2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing

Estimating the Energy Consumption of Executing Software Processes

Vivek Kumar Singh
Department of Information Systems
National University of Singapore
Singapore
Email: viveks@comp.nus.edu.sg

Kaushik Dutta
Department of Information Systems
National University of Singapore
Singapore
Email: duttak@nus.edu.sg

Debra VanderMeer
Decision Sciences and Information Systems
Florida International University
Miami, FL USA
Email: debra.vandermeer@fiu.edu

***Abstract*—Power consumption in data centers is significant across the globe. The use of cloud-based services, e.g., infrastructure as a service and software as a service (such as Google Docs, Microsoft Office 365, Salesforce.com), is becoming a standard practice in modern IT frameworks. This paradigm shift in the IT industry indicates that the demand for data-center-based services will continue to increase in the future, with concomitant increases in power consumption. In such a scenario, optimizing the IT resources to improve energy efficiency is a necessity. The first step of such an optimization at the application level is knowing how much energy an application is consuming. One of the main challenges in this domain is developing a software-based energy metering tool that can measure an OS processes' energy consumption. Many existing solutions depend on an external watt-meter or other hardware-based enhancements; these are not practical for real-world use in data centers. To overcome the limitations of existing solutions, we have developed an OS process-level power metering tool that can accurately estimate the energy usage of each OS process running on a Linux server without an online watt-meter. Based on a set of experiments, we demonstrated that our method and implementation provides energy consumption estimation for complex e-business applications with above 95% accuracy.**

***Keywords*-Energy, Power Meter, Green IT, Measurement, Modeling**

I. Introduction

IT power consumption is a significant concern. Recent estimates show that power usage by data centers worldwide ranged from 1.1% to 1.5% of all worldwide power consumption in 2010 [1] (roughly 332 billion kWh [2]). In the broad field of technology, the issue of energy efficiency is receiving substantial attention along multiple fronts, from data center design to hardware design. Data center designers have worked to improve the infrastructure within data centers to improve the efficiency of cooling systems and power delivery [3], [4], while processor designers have proposed chip designs that adjust power use based on workloads [5], and memory designers have improved energy usage by employing multi-level caches (L1 and L2 cache) for frequently accessed datasets in memory [6].

All these solutions impact the IT infrastructure, but do not consider the workload subjected to it, i.e., they do not address the scope of the workload generated by the software that runs on it. Recently, the Information Systems community has called for a broad research focus on improving IT energy efficiency [7]. In line with this call, energy efficiency needs to be addressed not only in hardware, but also in software systems as well.

The ability to measure the power consumption associated with software applications is an important step towards meeting this objective. Capra and Merlo [7] cite several major goals on the path toward energy efficient information systems: (1) the identification and operationalization of proxy metrics for computational complexity and thermodynamic depth of specific software applications; (2) the development and implementation of a tool capable of measuring these metrics; and (3) the measurement of actual power consumption by applications using such a tool. In this work, we propose a method of measuring the energy consumption of operating system (OS) processes, and present a software implementation of it as well as a set of experimental results that demonstrate the accuracy of our method.

The task of measuring software energy consumption may seem similar to measuring the usage of resources such as memory and CPU by OS processes, and so may seem simple at first glance. However, OS kernel APIs do not directly measure and make available energy consumption data for OS processes.

In the absence of any direct OS level API, researchers have resorted to both direct [7] and indirect [8] measurement approaches. While these current methods represent substantial advances in energy metering for software, they share a set of two properties that preclude usage in most real-world settings: (1) both methods require constant hardware presence (usage of a watt-meter [9], and in one case further hardware enhancements [7]) to gather energy usage data; and (2) both methods consider only the CPU's energy consumption, rather than the entire set of hardware resources.

In this research, we propose and develop a tool to measure process-level energy consumption. Our method incorporates all aspects of an OS process's energy usage, including CPU, memory, I/O and network communication. Further, our approach requires a watt-meter only during the calibration phase; at runtime, our method requires no additional hardware. Once our tool has been calibrated on a set of hardware, our method calls for a software implementation to estimate process-level energy usage.

Our specific contributions include:

978-0-7695-5046-6/13 $26.00 © 2013 IEEE
DOI 10.1109/GreenCom-iThings-CPSCom.2013.40

1) Developing an approach for estimating process-level energy consumption that does not require continuous use of additional devices (such as a watt-meter or other hardware enhancement).
2) Identifying the relevant system parameters that determine energy consumption.
3) Developing an implementation of our approach to demonstrate the practical applicability of our method.
4) Demonstrating that our method is accurate for both overall energy usage on a system, as well as individual operating system processes.

In Section II, we discuss related work. In Sections III and IV, we describe our solution approach and the details of our method, respectively. In Section V, we describe our experimental results demonstrating the accuracy of our approach. In Section VI, we conclude this article and describe our plans for future work.

II. Related Work

In this section, we review work in the literature related to our work, including both implementations for resource monitoring and models for resource utilization.

In terms of monitoring tools, virtually every major operating system provides basic tools for tracking resource usage details for system components, e.g., CPU, memory, disk or network resources. For example, on the Linux platform, open source tools such as *top*, *sar*, and *vmstat* provide a wrapper over the basic parameters provided by the *Procfs* filesystem. In Windows, similar resource data can be accessed through the *Task Manager*, the *perfmon* tools, and related APIs. These tools track operational usage parameters only (e.g., percentage CPU used in an interval, or disk reads/writes); they do not track energy consumption by the system (OS level) or at the application level (process level).

There are some examples of software for energy usage tracking. For instance, the open source tools like *Powerstat* [10] and *Powertop* [11] for the Linux platform can provide power drawn by the system, but are only applicable for laptops. These tools fetch power-related parameters from kernel modules, e.g., ACPI [12]. Another example is the Joulemeter [13] tool from Microsoft Research. This is a Windows-based tool, which uses an external power metering device to provide the energy consumption details. It has limitations as well: (1) it works only on the Windows platform; (2) it measures virtual machine (VM)-level energy consumption (rather than process-level usage); and (3) it relies on the presence of an external watt-meter connected to the computer.

Apart from commercial or open source tools, researchers have looked at both direct and indirect methods for measuring energy consumption.

Capra and Merlo [7] propose a method of directly measuring energy usage. Their approach involves hardware enhancement, specifically a CPU clamp, as well as a watt-meter to measure actual consumption. This method assumes that CPU energy consumption is representative of overall system energy consumption. However, a typical OS process will access other resources, such as reading/writing from disk, communicating over the network and reading/writing to memory - all of which require energy; measuring only the CPU's energy consumption does not capture the full picture of energy usage.

In the case of indirect measurement, researchers have developed mathematical models of energy consumption of a computer. The most widely used equation in that respect is $Energy = \sum(k \times frequency \times voltage^2 \times \Delta time)$, where k is a constant [8]. This model, like Capra and Merlo's direct measurement method, approximates the energy usage of a computer as a function of the energy consumption of the CPU, and does not consider the energy usage of other components, such as disk, network, or memory. Modern computers typically have a multi-core CPU architecture, where multiple processes can share a single core of the CPU and the multiple threads of the same process can run on the same core of the CPU. In such a framework, applying this type of model to determine process-level energy consumption is a challenging task. Further, modern business applications make extensive use of hard disk, memory, and network resources. Each of these tasks requires some measure of power, which needs to be considered in computing the total energy consumptions of an OS process in a computer. Mathematically modeling such complex scenario in real business applications is challenging, if not impossible.

Other researchers have also applied mathematical modeling in the context of energy consumption measurement. For example, Zedlewski et al. [14] modeled the energy consumption of hard disk read/write operations, and Mahesri and Vardhan [15] provided a comparative study of power consumption of different components of a laptop. Economou et al. [16] developed a model of a system using a linear model called "Mantis." In the calibration phase, the approach captures various hardware parameters. The trained linear model is then used to predict the power consumption of the system. One major limitation of this approach is in the assumption of linearity – this holds true for the simple applications on which the authors tested the approach; however, for complex real-life business applications (such as an online e-business application or an OLTP application) the linear assumption does not hold, resulting in reduced accuracy. Perhaps closest to our own work is the approach proposed by Li et al. [17]. Here, the authors modeled energy consumptions as a function of CPU, memory and disk utilization. Unlike previous models, instead of a linear model, they model utilization as a piecewise linear function, where a different linear function is used at different levels of CPU utilization. However, the approach was still demonstrated in the context of VM power metering, rather than executing processes at the application level.

To our knowledge, our work here is the first attempt to measure energy usage at the OS process level instead of OS or VM level. After a calibration phase which requires a watt-meter, our model provides a real-time estimate of energy usage by OS processes, in a manner similar to *top* on Linux or the Windows *Task Manager*. No additional hardware enhancements are required, and our model does not assume

linearity.

III. Solution Approach

The energy consumed by the hardware in a server is mainly due to the operations driven by the software running on it. Executing processes share hardware resources through the operating system. Resource usage is captured by the operating system by tracking specific usage parameters. For example, in Linux, the Procfs [18] filesystem provides detailed data regarding a process' resource usage, including disk, CPU, memory, and network usage. The energy usage of an OS process p is a function of the overall hardware resource usage by the process. So we have,

$$Energy_p = \mathcal{F}(cpu_p, disk_p, network_p, memory_p) \quad (1)$$

where $Energy_p$ is the energy consumed by process p. $Energy_p$ is a function $\mathcal{F}$ of CPU usage cpu_p, disk usage $disk_p$, network usage $network_p$ and memory usage $memory_p$ of the process p.

Our proposed approach consists two main steps.

1) Identify the process-level metrics provided by the OS for CPU, disk, network and memory usage.
2) Develop a model $\mathcal{F}$ of energy consumption.

To identify $\mathcal{F}$, we collect OS-level energy utilization ($Energy_o$) using a watt-meter [9] along with OS-level overall CPU (cpu_o), disk ($disk_o$), network ($network_o$) and and memory ($memory_o$) utilization data. Thus, for the OS level we will have following equation,

$$Energy_o = \mathcal{F}(cpu_o, disk_o, network_o, memory_o) \quad (2)$$

From Equation 2, using the OS-level energy and resource utilization data, we derive the model $\mathcal{F}$, which is then used in Equation 1 to estimate the process-level energy consumption from the process-level resource utilization data. In this approach, we need the watt-meter only to collect the OS-level data in the calibration stage, which is used to create the model $\mathcal{F}$. Once $\mathcal{F}$ has been determined from the OS-level data, we estimate process-level energy consumption online without the need for a watt-meter. Such calibration is required only once for a particular configuration of hardware and OS. If two machines have similar configurations both at the hardware and the OS level, the model $\mathcal{F}$ will remain unchanged and the model derived from one computer can be utilized on another. The overall approach is depicted in Figure 1.

Having described our high-level approach for OS process level energy measurement, we describe in detail the resource usage metrics used and modeling techniques in our approach in Section IV. We describe the details of our approach in the context of Linux server, which is predominant in modern data centers; however, our approach is equally applicable for other operating systems, using the resource usage parameters available on those operating systems.

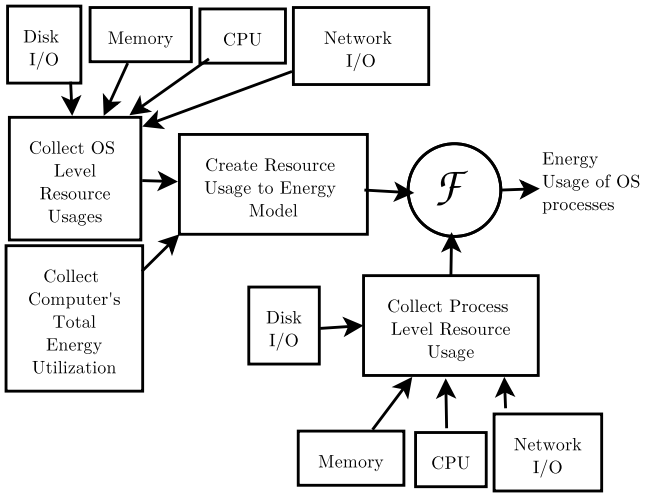

Fig. 1. Approach for Process Level Energy Measurement

IV. Solution Details

In our proposed approach, there are two phases. In the first phase, the calibration phase, we measure the OS-level resource usage parameters, i.e., the total resource usage by the OS for use in modeling $\mathcal{F}$ in Equation 2. In the second phase, we measure the process-level resource usage parameters to estimate the energy consumption of each OS process. We focus on four major resources: CPU, disk, memory, and network.

A. Resource Usage at the OS Level

In this section, we describe the specific usage parameters for CPU, memory, disk and network that we consider in the model. We also describe how we gather the system-wide OS-level data for these usage parameters.

For the CPU, we are primarily interested in overall percentage CPU utilization. To find the overall percentage CPU utilization of a system, we periodically collect the following CPU utilization parameters from the *Procfs* filesystem.

- P_{user}: Percentage of CPU utilization from user programs in user space
- P_{nice}: Percentage of CPU utilization due to nice priority processes executing in user space/mode
- P_{system}: Percentage of CPU used in execution of kernel space instruction excluding interrupts and soft irqs
- P_{iowait}: Percentage of CPU used waiting for I/O to finish
- P_{irq}: Percentage of time used to service interrupts
- P_{soft}: Percentage of time CPU service software interrupts
- P_{idle}: Percentage of time CPU is idle
- Intr/sec: Number of interrupts per second

Next, we find the total number of CPUs in a system from the */sys/devices/system/cpu/present* file. This information is used in parsing the */proc/stat* file, which contains the time spent by each individual CPU in different task categories: *user*, *nice*, *system*, *iowait*, *irq*, *soft* and *idle*. The CPU utilization is represented in */proc/stat* file as follows: $(CPU_i, T_{(i,usr)}(t), T_{(i,nice)}(t), T_{(i,system)}(t), T_{(i,idle)}(t),$

$T_{(i,iowait)}(t), T_{(i,irq)}(t), T_{(i,soft)}(t), T_{(i,steal)}(t)$, $T_{(i,guest)}(t)$), where CPU_i represents the i^{th} CPU. $T_{(i,j)}(t)$ represents the total time spent by CPU_i in the j^{th} category $j \in$ {*nice, system, iowait, irq, soft, and idle*} category starting from the computer's bootup time.

Based on these parameters, the total time spent on j^{th} category by all CPUs is computed as follows,

$$T_{total,j}(t) = \sum T_{(i,j)}(t), \forall i \tag{3}$$

And the total time spent in all categories by all CPUs is computed as follows,

$$T_{total}(t) = \sum T_{total,j}(t), \forall j \tag{4}$$

We take two snapshots of the data from the */sys/devices/system/cpu/present* file at two different times, and compute the percent utilization of CPU on j^{th} category, P_j as,

$$P_j = (\Delta T_{total,j} / \Delta T_{total}(t)) \times 100 \tag{5}$$

Lastly, we compute the total percentage CPU utilization $P_{overall}$ by the system as,

$$P_{overall} = 1 - P_{idle} \tag{6}$$

In our setup, we have configured our implementation to take a snapshot of *Procfs* file system every second. We use $P_{overall}$ as the CPU utilization parameter in Equation 1.

We collect memory usage data from the */proc/meminfo* file in the *Procfs* filesystem. In this file, $memTotal$, $memFree$ and $buffers$ represent the total memory, free/available memory, and buffer size of the system, respectively, in kilobytes.

For capturing memory usage, we compute two memory usage parameters:

- P_{memory}: Percentage of memory used by the system.

$$P_{memory} = \frac{memFree}{memTotal} \times 100 \tag{7}$$

- P_{Buffer}: Percentage of buffer used by the system.

$$P_{Buffer} = \frac{Buffer}{memTotal} \times 100 \tag{8}$$

We obtain disk usage statistics from the */proc/diskstats* file in the *Procfs* filesystem. This file contains the amount of data read from and written to different partitions of the disk since the time of computer bootup. The values represent the number of *sectors* (1 sector = 512 bytes) read or written. In our experiment, we have considered the widely used SCSI hard disk (other disk architectures would provide data structures similar to those described here). The primary partition is represented as sda_i and the secondary partition is represented by sdb_j. The data in the file is represented as $(sda_i, Disk_{(read,i)}(t), Disk_{(write,i)}(t))$ and $(sdb_j, Disk_{(read,j)}(t), Disk_{(write,j)}(t))$ where $Disk_{(read,i)}(t)$, $Disk_{(write,i)}(t)$, $Disk_{(read,j)}(t)$ and $Disk_{(write,j)}(t)$ are the number of sectors read from primary partition i, written to primary partition i, read from secondary partition j, and written to secondary partition j at time t, respectively. We read the */proc/diskstats* file every second to compute $Disk_{read/sec}$ and $Disk_{write/sec}$:

- $Disk_{read/sec}$: Number of sectors read per second from the disk.

$$Disk_{read}(t) = (\sum Disk^{sda}_{(read,i)}(t) + Disk^{sdb}_{(read,j)}(t)), \forall i, \forall j \tag{9}$$

$$Disk_{read/sec} = \Delta Disk_{read}(t) \tag{10}$$

- $Disk_{write/sec}$: Number of sectors written per second on the disk

$$Disk_{write}(t) = (\sum Disk^{sda}_{(write,i)}(t) + Disk^{sdb}_{(write,j)}(t)) \forall i, \forall j \tag{11}$$

$$Disk_{write/sec} = \Delta Disk_{write}(t) \tag{12}$$

Both $Disk_{read/sec}$ and $Disk_{write/sec}$ are used in our model as disk usage parameters.

We obtain network statistics from the */proc/net/dev* file of the *Procfs* filesystem. We periodically (every second) read the */proc/net/dev* file to get $(eth_i, data_{(i,received)}(t)$ and $data_{(i,transmit)}(t))$, where eth_i represents i^{th} network interface in the machine, $data_{(i,received)}(t)$ and $data_{(i,transmit)}(t)$ represent the total data received and transferred from i^{th} interface measured at time t (at a per-second granularity). The data transferred and received is measured in bytes. Following this, we compute two parameters to represent network usage in our model.

- $Network_{transmit/sec}$: Kilobytes of data transmitted per second through all the network interfaces in the system.

$$Network_{transmit}(t) = \sum data_{(i,transmit)}(t), \forall i \tag{13}$$

$$Network_{transmit/sec} = \frac{\Delta Network_{transmit}(t)}{(1024)} \tag{14}$$

- $Network_{receive/sec}$: Kilobytes of data received per second through all the network interfaces of the system.

$$Network_{receive}(t) = \sum data_{(i,receive)}(t), \forall i \tag{15}$$

$$Network_{receive/sec} = \frac{\Delta Network_{receive}(t)}{(1024)} \tag{16}$$

In the OS-specific model in Equation 2, we use both $Network_{transmit/sec}$ and $Network_{receive/sec}$ as the network usage parameter.

B. Resource Usage Metrics at Process Level

In this section we describe the process-specific resource usage parameters for four resources: CPU, memory, disk and network.

We parse the */proc/<PID>/stat* file to calculate the CPU utilization for a process, *PID* represents the process ID. For each process, we draw the following parameters from the *stat* file for the process.

- $utime$: Amount of time the process has been scheduled in user mode

- $stime$: Amount of time the process has been scheduled in kernel mode
- $cutime$: If the process has child threads, this parameter represents the time spent for the process when the child threads are executing in user mode
- $cstime$: Amount of time a process is waiting when the child threads are scheduled in kernel mode

From these CPU usage parameters we compute, for each process, the total user time and system time up to time t as follows,

$$T_{usr}^{process}(t) = (utime + cutime)$$
$$T_{system}^{process}(t) = (stime + cstime) \quad (17)$$

where, $T_{usr}^{process}(t)$, $T_{system}^{process}(t)$ represents the time spent by a process and its threads in user mode and system mode, respectively, until time t.

We read the */proc/<PID>/stat* file periodically (every second) to compute the percentage CPU consumption by the process in both user mode and system mode within a periodic interval as follows.

$$P_{usr}^{process} = \frac{\Delta T_{usr}^{process}(t)}{\Delta T_{total}(t)} \times 100 \quad (18)$$

$$P_{system}^{process} = \frac{\Delta T_{system}^{process}(t)}{\Delta T_{total}(t)} \times 100 \quad (19)$$

where $P_{usr}^{process}$ and $P_{system}^{process}$ represent the percentage of CPU utilization by the process in user and system mode, respectively.

The total CPU utilization of the process during the periodic interval can be computed as,

$$P_{overall}^{process} = P_{usr}^{process} + P_{system}^{process}$$

To gather the memory used by a process, we parse the */proc/<PID>/status* file in the *Procfs* filesystem, where *PID* represents the Linux process ID of the process. The parameter used from the file */proc/<PID>/status* is $VmSize$. This parameter represents the amount of memory used by the particular process at any point of time, in kilobytes. We obtain the total memory of the system $memTotal$ from the */proc/meminfo* file. Thus, we compute the percentage memory utilization by the process at a time t, $P_{memory}^{process}$, as follows,

$$P_{memory}^{process} = \frac{VmSize}{memTotal} \times 100 \quad (20)$$

To obtain the disk I/O usage details for a process, we read the */proc/<PID>/io* file of the *Procfs* filesystem, where *PID* represents the Linux process ID of the process. This file provides the amount of data written to ($write\_bytes$) and read from ($read\_bytes$) the disk at any time, calculated from the time of creation of the process. We sample this file every second to obtain the following two disk usage parameters for use in Equation 1.

- $Disk_{read/sec}^{process}$: This represents number of disk sectors read by the process per unit time.

$$Disk_{read/sec}^{process} = (\Delta read\_bytes)/512 \quad (21)$$

- $Disk_{write/sec}^{process}$: This represents number of disk sectors written by the process per unit time.

$$Disk_{write/sec}^{process} = (\Delta write\_bytes)/512 \quad (22)$$

We monitor data transfers from the connected sockets of different processes using the following files in the *Procfs* filesystem: */proc/net/tcp*, */proc/net/tcp6*, */proc/net/raw*, and */proc/net/udp*. Each of these files provides the data transmitted and data received by an *inode*. The file descriptor, *fd*, corresponding to the *inode* can be obtained from the */proc/<PID>/fd* directory. There may be multiple file descriptors associated with a process. The */proc/net/tcp* provides the data transfer as *tx_queue* (transmitted data) and *rx_queue* (received data) for each *inode*.

We first find all the file descriptors fd_i for a process i in */proc/<PID>/fd* and their corresponding *inode* numbers $inode_i$. Let $tx\_queue_{inode_i}$ and $rx\_queue_{inode_i}$ represent the data transmitted and received, respectively, by $inode_i$. Then, we compute two network usage parameters related to data transmitted and received by a process as follows,

- $Network_{transmit/sec}^{process}$: Rate of data transmitted by a process i.

$$Network_{transmit/sec}^{process}(i) = \sum_{\forall inode_i} tx\_queue_{inode_i}, \quad (23)$$

- $Network_{receive/sec}^{process}$: Rate of data received by a process i.

$$Network_{receive/sec}^{process}(i) = \sum_{\forall inode_i} rx\_queue_{inode_i}, \quad (24)$$

We use $Network_{transmit/sec}^{process}$ and $Network_{receive/sec}^{process}$ in our process specific model in Equation 1.

C. Data Collection and Modeling

In the data collection and modeling phase, we collected the OS-level resource usage and total energy consumption at varying system loads. Figure 2 depicts the architecture used for the data collection phase. The target machine is the server where we want to estimate the energy consumption of each process. This machine is a Dell server-class machine with an i7 quad-core processor and 8 GB RAM. We installed a benchmark application called RUBiS on the server. RUBiS is a web-based auction benchmark application (similar to eBay), where a user can do three basic actions: login, browse, and sell/buy. On the same local area network, we configured a laptop with JMeter [19], a widely used open source load/stress testing tool for web-based applications. We used JMeter to simulate multiple simultaneous users accessing the web benchmark application. By varying the number of simultaneous users in JMeter, we can vary the load on the web benchmark application.

We configured a watt-meter [9] between the server and the power outlet to capture the power consumption levels of the server (the watt-meter is used only for calibration; it is not needed for online estimation of process power consumption).

We connected the watt-meter to the laptop via USB cable to store timestamped energy consumption data at periodic intervals (e.g., every second) to the laptop's hard drive.

We then ran a set of load tests with different numbers of simultaneous users in JMeter to vary the workload on the benchmark application, and periodically (e.g., every second) gathered OS-level resource usage using a simple server program.

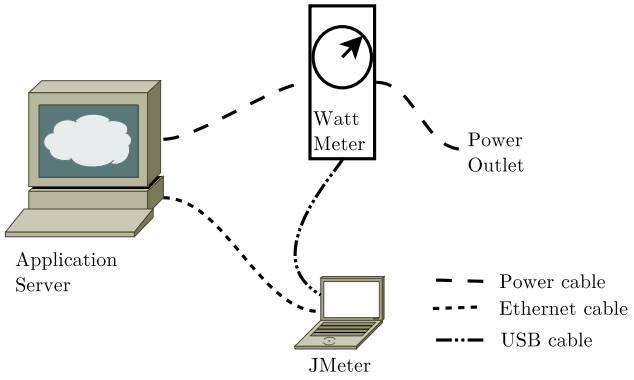

Fig. 2. Data Collection Architecture

This data collection exercise resulted in a set of timestamped OS-level resource usage data, along with the corresponding energy usage by the computer at varying workloads. We used data to model the function $\mathcal{F}$. As the relationship of energy usage with CPU, disk, network and memory usage is complex (i.e., non-linear) for real-life applications like RUBiS, we believed that the linear regression model, as followed in past research, would not be as accurate as we would like. Instead, we used support vector machine (SVM)-based regression modeling using the R statistical tool [20]. Specifically, we used epsilon regression of SVR with non-linear "radial" kernel. The output of the SVM regression is the trained model, which represents the function $\mathcal{F}$.

Once the model $\mathcal{F}$ has been built, we can estimate the process-level energy usage for each process in the OS. At run-time, we periodically (every second) collect process-level resource usage data. Based on the derived model $\mathcal{F}$ and the periodic resource usage data, the tool can estimate per-process energy usage during the corresponding period.

V. Experimental Study

In this section we demonstrate the accuracy of our method for estimating process-level energy usage. We implemented our approach on a Linux-based enterprise operating system (CentOS release 6.3).

To accurately capture the total energy usage of the server, we extended the open source PowerStat [10] tool for server based Linux system. This tool is integrated with the Watts-Up-Pro watt-meter [9] and has been released to the open source community [21]. We configured this software to periodically capture the total energy usage by the computer from the watt-meter and store it with a timestamp for each data point. In parallel, we ran a program to collect timestamped process-level resource usage data. Based on these timestamps, we can synchronize the energy consumption and resource usage data. We collected these data points at one-second intervals.

To simulate variation in workloads on the server, we ran the RUBiS benchmark application in the server and varied the workload of the RUBiS benchmark using simulated users from JMeter. We varied the total number of simulated simultaneous users in JMeter from 10 to 1,500 over the course of the experiment. This resulted a variation in the total power consumption of the server from 58 watts to 130 watts. We ran the experiment for one hour and collected 3,600 data points.

We present the correlation between the total energy consumptions of the system and the various resource usage parameters in Table I. With the exception of P_{iowait}, all CPU usage parameters have a high correlation with the system's total energy consumption. In particularly, both the P_{usr} and P_{system} parameters have a very high correlation (above 0.8) with energy usage, while P_{idle} has very high negative correlation (-0.83). This indicates the energy usage of the system will increase with the increase of P_{usr} and P_{system}, and with the decrease of idle CPU P_{idle}. Table I also demonstrates that the total energy consumption of the system is also related to other resource usage parameters, such as P_{memory}, P_{buffer} and $Disk_{write/sec}$. The dependency of energy consumption on network parameters ($Net_{transmit/sec}$ and $Net_{receive/sec}$) is not significant.

Next, we apply a supervised learning method to create the model $\mathcal{F}$ and use it to estimate process level energy consumption. We divided the collected OS-level data points into two random sets for training and testing. The training data set contains 20% of the data points (i.e., 720 data points) and the testing data set contains rest of the data points (i.e., 2,880 data points). First, we used the training data set to model the server's energy usage as a function of server's total resource usage, i.e., to estimate the function $\mathcal{F}$. Next, in the testing phase we used $\mathcal{F}$ to estimate the total OS-level energy consumption and the energy consumption of each process, and compare it with actual energy measurement collected through the watt-meter. For supervised learning-based modeling, we used support vector machine (SVM) [22]. For comparison purposes, we built a model using linear regression based modeling, which is the predominant model-based energy-estimation technique in the existing literature [23]. We discuss the results of this experiment next.

First, we compare the server's total energy consumption as estimated by the model to the actual value as measured by the watt-meter on testing data set. We computed the percentage error of the estimated total energy compared to the actual total energy consumptions as measured by watt-meter. We plotted the distribution of the accuracy percentage in energy estimation in Figure 3 for both SVM and linear regression based modeling, as well as the actual energy consumption as measured by the watt-meter. As can be seen from Figure 3, in the SVM case, 90% of the data points are above 95% accuracy and only about 10% of the estimated energy value data points

TABLE I
CORRELATION OF POWER WITH PARAMETERS

Parameters	Correlation with Energy Consumption	Parameters	Correlation with Energy Consumption
P_{usr}	0.8357564	P_{memory}	-0.3290601
P_{system}	0.8195136	P_{buffer}	-0.4539801
P_{iowait}	-0.04192834	$Disk_{read/sec}$	-0.08099673
P_{irq}	0.4619991	$Disk_{write/sec}$	0.3768353
P_{soft}	0.6421051	$Net_{transmit/sec}$	-0.003412124
P_{idle}	-0.8314755	$Net_{receive/sec}$	0.1519203

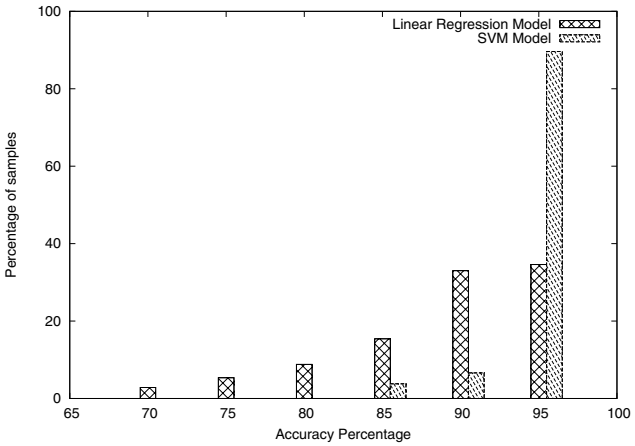

Fig. 3. Accuracy Distribution comparison for SVM vs. Linear Modeling

had more than 10% error. In contrast, the accuracy distribution for linear regression modeling is significantly less accurate, as demonstrated in Figure 3. For linear modeling, only 35% data points have accuracy above 95% and about 30% data points have accuracy of 80% or less. Table II provides more detail, including the maximum, mean, and minimum errors for various percentile data points for both the linear and SVM modeling approaches. These clearly demonstrate that our SVM modeling approach provides higher accuracy for estimating the total energy consumptions than traditional linear modeling-based approach.

Having demonstrated the accuracy of total OS-level energy consumption estimation using our proposed approach, we next

TABLE II
ERROR DISTRIBUTION COMPARISON

Percentile	Statistical values	SVM Model	Linear Model
100	Mean Error Percentage	5.23	8.34
95	Mean Error percentage	3.29	6.72
	Max Error percentage	20.82	30.19
90	Mean Error percentage	2.65	5.64
	Max Error percentage	10.65	20.92
80	Mean Error percentage	1.95	4.31
	Max Error percentage	5.84	12.48
NA	Min Error percentage	0.004	0.025

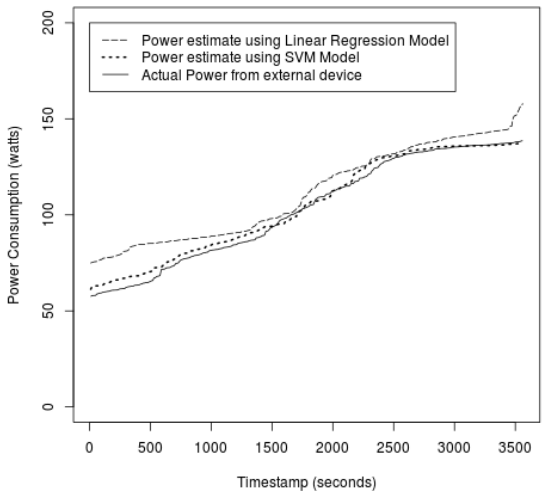

Fig. 4. Comparison of actual vs. estimated energy measurement: SVM vs. Linear Modeling

consider the accuracy of our approach in estimating the energy consumption for individual OS processes.

We note that there is no readily available tool at present to measure the process level energy consumptions – if there were, our research here would be unnecessary – so we resort to an indirect method of computing the accuracy of our approach. For each test data point, we estimated the process-level energy consumption for each OS process and then summed estimated energy use across all processes concurrently running in the server to compute an estimate of the server's total energy consumption. We compared this estimated total energy with the actual energy consumed by the server as measured by the watt-meter, and present the results in Figure 4.

From Figure 4, we can see that the total computed energy using the process level energy estimation by SVM modeling matches closely with the actual energy consumption of the server across a wide range of our experimental data points. The overall average error-percentage of our process level energy estimation is less than 4%. For comparative purposes, we also plotted the estimated total energy based on linear regression modeling. The linear regression model performs less accurately than the SVM model, with an average error value of 15% compared to 4% for SVM based modeling. Further, we note that the linear regression model shows a higher deviation from the actual energy values at the higher and lower total energy consumption data points in our experimental results. For the mid-experiment data points, the linear regression model performs slightly better, though it provides substantially lower accuracy than the SVM modeling approach.

To summarize, based on the set of experimental results presented here, we demonstrated the following:

1) Although the CPU is a major resource usage parameter in predicting energy consumption of an OS or processes

running in the OS, other resource usage parameters such as disk and memory also play significant roles in energy consumption.

2) The SVM-based modeling approach provides high accuracy in estimating OS level total energy consumption compared to the most popular linear model based approach.
3) The SVM-based modeling also provides high accuracy in estimating the energy consumptions of individual OS process.

VI. Conclusion

In this research, we described an approach and implementation for estimating process-level energy consumption. Our implementation is Linux-based, but our approach is generalizable to other operating systems. Our approach has high accuracy (over 95%), with several advantages compared to existing approaches: (1) it does not require a watt-meter or other additional hardware changes for online estimation (the watt-meter is required for the calibration step only); (2) it considers resource usage across all resources in a server (rather than assuming that CPU usage is a valid proxy for total energy consumption of the system); and (3) it relies on SVM based non-linear modeling rather than the linear regression model (which has been used in most of the related literature), resulting in very low error rates even for complex multi-tier e-business applications, such as RUBiS. In the future, we intend to further explore this approach with a broader set of experiments on a wider range of applications and hardware to determine the robustness of our approach.

Acknowledgement

This project was partially funded by National University of Singapore Grant R-253-000-087-133.

References

[1] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, p. 034008, 2008.
[2] T. C. I. Agency, "The world factbook: Country comparison:: Electricity - consumption," 2012.
[3] O. C. Project., "Hacking conventional computing infrastructure," 2011.
[4] G. Inc., "Efficient data centers." 2011.
[5] B. Childers, H. Tang, and R. Melhem, "Adapting processor supply voltage to instruction-level parallelism," in *Kool Chips 2000 Workshop*, 2000.
[6] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The filter cache: an energy efficient memory structure," in *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1997, pp. 184–193.
[7] E. Capra and F. Merlo, "Green it: everything starts from the software," 2009.
[8] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '03. New York, NY, USA: ACM, 2003, pp. 160–171. [Online]. Available: http://doi.acm.org/10.1145/781027.781048
[9] "Electronic Educational Devices. Watts Up Pro." https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0, 2009, [Online; accessed 1-April-2013].
[10] "Powerstat," http://powerstat.sourceforge.net/, 2012, [Online; accessed 1-April-2013].
[11] "PowerTop," https://lesswatts.org/projects/powertop/, 2012, [Online; accessed 1-April-2013].
[12] A. L. Brown, "The state of acpi in the linux kernel," in *Linux Symposium*, 2004, p. 121.
[13] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 39–50.
[14] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, "Modeling hard-disk power consumption," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, vol. 28, 2003, pp. 32–72.
[15] A. Mahesri and V. Vardhan, "Power consumption breakdown on a modern laptop," in *Power-aware computer systems*. Springer, 2005, pp. 165–180.
[16] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *In Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, 2006, pp. 70–77.
[17] Y. Li, Y. Wang, B. Yin, and L. Guan, "An online power metering model for cloud environment," in *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*. IEEE, 2012, pp. 175–180.
[18] T. Bowden, B. Bauer, J. Nerin, S. Feng, and S. Seibold, "The/proc filesystem," *Linux Kernel Documentation*, 2000.
[19] "Foundation, A.S. Apache JMeter." http://jmeter.apache.org/., 2012, [Online; accessed 1-April-2013].
[20] "R - Open Source Statistical Tool," http://www.r-project.org, 2012, [Online; accessed 1-April-2013].
[21] V. K. S. Kaushik Dutta, "Powerstat for Server," http://powerstat.sourceforge.net/, 2012, [Online; accessed 1-April-2013].
[22] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.
[23] G. A. Seber and A. J. Lee, *Linear regression analysis*. Wiley, 2012, vol. 936.