Illinois at Urbana-Champaign, Urbana, Tech. Rep. IMPACT-12-01, Mar. 2012.

[25] M. Burtscher, I. Zecena, and Z. Zong, "Measuring GPU Power with the K20 Built-in Sensor," in *Proceedings of the 7th Workshop on General Purpose Processing on Graphics Processing Units (GPGPU)*. ACM, 2014, pp. 28–36.

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

[27] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*. ACM, Jun. 2010, pp. 280–289.

[28] F. Bellosa, "The Benefits of Event: Driven Energy Accounting in Power-sensitive Systems," in *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop*. ACM, 2000, pp. 37–42.

[29] G. Contreras and M. Martonosi, "Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, Aug. 2005, pp. 221–226.

[30] K. Singh, M. Bhadauria, and S. A. McKee, "Real Time Power Estimation and Thread Scheduling via Performance Counters," *SIGARCH Computer Architecture News*, vol. 37, no. 2, pp. 46–55, Jul. 2009.

[31] W. L. Bircher and L. K. John, "Complete System Power Estimation Using Processor Performance Events," *IEEE Transactions on Computer*, vol. 61, no. 4, pp. 563–577, Apr. 2012.

[32] C. Isci and M. Martonosi, "Phase Characterization for Power: Evaluating Control-flow-based and Event-counter-based Techniques," in *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Feb. 2006, pp. 121–132.

[33] C. Isci, G. Contreras, and M. Martonosi, "Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, Dec. 2006, pp. 359–370.

[34] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-level Full-system Power Models," in *Proceedings of the 2008 Workshop on Power Aware Computing and Systems (HotPower)*. USENIX, 2008.

[35] J. D. Davis, S. Rivoire, M. Goldszmidt, and E. K. Ardestani, "CHAOS: Composable Highly Accurate OS-based Power Models," in *Proceedings of the 2012 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, Nov. 2012, pp. 153–163.