Low-cost Estimation of Sub-system Power

Yuwen Sun, Lucas Wanner, Mani Srivastava
Department of Electrical Engineering
University of California Los Angeles
Los Angeles, USA
{sun831011, wanner, mbs}@ucla.edu

***Abstract*—Real-time, fine-grained power consumption information enables energy optimization and adaptation for both operating system (OS) and applications. Due to the high cost associated with dedicated power sensors, however, most computers do not have the ability to measure disaggregated power consumption at a component or subsystem level. We present DiPART (Disaggregated Power Analysis in Real Time), a tool to estimate subsystem power consumption based on performance (event) counters and a single, system-wide power sensor. With only one power sensor for overall system power consumption, DiPART is able to self-adapt to variations in subsystem power consumption present across nominally identical hardware. We validate the approach using a cluster of Intel Atom-based nodes that has been instrumented for subsystem power measurements (CPU, RAM and disk). DiPART was tested across nodes in the cluster using varied benchmarks, resulting in a 40% reduction in estimation error when compared to a static model.**

Keywords-linear model; adaptive model; power estimation; performance (event) counter; subsystem power disaggregation

I. INTRODUCTION

Power consumption is one of the leading concerns in both hardware [1] and software [2] design. Real-time knowledge of disaggregated power consumption is of particular importance for energy minimization and optimization in software, from applications to scheduling in operating systems (OS) [3]. Recent efforts have shown that knowledge of power consumed on a subsystem level, e.g. power consumed by CPU, memory, disk, and other peripherals can enable additional optimizations in applications such as storage encryption, virtualization, and application sandboxing, as well as exploration of application tradeoffs such as local computation vs. communication and computation offloading [4][5][6].

Several research efforts have explored methods to measure and estimate disaggregated power consumption efficiently. Two main categories of methods have emerged in this field: direct instrumentation and measurement with subsystem level power sensors and meters [2] [7], and indirect estimation based on co-related information, such as temperature [8][9] and performance counters [10][11][12][13][14][15]. Direct sensing provides a more accurate but very expensive method to measure subsystem power consumption, while the indirect estimation methods provide a cheaper alternative solution for subsystem power consumption estimation, especially in the case of performance counters, since neither additional hardware nor sensors are needed.

Performance counters are registers built into microprocessors, which act as counters for hardware-related and software-related events. The number and purpose of counters available change with CPU architecture, but counters typically account for events such as L1 cache misses, number of instructions executed, branch predictions missed, etc. Performance counters were initially introduced in the Intel Pentium architecture, as documented in [16].

Power estimation based on performance counters typically relies on a regression model that correlates the values for different performance counters with power consumption. However, due to variations across nominally identical hardware [6][17], power consumption characteristics of training and testing machines can be significantly different, leading to increased errors in estimation when transplanting a model to other (nominally identical) machines.

We built a tool, DiPART, for subsystem power consumption estimation based on a linear regression model with performance counters, and calibrated with a single system-wide power sensor. Through this calibration DiPART can be applied across different platforms, as shown in figure 1. We initially train the model using a vector of performance counters and power information available in an instrumented platform for the CPU, RAM, and disk subsystems.

We evaluate DiPART with a cluster of nominally identical nodes, instrumented for real-time subsystem level power consumption measurements. DiPART is of an adaptive mechanism to be re-trained on a testing machine without sensing the power consumption of each subsystem. With this adaptive mechanism, only the power consumption information for the overall system is needed, and the delta between the system power consumption is gradually compensated by the coefficients for each subsystem. Traces comparing measured and estimated power for each subsystem show that DiPART effectively compensates for deltas in subsystem power consumption across different machines.

Two scenarios were considered in this work: training and testing on the same machine (node); training and testing on the different machines (separate nodes). A significant difference in baseline power consumption is observed among nodes in cluster, which emphasizes the importance of the adaptability built in DiPART. We apply the DiPART across node for cross-validation, and show that the adaptive method yields a 40% reduction in estimation error when transplanting a model from one machine to another, comparing with a static model.

The rest of the paper is organized as follows: Section II presents the related work. Section III introduces the DiPART in each subsystem (CPU, RAM and disk) and overall system model. Section IV details the proposed system setup, including the test platform, including cluster, power measurement strategies, and estimation service. The detailed experimental results and discussion is presented in Section V, which includes training, testing and cross-validation. At last, section VI presents our conclusions and future work.

II. Related Work

Dynamic power management and optimization techniques can be used to minimize energy consumption, ensure battery lifetime, or control thermal dissipation [18][19][20][21][22]. To accomplish this, detailed knowledge about the power state of various system components is required. Due to variations in power across workloads, temperature, and individual devices, static models of power consumption (e.g. derived from device datasheets) do not provide a realistic estimation of actual power consumed.

A. *Subsystem power disaggregation*

Dedicated power sensors can be introduced in hardware to accurately track subsystem level power consumption. The LEAP system [5] introduces power sensors for different components in a sensing node, which are exposed to software through the runtime system. For our test platform, we use a design similar to the Atom-LEAP system [23]. Qualcomm's Snapdragon Mobile Development Platform incorporates embedded power sensors to monitor power consumption of various hardware blocks [24]. These sensors can be queried during run time to obtain detailed disaggregated power consumption and current measurement. Nevertheless, the cost of introducing multiple power sensors into hardware designs has limited the widespread application of this technique. In our work, we use a single system-wide power sensor to calibrate the subsystem level power estimation model (DiPART).

B. *Power estimation by performance counters*

Performance counters have been used as a predictor of total system power consumption [3][14][25], temperature [22], and subsystem (typically CPU) power [26][27][28]. Similarly, software states can also be used to predict power consumption [29][30]. Our work differs from these in that we (i) use a single, system-wide power sensor to calibrate subsystem level power estimations, and (ii) through the use of this sensor, we are able to adapt to variations in power found in nominally identical hardware [17][31].

Linear regression has been used to train the power consumption models, and most models were built using performance counters [3][32][33].

The effectiveness of model-based power characterization has been questioned in [6]. Because many systems are not energy proportional to work performed, many power estimation models are able report high percentage accuracy

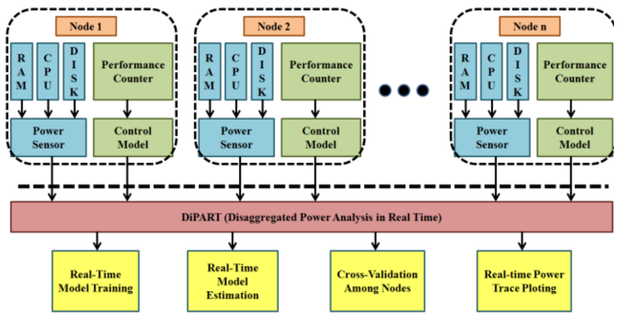
Node 1
R A M
C P U
D I S K
Performance Counter
Power Sensor
Control Model
Node 2
Node n
DiPART (Disaggregated Power Analysis in Real Time)
Real-Time Model Training
Real-Time Model Estimation
Cross-Validation Among Nodes
Real-time Power Trace Ploting

Fig. 1. The architecture of test platform.

simply due to a power baseline that is high compared with model errors. Our test platform suffers from poor energy proportionality, but we report errors in absolute terms. Furthermore, the main contribution of this work is a dynamic model adaptation scheme that handles the part-to-part power variability pointed out in [6].

III. Power Consumption Model

In this section we discuss the power estimation models for the CPU, RAM and disk subsystems. Total system power is estimated through a summation of all subsystems.

A. *CPU Power Consumption Model*

Ten performance counter variables were selected for the model of CPU power consumption, including Task counts, Context Switch counts, CPU Migration counts, Page Fault counts, Cycles counts, Instruction counts, Branches counts, Cache Refer counts, and Cache Miss counts. The performance counter vector, $V_{\text{Performance Counter}}$, combining these variables, is shown in E.3.1.

$$V_{\text{Performance Counter}} = \left\{ \begin{array}{c} Task\ Counts \\ Context\ Switch\ Counts \\ CPU\ Migration\ Counts \\ Page\ Fault\ Counts \\ Cycles\ Counts \\ Instruction\ Counts \\ Branches\ Counts \\ Branch\ Miss\ Counts \\ Cache\ Refer\ Counts \\ Cache\ Miss\ Counts \end{array} \right\} \qquad \text{E.3.1}$$

In order to capture most CPU activities, counters related to both software and hardware events were used, e.g. "Context Switch" counts software events (switching between processes) and "Cycles" counts hardware events (number of clock cycles).

The coefficients of each variable in $V_{\text{Performance Counter}}$ formed another vector $[\alpha,\ \beta \ldots \gamma]^{T}$, and the estimated power for CPU is the product of the performance counter variables vector and coefficients vector, plus a baseline constant $\lambda_{\text{constant CPU}}$. The linear model is shown in E.3.2.

$$P_{CPU} = [\alpha,\ \beta \ldots \gamma] \times V_{\text{Performance Counter}} + \lambda_{\text{constant CPU}} \qquad \text{E.3.2}$$

During the training phase, the coefficients and constant baseline value were obtained with a linear regression algorithm in Matlab, with the data of the performance counters and the monitored power consumption used as training inputs.

B. *Memory (RAM) Ppower Consumption Model*

Because the performance counters selected for CPU ($V_{\text{Performance Counter}}$) cover most aspects of software/hardware activity, they also capture events and activities related to power consumption in memory (RAM). However, for the RAM power consumption model, the coefficients vector ($[\Delta,\ \Gamma \ldots \Theta]^T$), as well as the constant baseline ($\lambda_{\text{constant RAM}}$) are different than the CPU model.

The RAM power consumption model is also linear and is shown in E.3.3. The coefficients vector and the constant baseline were obtained from the linear regression algorithm during the training phase, with the performance counter values and monitored power consumption used as training inputs.

$$P_{\text{RAM}} = [\Delta,\ \Gamma \ldots \Theta] \times V_{\text{Performance Counter}} + \lambda_{\text{constant RAM}} \quad \text{E.3.3}$$

C. *Disk Power Consumption Model*

Power consumption in storage systems is related to reading and writing, which result in disk I/O events. Therefore, instead of 10 performance counters used for CPU and RAM, two different variables are used for the disk model: Input Counts and Output counts. These counters correlated with I/O events to and from the disk. The vector of these variables, $W_{\text{Performance Counter}}$, is shown in E.3.4.

$$W_{\text{Performance Counter}} = \begin{Bmatrix} Input\ Counts \\ Output\ Counts \end{Bmatrix} \quad \text{E.3.4}$$

As with CPU and RAM, these variables are associated with a coefficient vector $[\varphi,\ \chi]^T$. The estimated power consumption for disk is shown in E.3.5.

$$P_{\text{Disk}} = [\varphi,\ \chi] \times W_{\text{Performance Counter}} + \lambda_{\text{constant Disk}} \quad \text{E.3.5}$$

During the training phase, the linear regression method was applied for obtaining the value of the coefficients, $[\varphi,\ \chi]$ and the constant baseline $\lambda_{\text{constant Disk}}$.

D. *System Power Consumption Model*

The system level power consumption model is a linear combination of the subsystem power consumption model, which is shown in E3.6.

$$\begin{aligned} P_{\text{system}} &= P_{\text{CPU}} + P_{\text{RAM}} + P_{\text{Disk}} \qquad \text{E3.6} \\ &= [\alpha,\ \beta \ldots \gamma] \times V_{\text{Performance Counter}} + \lambda_{\text{constant CPU}} \\ &\quad + [\Delta,\ \Gamma \ldots \Theta] \times V_{\text{Performance Counter}} + \lambda_{\text{constant RAM}} \\ &\quad + [\varphi,\ \chi] \times W_{\text{Performance Counter}} + \lambda_{\text{constant Disk}} \\ &= [\alpha + \Delta,\ \beta + \Gamma \ldots \gamma + \Theta] \times V_{\text{Performance Counter}} \\ &\quad + [\varphi,\ \chi] \times W_{\text{Performance Counter}} \\ &\quad + \lambda_{\text{constant}} \end{aligned}$$

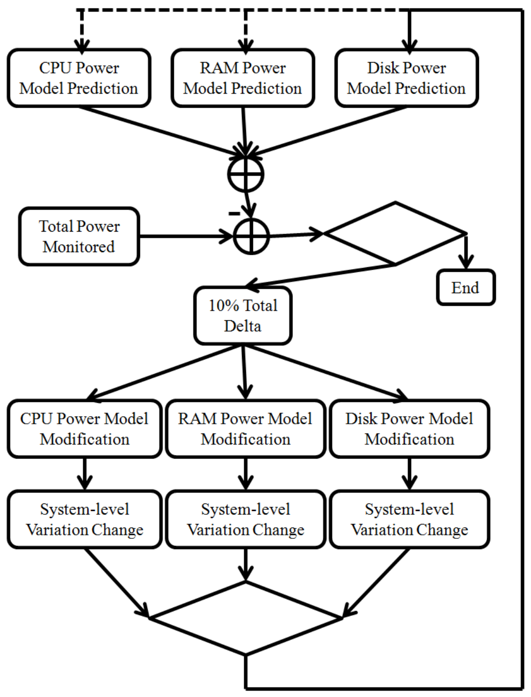
CPU Power Model Prediction
RAM Power Model Prediction
Disk Power Model Prediction
Total Power Monitored
-
End
10% Total Delta
CPU Power Model Modification
RAM Power Model Modification
Disk Power Model Modification
System-level Variation Change
System-level Variation Change
System-level Variation Change

Fig. 2. The Adaptive Model Mechanism.

$$= A \times V_{\text{Performance Counter}} + B \times W_{\text{Performance Counter}} + \lambda$$

After each subsystem level power consumption model has been trained, the system level power consumption model doesn't have to be trained again, when applying to the same machine. However, if applying the model into another machine, re-training is required. Even with the exactly same hardware and setup, the difference in power consumption is substantial, as will be discussed in the following sections.

E. *Error and Adaptive Model*

As long as functionally equivalent performance counters are available across systems, the power consumption model is transplantable from one system to another. In other words, the model can be trained in one (instrumented) machine and estimate on another one without subsystem power measurement instrumentation. However, because of variation in power consumption characteristics – which is present even in systems with the same identical hardware and software setup – directly applying the coefficients and constants from one training machine to another testing machine will result in high estimation errors. We observed these variations both in baseline power consumption as well as magnitude ratio for activity.

In order to estimate power with higher accuracy when transplanting the model across machines, re-training is needed. However, requiring power sensors for each subsystem for re-training would defeat the purpose of low-cost estimation mechanisms. We require only one, system-wide power sensor for re-training, as shown in Figure 2.

We start by calculating total estimated power consumption, and computing the delta between estimated total power consumption with measured total power. We then take a small portion of the delta (10%) and attempt to compensate this individually in the CPU, RAM, and disk models. Whichever compensation leads to the best improvement in shape fitting in estimated vs. modeled total power is chosen for this iteration. We continue this process until estimation error for total system power is smaller than some threshold, or a limit number of iterations is reached.

The percentage of the delta compensated in each iteration is adjustable, and larger percentage result in faster converge, but unstable training (oscillations), while smaller percentage results in slower converge The pseudo code for the adaptive mechanism is shown below.

```
#define V = $V_{\text{Performance Counter}}$
#define W = $W_{\text{Performance Counter}}$

Main(){
   Record data for a certain time period T;
   //Get power consumption for sublevel system
   $P\_CPU_{(k)} = [\alpha,\ \beta \ldots \gamma] \times V_{(k)} + \lambda_{\text{constant CPU}}$;
   $P\_RAM_{(k)} = [\Delta,\ \Gamma \ldots \Theta] \times V_{(k)} + \lambda_{\text{constant RAM}}$;
   $P\_Disk_{(k)} = [\varphi,\ \chi] \times W_{(k)} + \lambda_{\text{constant Disk}}$;

   //Calculated the energy consumed for sublevel system
   $En\_CPU = \sum_{k=0}^{T} (P\_CPU(k))$;
   $En\_RAM = \sum_{k=0}^{T} (P\_RAM(k)$;
   $En\_Disk = \sum_{k=0}^{T} (P\_Disk(k))$;

   For (i=0; i<Maximum iteration; i++){
      //Calculate the error of energy in system level
      Sys_Mod = En_CPU + En_RAM + En_Disk;
      Delta = Sys_Mod – system_power_monitored;

     //Model ratio change if compensate individually
      Rio_CPU =(En_CPU + Delta × 0.1)/En_CPU;
      Rio_RAM =(En_RAM + Delta × 0.1)/En_RAM;
      Rio_Disk =(En_Disk + Delta × 0.1)/En_Disk;

      //Calculate the error of system shape fitting
      $Err\_CPU = \sum_{k=0}^{T} ( system\_power\_monitored_{(k)} -$
          $Rio\_CPU \times P\_CPU_{(k)} - P\_RAM_{(k)} - P\_Disk_{(k)})^2$ ;
      $Err\_RAM = \sum_{k=0}^{T} ( system\_power\_monitored_{(k)} -$
          $Rio\_RAM \times P\_RAM_{(k)} - P\_CPU_{(k)} - P\_Disk_{(k)})^2$ ;
      $Err\_Disk = \sum_{k=0}^{T} ( system\_power\_monitored_{(k)} -$
          $Rio\_Disk \times P\_Disk_{(k)} - P\_RAM_{(k)} - P\_CPU_{(k)})^2$ ;

      //Compare Err_CPU, Err_RAM and Err_Disk;
      Find the one with smallest error (e.g. Err_CPU )
      Update coefficient and constant accordingly;
      (e.g. $[\alpha,\ \beta \ldots \gamma]_{new} = [\alpha,\ \beta \ldots \gamma] \times Rio\_CPU$,
          $\lambda_{\text{constant CPU}} = \lambda_{\text{constant CPU}} \times Rio\_CPU$ )

      Check Delta, exit if less than threshold;
      }
}
```

In each re-training iteration, the adaptive mechanism is able to explore which subsystem compensate the delta (difference between estimated and monitored power of system level), results in the best match between estimated and monitored power trace over a certain time period, and then adjust the subsystem model accordingly.

IV. Proposed Test Platform

For our training, testing and evaluation we use *Molecule* – a cluster of Intel Atoms. Each node in Molecule has been instrumented for CPU, memory (RAM), and disk power measurement, as in [5]. In this section we describe the Molecule environment.

A. *Intel Atom-based nodes*

Each Molecule node uses a dual-core Intel Atom D525 processor, with hyper-threading enabled, for a total of four virtual cores. Clock frequency is set to 1.8 GHz, with frequency scaling disabled. The processor has 1MB cache memory. Total RAM memory per node is 2GB, comprised by two 1GB DDR3 PC3-8500 memory DIMMs sourced from various vendors. For storage, we use an Intel 40GB X25-V solid-state drive.

Nodes run Ubuntu Linux Server version 11.04. Software is synchronized across the cluster, but each node maintains its own file system. For user authentication, we use the Network Information Service (NIS) protocol, with one of the nodes taking the server role. Likewise, one of the nodes exports a networked file system for data sharing across the cluster. Time is synchronized with the network time protocol (NTP). Other than NIS, NFS, NTP, and other essential kernel services, no resident software runs in the cluster. Hence, when there are no applications running, CPU utilization for all nodes is close to zero percent.

B. *DAQ Power Monitor*

To measure subsystem power, we introduce current sensing resistors (CSR) to relevant power lines, and measure voltage drop across these resistors to estimate current [5]. For the CPU, we introduce one 0.01Ω CSR to each of the power lines in the 4-pin ATX 12V CPU power connector. An identical resistor is used for the 5V SATA power line. For RAM, we use DDR3 risers with a 0.005 Ω CSR introduced in the 1.5V DIMM power supply line.

For measurement, we use National Instruments PCI-6254 data acquisition cards (DAQ). These cards are capable of acquiring 1 million samples per second across 16 differential channels. Two nodes in the cluster are equipped with one card each, and measurements are exported to all nodes through a shared networked file system. The Comedi control and measurement interface is used to interface with hardware [34]. The data acquisition software system occupies less than 1% CPU capacity.

C. *Sever and Data Acquisition*

The data acquisition (DAQ) system described above runs in two of the nodes, and provides real-time measured power for all nodes in the cluster through a shared file system. Each node periodically collects values for the relevant performance counters. The power measurement data synchronized and combined with the performance counter data, and exported in Json format to a remote server for re-training and evaluation.

The training server is set up on a remote PC, and receives data from the cluster and stores it locally for further processing. In order not to interfere with CPU, RAM, and disk utilization, the model runs on the server side. For estimation on the same machine, the model can be trained either online or offline, and the model estimated the subsystem power consumption in real-time. For the estimation across machines, the re-training was taken offline.

The DAQ monitoring system runs in the background to obtain the real-time power consumption for the overall system and each subsystem, including CPU, RAM and disk, for each node of the cluster. The DAQ acquisition frequency is set to 100Hz per channel. This data is then averaged for each 1-second period and sent to the estimation server in real-time. Performance counters are read and reset every second, and the data is sent to the server in real-time along with the measured power data.

V. Experiment Results

As measured power and performance counter data was collected in the background, we ran a series of benchmark applications, in order to generate different usage patterns for CPU, RAM and disk in the system. We trained the model with an initial set of power and performance counter traces, and then tested the system using new, independent runs of the same set of benchmarks on the same test machine. We then transplant the trained model to a new machine, and showed how our DiPART calibration adapts to variations in power consumption across hardware.

A. Benchmark and Applications

We use the following benchmark and applications for our training/testing experiments:

- UnixBench [35], a Linux benchmark suite with fundamental high-level features, integrating CPU, file I/O tests, and system-level various behaviors.
- CPU Burn-in [36] is a stability test program that attempts to drive core temperatures as high as possible.
- MD5 is used to verify 128-bit MD5 hashes of files, as a compact digital fingerprint of the file. MD5 requires intensive computation and I/O especially when the size of the verified file is large.
- GNU zip (gzip) [37] is a compression program that is popular in Linux operating system. Gzip is one of the test programs in SPEC CPU2000 Benchmark [38].
- The GNU Compiler Collection (GCC) [39] is a very commonly used compiler, which includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go. GCC is also one of the test programs in SPEC CPU2000 Benchmark [38].
- Monte Carlo Method is a statistical simulation method, which makes utilization of a series of random numbers to perform. A program of computing the value of Pi (ratio of Euclidean circle's circumference to diameter) by Monte Carlo Method is used.

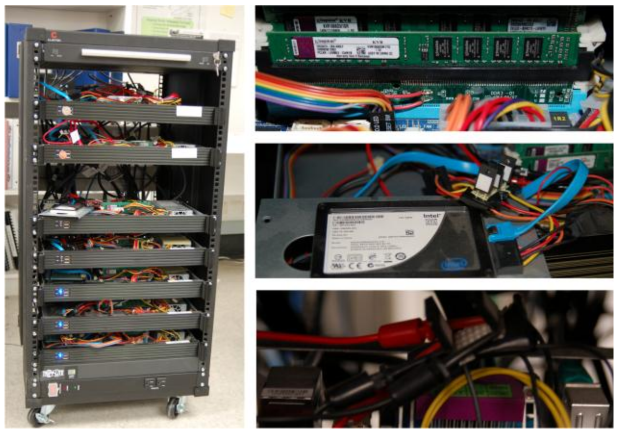
Fig. 3. The test platform (Cluster and power measurement for CPU, RAM and disk).

The construction and evaluation of DiPART for each subsystem, as well as the calibration/adaptation strategy, is explored in the following subsections.

B. Subsystem Power Models

Figures 4.a-d show traces of measured power consumption for CPU (blue lines), compared to power consumption estimated by the model (red lines), using the same data traces for training and testing for the selected benchmark and applications. This shows that the model is well trained. Figures 4.e-h show measured (blue) and estimated (red) power consumption for new, independent runs of the target benchmark applications for the selected benchmark and applications. This shows that the model accurately estimated power consumption for arbitrary applications. Table 1 shows minimum, maximum, and average estimation errors for CPU power with the testing dataset.

Figures 4.i-l show measured and estimated RAM power, using the same data traces for training and testing for the selected benchmark and applications. Figures 4.m-p show measured and estimated power traces for a new independent set of runs of the target applications for the selected benchmark and applications. Table 2 shows minimum, maximum, and average estimation errors for RAM power with the testing dataset.

Finally, Figures 4.q-r show measured and estimated disk power for the training dataset, and Figure 4.s-t show measured and estimated disk power with the testing dataset. For disk, we show only idle and active states, and not individual applications. We found no significant difference in disk power consumption across applications. Table 3 summarizes estimation errors for the disk subsystem.

C. Model Calibration and Self-Adaptation

Because of the aforementioned variations in power across hardware, transplanting a model trained with one machine to

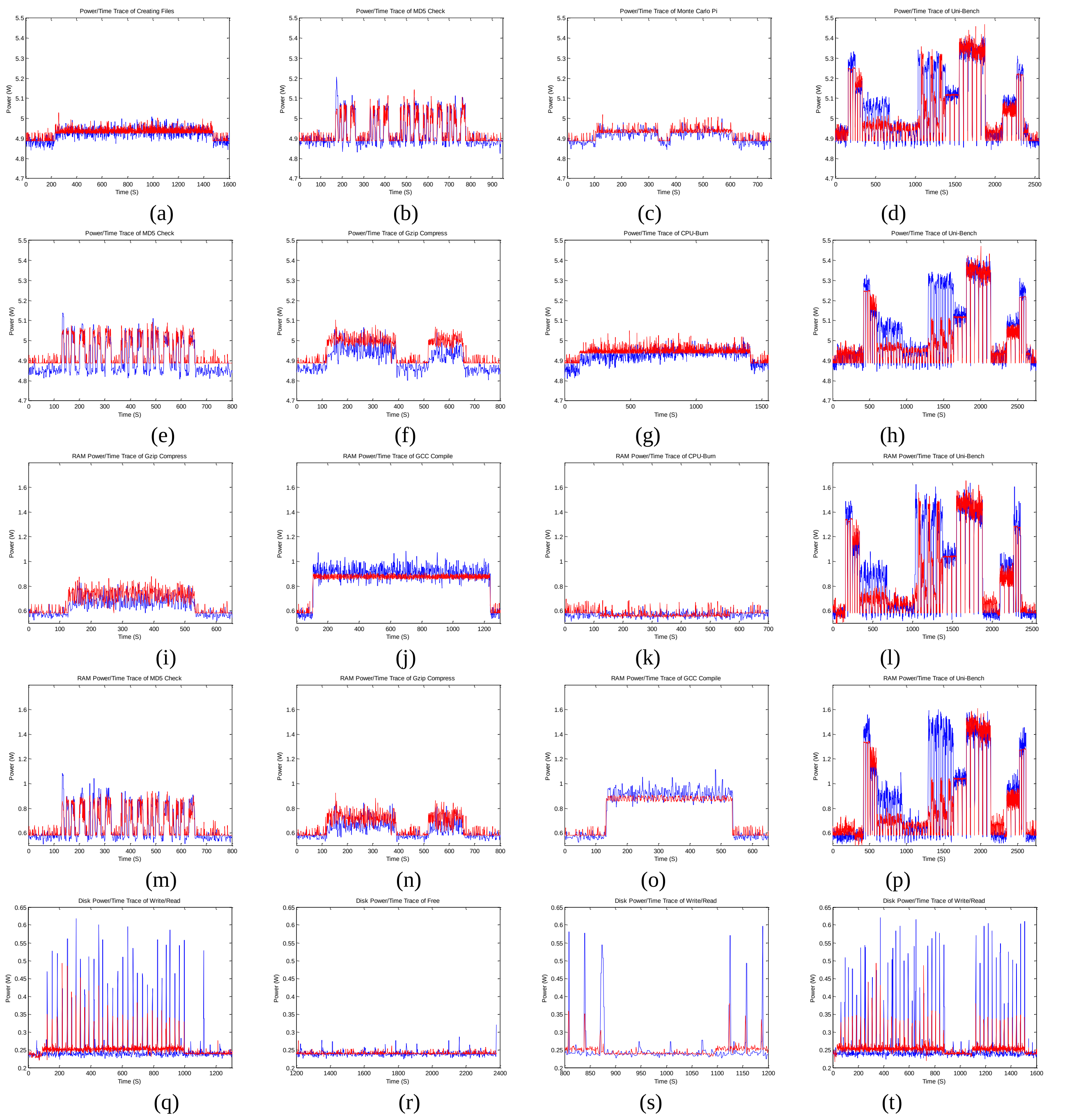

Fig. 4. (a)(b)(c)(d) are CPU power consumption model training result, Blue line is measured power and Red line is estimated power by model; (e)(f)(g)(h) are CPU power consumption model testing result, Blue line is measured power and Red line is estimated power by model; (i)(j)(k)(l) are RAM power consumption model training result, Blue line is measured power and Red line is estimated power by model; (m)(n)(o)(p) are RAM power consumption model testing result, Blue line is measured power and Red line is estimated power by model; (q)(r) are disk power consumption model training result, Blue line is measured power and Red line is estimated power by model; (s)(t) are disk power consumption model testing result, Blue line is measured power and Red line is estimated power by model; (a) Application of creating files; (b)(e)(m) Application of MD5 check; (j)(o) Application of GCC compiling; (c) Application of Monte Carlo Pi calculation; (d)(h)(l)(p) Uni-Benchmark suit; (f)(i)(n) Application of Gzip compressing; (g)(k) Application of CPU-Burn; (q)(t) Writing and reading event on disk; (r)(s) No event on disk.

a new nominally identical machine will result in high errors in estimation. This is illustrated in Figure 5. Figure 5.a-b shows the difference in CPU and disk measured power consumption between two nodes. Figure 5.c-d shows the estimation and measurement before and after the re-training and calibration phase is completed. Similarly, Figure 5.e-f

Table 1. CPU Model Estimation, DAQ Measurement and Error in Testing

CPU Test	Model Estimation / DAQ Measurement (Testing)						
	Time Duration (mins)	Model Estimation / DAQ Measurement			Error (Estimated - Measured)		
		Max Power (W)	Min Power (W)	Average Power (W)	Max Power (W)	Min Power (W)	Av. Power (W)
Creating Files	13.30	5.48 / 5.04	4.88 / 4.72	4.94 / 4.90	0.44	0.16	0.04
Md5 Check	8.33	5.10 / 5.10	4.88 / 4.80	4.96 / 4.94	0	0.08	0.02
Monte Carlo Pi	12.50	5.02 / 4.96	4.88 / 4.88	4.94 / 4.90	0.06	0	0.04
Gzip Compress	8.33	5.10 / 5.06	4.88 / 4.88	4.94 / 4.92	0.04	0	0.02
Gcc Compile	5.83	5.06 / 5.16	5.02 / 4.92	5.06 / 5.02	-0.1	0.1	0.04
CPU-Burn	20.83	5.04 / 5.02	4.92 / 4.86	4.96 / 4.94	0.02	0.06	0.02
Uni-Bench	40.83	5.46 / 5.42	4.88 / 4.84	5.04 / 5.08	0.04	0.04	-0.04

Table 2. RAM Model Estimation, DAQ Measurement and Error in Testing

RAM Test	Model Estimation / DAQ Measurement (Testing)						
	Time Duration (mins)	Model Estimation / DAQ Measurement			Error (Estimated - Measured)		
		Max Power (W)	Min Power (W)	Average Power (W)	Max Power (W)	Min Power (W)	Av. Power (W)
Creating Files	13.30	0.70 / 0.74	0.56 / 0.52	0.58 / 0.58	-0.04	0.04	0
Md5 Check	8.33	0.94 / 1.04	0.58 / 0.52	0.70 / 0.70	-0.1	0.06	0
Monte Carlo Pi	12.50	0.72 / 0.62	0.54 / 0.50	0.60 / 0.58	0.1	0.04	0.02
Gzip Compress	8.33	0.92 / 0.84	0.58 / 0.52	0.70 / 0.64	0.08	0.06	0.06
Gcc Compile	5.83	0.92 / 1.12	0.84 / 0.82	0.88 / 1.00	-0.2	0.02	-0.12
CPU-Burn	20.83	0.72 / 0.66	0.54 / 0.52	0.56 / 0.56	0.06	0.02	0
Uni-Bench	40.83	1.60 / 1.60	0.48 / 0.52	0.86 / 0.94	0	-0.04	-0.08

Table 3. Disk Model Estimation, DAQ Measurement and Error in Testing

Disk Test	Model Estimation / DAQ Measurement (Testing)						
	Time Duration (mins)	Model Estimation / DAQ Measurement			Error (Estimated - Measured)		
		Max Power (W)	Min Power (W)	Average Power (W)	Max Power (W)	Min Power (W)	Av. Power (W)
Write/Read	26.70	0.51 / 0.62	0.22 / 0.22	0.26 / 0.26	-0.11	0	0
Free	50.30	0.26 / 0.28	0.23 / 0.23	0.24 / 0.24	-0.02	0	0

compares measured and estimated RAM power before and after re-training, and Figure 5.g-h shows estimated and measured system level power consumption, before and after the re-training process is completed.

The error in percentage of each subsystem level model is shown in figure 5.i-p, during the adaptive process. In each iteration of the adaptive mechanism of DiPART introduced in section 3.E, the difference of the model estimated power and measured power was shown in figure 5.i-p, and the error approaching to 0 as the iterations increase. In the first case shown in figure 5.i-l, the model was trained in node 1 and tested on node 2, and the estimating power larger than the measured power. In first 15 iterations, the delta compensated RAM model first, and followed by CPU model in the next 10 iterations, and compensated disk model at last in the following 70 iterations. Also, the estimated and measured power consumption traces were shown in figure 5.c, d, g, h for both before and after adaptive re-training. From the figure, it shows the algorithm compensate the difference, which resulted in a 40% decrease in error. Similar in second case shown in figure 5.m-p, the model was trained in node 2 and tested on node 3, and the estimating power was less than the measured power. In first 7 iterations, the delta compensated RAM model first, and followed by disk model in the next 50 iterations. In this case, CPU model was not compensated.

D. Discussion

The models estimated each subsystem (CPU, RAM and disk) very well, and the trace of the power consumption followed the measured value, according to the application and benchmarks running on nodes, as shown in figure 4. From the figure, it is clear that applications have different maximum power, as well as different shape of the power consumption trace, since the different function units and orders they took. As pointed out by [6], the baseline power consumption is very large and difficult to remove, which is also verified in this paper. However, even with a large baseline, DiPART is still able to detect changes is power consumption.

For our training and testing process, more than 500 hours data were collected from test platform, and each application and benchmark was executed more than 20 times. The data was then randomly divided into 10 groups, each of which contains at least one run of every benchmark. The models were trained and tested by 10 fold cross validation method. From the results summarized in Tables 1-3, we see upwards of 15% (0.7W) increase for CPU power consumption

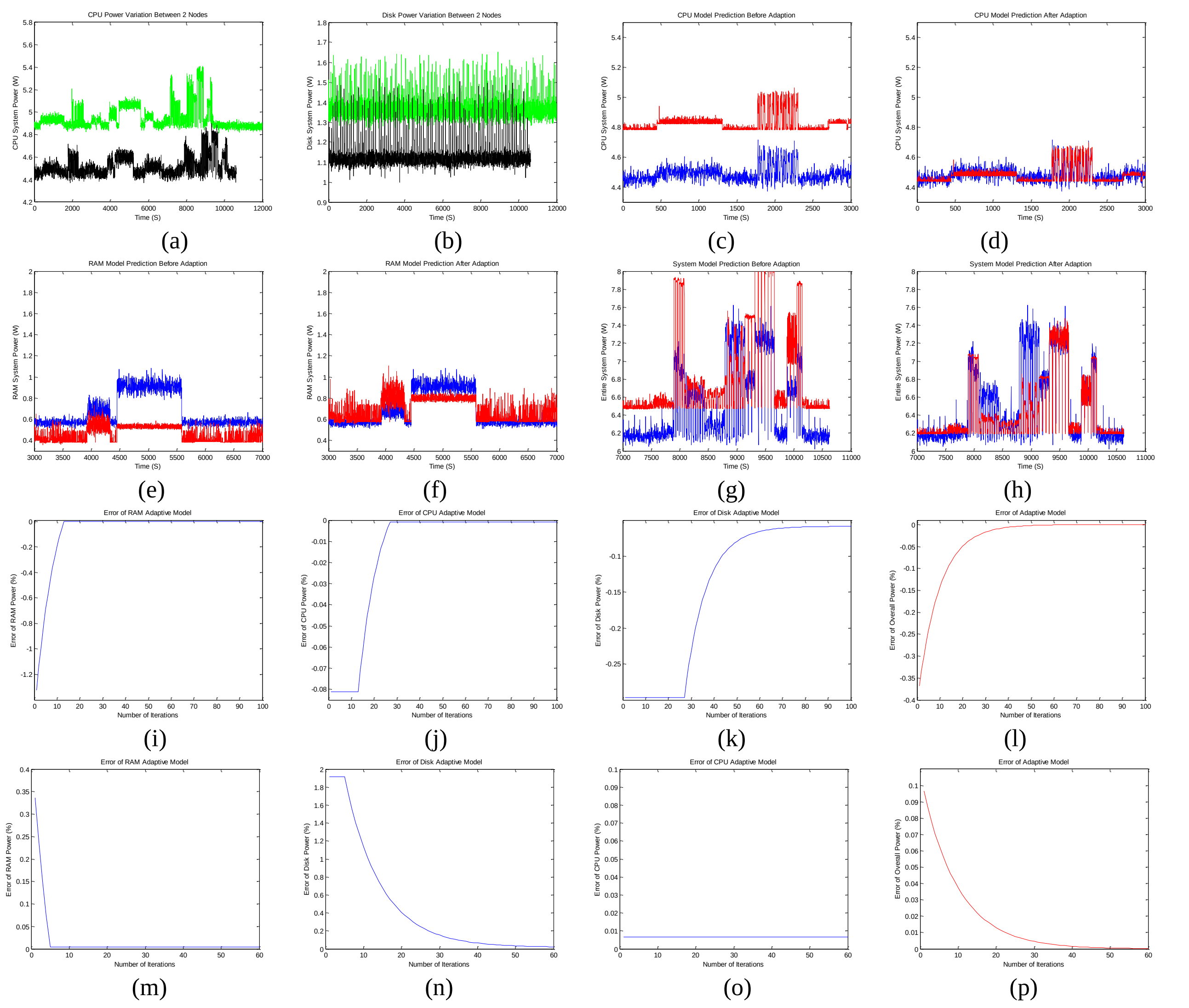

Fig. 5. (a) The CPU baseline difference between 2 nodes; (b) The disk baseline difference between 2 nodes; (c)(g) CPU power consumption model cross-validation trained on node 1 and tested on node 2, before adaption, the estimating in Red and measuring in Blue; (d)(h) CPU power consumption model cross-validation trained on node 1 and tested on node 2, after adaption, the estimating in Red and measuring in Blue; (e) RAM power consumption model cross-validation trained on node 2 and tested on node 3, before adaption, the estimating in Red and measuring in Blue; (f) RAM power consumption model cross-validation trained on node 1 and tested on node 2, after adaption, the estimating in Red and measuring in Blue; (i)(m) Error of RAM model during adaption process; (j)(o) Error of CPU model during adaption process; (k)(n) Error of disk model during adaption process; (l)(p) Error of system level power during adaption process; (i)(j)(k)(l) Model trained on node 1 and tested on node 2; (m)(n)(o)(p) Model trained on node 2 and tested on node3.

between the minimum and maximum power (due to the high baseline of CPU), a 220% (1.1W) increase for the RAM, and a 182% (0.4W) increase for the disk. Comparing the average power consumption between the measured and model estimated value, CPU model obtained an error less than 0.04W, RAM model obtained the error less than 0.12W, and disk model obtained error equal to 0. The accuracy of average power resulted in the accuracy estimating energy consumption. The comparison of maximum and minimum power is also important, since it shows how well the estimating trace follows the measured one, and the difference statistically shows the dynamic change of the power consumption.

The adaption feature of the DiPART is fundamental, due to the variations among different platform even with the exactly same hardware, which has been shown in figure 5.a-b. The baseline power of the 2 nodes was 4.4W and 4.9W for CPU, 1.1W and 1.35W for Disk respectively, therefore when applying the model trained in node 1 and tested on node 2, the error between model estimation and measurement value were large before the adaptive re-training, which a difference in baseline is obvious in figure 5.c, g. After re-training the model by the adaptive method, the baseline difference was eliminated, as shown in figure 5.d, h.

VI. Summary And Future Directions

In this paper, we built DiPART, a tool for low-cost real-time disaggregated power measurement. Our tool uses linear models for estimating subsystem level power consumption based on performance counters, and calibrated by a single, system-wide power sensor. We used a platform instrumented for subsystem power monitoring to train and validate our model. We showed that the model accurately estimates power consumption, and adapts to variations in power when transplanted to new hardware.

When tested on the same machine used for training, the model obtained high estimation accuracy (less than 0.03W for system). When transplanted to a new machine, the adaptation mechanism was able to reduce estimation errors by more than 40%.

We show that the performance counters model can be accurate and reliable and an adaptive mechanism can effectively compensate for variations in power present across platforms. In the future, other subsystems besides CPU, memory (RAM), and storage (disk) will be considered. In principle, the adaptive disaggregation mechanism can be implemented in a platform with a larger number of subsystems; however, the accuracy and the training speed will be decreased accordingly.

A linear model was implemented in this paper and more sophisticated models will be implemented in future work for increased accuracy. We will explore the tradeoff between accuracy and complexity in more sophisticated models.

VII. Acknowledgments

This material is based upon work supported by the NSF under awards #CNS-0905580 and CCF-1029030. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. Lucas Wanner is supported in part by CAPES/Fulbright grant #1892/07-0.

References

[1] T.N. Mudge, “Power: Afirst class design constraint for future architecture and automation,” in High Performance Computing (HiPC), pp. 215-224, 2000.

[2] V. Tiwari, S. Malik, and A. Wolfe, “Power analysis of embedded software: A first step towards software power minimization,” IEEE Transactions on Very Large Scale Integration (L\VLSI) System, Vol. 2, No. 4, pp. 437-445, Dec 1994.

[3] K. Singh, M. Bhadauria and S.A. Mckee, “Real time power estimation and thread scheduling via performance counters,” ACM SIGARCH Computer Architecture News, Vol. 37, No. 2, pp. 46-55, May 2009.

[4] P. A. H. Peterson, D. Singh, W. J. Kaiser, and P. L. Reiher, “Investigating energy and security trade-offs in the classroom with the atom LEAP testbed,” In Proceedings of the 4th conference on Cyber security experimentation and test (CSET'11). USENIX Association, Berkeley, CA, USA, Nov, 2011.

[5] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. J. Kaiser, “The low power energy aware processing (LEAP) embedded networked sensor system,” In Proceedings of the 5th international conference on Information processing in sensor networks (IPSN '06), pp. 449-457, New York, NY, 2006.

[6] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, “Evaluating the effectiveness of model-based power characterization,” In Proceedings of the 2011 USENIX conference on USENIX annual technical conference (USENIXATC'11), 2011.

[7] J. Russell and M. Jacome, “Software power estimation and optimization for high performance, 32-bit embedded processors,” In Proceedings of the International Conference on Computer Design, October 1998.

[8] H. Li, P. Liu, Z. Qi, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, “Effcient thermal simulation for run-time temperature tracking and management,” In International Conference on Computer Design, pp. 130-133, 2005.

[9] P. Liu, Z. Qi, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, “Fast thermal simulation for architecture level dynamic thermal management,” In IEEE/ACM International Conference on Computer Aided Design, pp. 639-644, 2005.

[10] F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems,” in ACM SIGOPS European Workshop, September 2000.

[11] C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: methodology and empirical data,” in International Symposium on Microarchitecture, December 2003.

[12] W. L. Bircher, M. Valluri, J. Law, and L. John, “Runtime identification of microprocessor energy saving opportunities,” in International Symposium on Low Power Electronics and Design, pp 275-280, August 2005.

[13] P. Ranganathan, P. Leech, D. Irwin and J. Chase, “Ensemble-level power management for dense blade servers,” in International Symposium on Computer Architecture, June 2006.

[14] T. Li and L. John, “Run-yime modeling and estimation of operating system power consumption,” in Conference on Measurement and Modeling of Computer Systems, June 2003.

[15] K. Lee and K. Skadron, “Using performance counters for runtime temperature sensing in high performance processors,” in High-Performance Power-Aware Computing, April 2005

[16] T. Mathisen “Pentium Secrets,” Oct 14th, 1999. Aviable at http://www.gamedev.net/page/resources/_/technical/general-programming/pentium-secrets-r213?

[17] L. F Wanner, R. Balani, S. Zahedi, C. Apte, P. Gupta, M. B Srivastava, “Variability-aware duty cycle scheduling in long running embedded sensing systems,” *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*.

[18] T. Chen, J. Huang, L. Xiang, and Q. Shi, “Dynamic power management framework for multi-core portable embedded system,” In *Proceedings of the 1st international forum on Next-generation multicore/manycore technologies* (IFMT '08). ACM, New York, NY.

[19] V. Devadas and H. Aydin, “On the interplay of dynamic voltage scaling and dynamic power management in real-time embedded applications,” In *Proceedings of the 8th ACM international conference on Embedded software* (EMSOFT '08). pp. 99-108, New York, NY, 2008.

[20] R. Jejurikar and R. Gupta, “Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems,” In *Proceedings of the 2004 international symposium on Low power electronics and design* (ISLPED '04), pp. 78-81, New York, NY, 2004.

[21] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, “Battery lifetime prediction for energy-aware computing,” In *Proceedings of the 2002 international symposium on Low power electronics and design* (ISLPED '02), pp.154-159, New York, NY, 2002.

[22] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner, “Event-Driven Energy Accounting for Dynamic Thermal Management,” In Proceedings of COLP, 2003.

[23] D. Singh and W. J. Kaiser, “The Atom LEAP Platform For Energy-Efficient Embedded Computing,” UC Los Angeles: Actuated,

Sensing, Coordinated and Embedded Networked Technologies (ASCENT). Technical Report, 2010.

[24] Qualcomm. Snapdragon MSM8660 Mobile Development Platform. Available at https://developer.qualcomm.com/

[25] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," In *Proceedings of the 2001 international symposium on Low power electronics and design* (ISLPED '01), pp. 135-140, New York, NY, USA, 2001.

[26] Y. Cho, Y. Kim, S. Park, and N. Chang, "System-level power estimation using an on-chip bus performance monitoring unit," In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08), pp. 149-154, Piscataway, NJ, 2008.

[27] 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 '05), pp. 221-226, New York, NY, 2005.

[28] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," In *Proceedings of the 24th ACM International Conference on Supercomputing* (ICS '10), pp. 147-158, New York, NY, USA, 147-158, 2010.

[29] A. Hoeller Jr. and A. A. Fröhlich, "On the Monitoring of System-Level Energy Consumption of Battery-Powered Embedded Systems," In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2608-2613, Anchorage, AK, USA, 2011.

[30] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis* (CODES/ISSS '10), pp. 105-114, New York, NY, USA, 2010.

[31] D. Bull, S. Das, K. Shivashankar, G.S. Dasika, K. Flautner and D. Blaauw, "A Power-Efficient 32 bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation," *IEEE Journal of Solid-State Circuits*, vol.46, no.1, pp.18-31, Jan. 2011.

[32] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," In Proceedings of the 1st ACM Symposium on Cloud computing (SOCC '10). ACM, 2010.

[33] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-Level Full-System Power Models," In HotPower, San Diego, December 2008.

[34] Comedi: Linux Control and Measurement Device Interface. Available at http://www.comedi.org/

[35] UnixBenchmark Suit, available at http://www.tux.org/

[36] CPU Burn-in application, available at http://www.cpuburnin.com

[37] Gzip application, available at http://www.gzip.org/

[38] SPEC CPU2000 Benchmark Suit, available at http://www.spec.org/cpu2000/CINT2000/164.gzip/docs/164.gzip.html

[39] GNU Compiler Collection (GCC), available at http://gcc.gnu.org/