A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models

Gaurav Dhiman
gdhiman@cs.ucsd.edu

Kresimir Mihic
kmihic@ucsd.edu

Tajana Rosing
tajana@ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404

ABSTRACT

In this paper we present a system for online power prediction in virtualized environments. It is based on Gaussian mixture models that use architectural metrics of the physical and virtual machines (VM) collected dynamically by our system to predict both the physical machine and per VM level power consumption. A real implementation of our system shows that it can achieve average prediction error of less than 10%, outperforming state of the art regression based approaches at negligible runtime overhead.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Distributed Systems

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Virtualization, Power, Workload Characterization, Gaussian Mixture Models, Regression

1. INTRODUCTION

Power consumption is a critical design parameter in modern data center and enterprise environments, since it directly impacts both the deployment (peak power delivery capacity) and operational costs (power supply and cooling). The energy consumption of the compute equipment and the associated cooling infrastructure is a major component of these costs. The electricity consumption for powering the data centers in the US is projected to cross $7B by the end of 2010 [13, 14].

Modern data centers use virtualization (eg. Xen [3] and VMware) to get better fault and performance isolation, improved system manageability and reduced infrastructure cost through resource consolidation and live migration [6]. Consolidating multiple server applications running in different virtual machines (VMs) on a single physical machine (PM) increases the overall utilization and efficiency of the equipment across the whole deployment. Thus, the creation, management and scheduling of VMs across a cluster of PMs in a power aware fashion is key to reducing the overall operational costs. Policies for power aware VM management have been proposed in previous research [15] and are available as commercial products as well. These policies require accurate understanding of the power consumption of the PM, as well as its breakdown among the constituent VMs for optimal decision making. Currently they treat the overall CPU utilization of the PM and its VMs as an indicator of their respective power consumption, and use it for guiding the power management policy decisions (VM migration, DVFS etc.). However, as we show in the latter sections, based on the characteristics of these different co-located VMs, the overall power consumption of the PM can vary by more than 2x at same CPU utilization levels. This observation underlines the main objective of this paper: to develop a system that can dynamically estimate power consumption of the whole machine as well as of its constituent VMs.

Our system is based on a power model that co-relates power consumption to the architectural metrics (instruction throughput, memory access rate etc.) of the workloads running inside the VMs. These metrics are collected dynamically at a per VM level, and are fed to the model to make the power prediction. The system is non-intrusive and requires no changes to the VMs and the applications running inside it. The model itself is based on Gaussian Mixture Models (GMMs), that are trained by running a small set of benchmark applications. We implement our system on a state of the art machine running Xen as the virtualization technology, and show that it can perform online power prediction with an average error of less than 10% across workloads with varying characteristics and utilization levels. Our evaluation shows that our model outperforms regression models based on just CPU utilization [9] by a factor of 5, and the architectural metrics by a factor of 3. Furthermore, we show that our methodology also scales across different machine configurations, which is common in modern real world deployments, incurring minimal runtime overhead.

Based on this discussion, the *primary contributions* of our paper are as follows: (1) We propose a system for capturing both PM and VM level power consumption in virtualized environments. To our knowledge, this is the first work targeting this problem. (2) We propose a power modeling methodology based on Gaussian Mixture Models that is independent of the configuration of the machine, making it extremely scalable. (3) We implement and evaluate our system on real machines, and show that it can achieve high accuracy at minimal runtime overhead compared to state of the art regression based algorithms.

*This work has been funded in part by Sun Microsystems, UC MICRO, Center for Networked Systems (CNS) at UCSD, MARCO/DARPA Gigascale Systems Research Center and NSF Greenlight.

DAC'10, June 13-18, 2010, Anaheim, California, USA

The rest of the paper is organized as follows: we discuss the related work in section 2 followed by motivation on our approach towards power modeling in section 3. We then present the details of our system design in section 4, followed by the discussion on our implementation, experimental methodology and results in section 5. We finally conclude with a discussion on the major results and future directions in section 6.

2. RELATED WORK

Most of the prior work has focused on modeling and predicting power for standalone systems [5, 7, 11, 12, 16]. Bellosa et al [4] were the first to propose power models based on correlation between power consumption and performance counters that captured activity across various functional units of the CPU. Similarly, Contreras et al [7] proposed a regression based power model for a PXA255 processor based on performance counter activity. These models were primarily targeted towards embedded processors, and exploited fine grained utilization information. At a more coarse grained level, Li et al [12] identified a strong co-relation between IPC (Instructions per cycle) and CPU power consumption and utilized it to estimate run time power consumption of OS routines using a power/IPC regression model. Bircher et al [5] extend this idea across other subsystems like memory, I/O etc. to provide a complete system power estimation. At server level, Lewis et al [11] provide a detailed model to account for all system components (CPU, memory, disk, fans etc.) based on regression, while Fan et al [9] propose simple power models based on just CPU utilization to account for the whole system power.

Most of these models have been proposed and verified in either single core based systems ([4, 5, 7, 12]) or with just one thread running at any point in time [11], which limits their applicability for multi-core based systems capable of running multiple threads at a time. In addition, these models do not apply to virtualized environments, where the OS itself runs on a virtualized hardware, with no access to hardware performance counters.

The existing work on power estimation/modeling in virtualized environments are based on using CPU utilization [15], which as we show in this paper, can be misleading. We build upon the prior work on power modeling in terms of co-relating architectural metrics of the workload to its power consumption, and propose a power accounting system, that is able to provide complete observability into the power consumption at the VM level across a cluster of PMs.

3. MOTIVATION

The power consumption of a given PM can be divided into two parts:

(1) *Baseline Power* (P_{base}): The power consumption when the machine is idle. This comprises of power consumption due to the fans, CPU, memory, I/O and other motherboard components in their idle state.

(2) *Active Power* (P_{active}): The power consumption due to the execution of the workload. The active power component is determined by the kind of workload that executes on the machine, and the way it utilizes CPU, memory and I/O components.

The primary focus of this paper is on dynamic estimation of P_{active} of the whole system as well as each VM running on it. This is important, since it is representative of the power consumption of the workload, and can be used by VM management policies for better power management and provisioning across the cluster of systems [9, 15].

In a virtualized system, there might be multiple VMs, each with its own OS, running together on a PM. For proper power accounting in such a scenario, we should be able to breakdown the active power

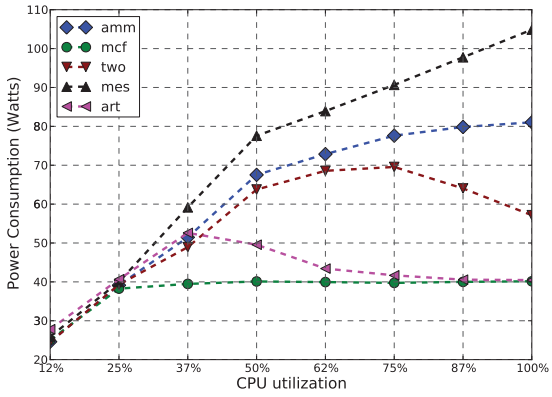


Figure 1: Active Power consumption of application at different CPU utilization levels

of the system at each VM level. To develop such a power accounting system for VMs, we first try to understand the relation between their execution characteristics and power consumption in isolation. For this purpose, we perform experiments on a Dual Quad core Xeon (hyper-threading equipped) [1] based machine. For all our experiments, we measure and collect the power output from the power supply of the machine. We first estimate the P_{base} of the system by measuring the power when the machine is idle. We then run different benchmarks from the CPU2000 suite in a VM at different CPU utilization levels, and record the system power periodically. We use multiple threads of the benchmarks to generate higher utilization levels. We then subtract the baseline power (P_{base}) from the measured power to estimate the active power (P_{active}) due to the benchmark execution. Note, in this paper we do not model fan power consumption. Recent work has shown that fan power can be estimated through modeling of its power characteristics and fan speed [17] that can then be plugged into the model we present in this paper. In all our experiments, the fan speed was constant, which ensures that P_{active} does not include fan power component.

Figure 1 plots the active power consumption (P_{active}) of five benchmarks (ammp, mcf, mesa, twolf, art) at different utilization levels. We can clearly observe that the power consumption has no direct corelation to the CPU utilization. For instance, for mesa, the power consumption increases almost linearly to the increase in utilization, since it has high IPC, which implies higher CPU power consumption. We observe that the slope of increase changes at 50% utilization. This happens since our machine has eight cores and sixteen CPUs (due to hyper-threading), with two CPUs per core. When we reach 50% utilization that corresponds to eight threads of the benchmark, and beyond that the threads start sharing the pipeline, which reduces the individual IPC of threads sharing the pipeline. Thus, the contribution of new threads to power consumption beyond the 50% utilization point is lower. In contrast, for mcf, we observe that the power consumption saturates at 25% utilization. This happens since mcf has very high memory access rate (Memory accesses per cycle or MPC), and adding more threads increases cache contention, and results in lower IPC per thread. For art, we observe that the power consumption actually starts to drop after 50% utilization. This happens due to pipeline sharing and higher cache conflicts, which significantly reduce the overall IPC. In summary, we observe that the power consumption of a workload is much more than a function of just its CPU utilization level. The power consumption between two workloads at same level of CPU utilization can be as high as 70 Watts. Architectural metrics

like IPC, MPC, cache conflicts etc. are very important to predict the power consumption of a workload at any given CPU utilization level.

Based on these observations, we design a system, that can dynamically capture the VM level and whole PM level architectural metrics and accordingly predict the active power consumption of the PM as well as its constituent VMs. The prediction is done using a Gaussian mixture model (GMM) based predictor that can map the architectural metrics to power consumption. The choice of selecting GMM as opposed to regression based approaches adopted by existing state of the art is motivated by the observation in this section, that power consumption is a function of architectural interactions at different levels (IPC, memory, cache etc). This results in different clusters of power consumption values, with each cluster representative of some architectural interaction that is difficult to capture through a single set of coefficients generated by regression. As we show in section 5, multiple Gaussian components of a GMM are able to represent and capture these interactions and clusters much better, and consequently outperform regression based approach.

4. DESIGN

The design of our system is based on a client server computing model as shown in Figure 2. The clients are the PMs in the cluster responsible for characterizing the architectural metrics of the running VMs and updating the server with this information. The server is a machine that collects this data from all the client PMs in the cluster, and predicts the active power consumption of the client and its constituent VMs using a power model based on a GMM in the form of a power report as shown in Figure 2. This information can then be used by the policies running on the server for effective power provisioning, power and thermal management through VM migration, DVFS etc. In this section we will describe the design of these components in detail.

4.1 Clients

The clients in our system are the PMs in the cluster that are part of a common virtualized environment as shown in Figure 2. We use Xen for virtualization in our setup, since it is open source and forms the baseline technology for several commercial products like XenSource and Sun xVM. In Xen, a VM is an instance of an OS that is configured with virtual CPUs (VCPUs) and a memory size. The number of VCPUs and memory size is configured at the time of VM creation. A PM can have multiple VMs active on it at any point in time, and Xen multiplexes their VCPUs across the real physical CPUs (PCPUs) in the machine. Thus, a VCPU is analogous to a thread, and a VM is analogous to a process in a system running a single OS like Linux. Besides this, Xen has a control domain, known as Domain-0 or Dom-0, which provides interfaces to the user to create and manage VMs in the system.

From the discussion in the previous section, we know that power consumption of each VM is a function of the architectural metrics like IPC, MPC etc. and the CPU utilization of the workloads running inside it. Thus, we need mechanisms to extract these metrics dynamically in a way that is minimally invasive for the running applications. For obtaining the CPU utilization of each VM, we make use of Xen-API, that is implemented by Xen over Dom-0. However, Xen has no default mechanism to collect architectural metrics (like IPC, MPC etc.) of VMs at run time. For this purpose, we augment Xen and implement a "performance counter manager" inside the Xen scheduler to dynamically characterize the IPC, MPC and CTPC (cache transactions per cycle) of the running VMs at runtime using performance counters widely available in modern CPUs (see Figure 2). The system starts by characterizing the VCPUs as

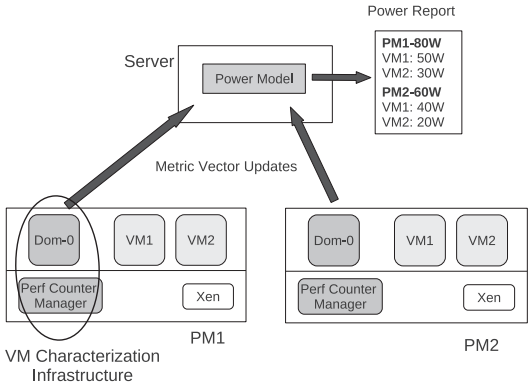


Figure 2: Overall System Design

they are scheduled on the PCPUs by programming their respective performance counters to account for the number of instructions executed, memory accesses and cache transactions. Using these counts it is able to dynamically maintain the architectural metrics for each VCPU as they get scheduled on and off the PCPUs, and uses them to derive the VM level metrics as well. It is able to isolate these metrics at per VM level by storing them in the VM data structure and also aggregate them to derive the client/PM level metrics. This information is regularly collected by an application sitting inside Dom-0 of Xen. The information for each VM (architectural metrics + CPU utilization) and the whole client is then grouped together in the form of a *metric vector* and transmitted to the server periodically as illustrated in Figure 2. The overall design of our infrastructure is similar to the system proposed in [8]. As desired, this system is minimally invasive and requires no modifications to the OS and applications running inside the VM.

The server gets the metric vectors from all the clients in the cluster, and uses the GMM for each client (described in the following section) to predict the active power consumption of the client and its breakdown at per VM level in a power report as shown in Figure 2.

4.2 Server and Power Prediction Model

The server runs the power prediction model in our system (as shown in Figure 2), and is responsible for generating client power reports based on their metric vector updates. The power model is specific to each client in the system and is trained offline. The underlying theory behind the model is GMMs and Gaussian Mixture vector quantization (GMVQ) [10] based training and classification. GMMs are essentially a collection of Gaussian distributions that we use to capture the different possible architecture level interactions (IPC, cache and memory usage etc.) for a given client system. The methodology comprises of first building the GMM based on power and metric values from some representative benchmarks that display different level of architecture interactions and power levels. We refer to this as the *training phase*. For this purpose, GMVQ is used to fit a GMM to the training data (metric and power values), which essentially tries to find the "best" model by minimizing the mismatch between the model being considered and the training data being observed. Once the GMM is formed, it can be used for performing online power prediction based on the client level metrics in the *prediction phase*. The predictor employs a GMVQ based classifier, that maps the input metric vector to the Gaussian component "closest" to it, and uses the components parameters (average power in our case) to perform online prediction. We now present the details on how these phases are accomplished in our system.

Training Phase: To perform power prediction, the model must first understand the relationship between the metric vector of a workload and the active power consumption of the client it executes on. This necessitates a training phase, where the model is constructed by feeding it with metric vectors and the corresponding P_{active} of the client system for workloads with varying characteristics. Let the input training vector be of the form $\mathrm{x} = \{x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{power}\}$, where the IPC, MPC, CTPC and CPU utilization metrics are collected by our infrastructure discussed above, and P_{active} is measured using our setup described in section 3. Once we have such input vectors for different kind of workloads, we build the GMM using the following steps:

(1) We divide the CPU utilization space (0-100%) into bins, and assign the input vector to the bin corresponding to its CPU utilization (x_{util}). The size of the bins are decided experimentally during the training phase itself, and will be discussed shortly.

(2) Inside each bin, we then fit the vectors using GMVQ that uses the quantizer mismatch (QM) distortion [2]. This fitting generates multiple Gauss components (g_i) in each bin, where each component captures cluster of vectors representative of the relationship between the architectural metrics and power consumption within that cluster. Each g_i found by the algorithm is described by its mean $\mathrm{m}_i = \{m_{ipc}, m_{mpc}, m_{ctpc}, m_{util}, m_{power}\}$, covariance K_i $(x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{power})$ and probability p_i. These Gaussian components form a power prediction model for the corresponding client.

(3) The power prediction model is then tested on the training vectors. For this purpose we remove the x_{power} from the training vectors and feed it to the model as test vectors. The model then uses a methodology based on GMVQ classification [10] for power prediction. The classifier uses the minimum distortion, or the "nearest-neighbor" approach to classify test vector, which is the same distortion measure (QM distortion) used to design GMM, as the classification rule. Based on this, it selects the best Gaussian component from the mixture, and then predicts the mean power of the selected Gaussian component m_{power} as P_{active} for the test vector. The prediction is then compared to x_{power} of the vector to measure the accuracy of the predictor.

(4) We do multiple iteration of steps 1-3 with different bin sizes (ranging from 20% to 100%), and settle for the GMM which gives the least prediction error. The intuition behind using multiple bins is that it allows formation of more fine grained Gaussian, which can potentially capture the power clusters in the training data better, and hence result in better predictions.

Prediction Phase: Once the GMM for a client is trained, it can be used to perform online power prediction using the GMVQ classifier based predictor described above in step 3 dynamically. The client metric vector is used to predict the P_{active} for the whole client system, and the VM level metric vectors are used to perform VM level power breakdown.

To illustrate the operation of the proposed algorithm, consider the following example. Let the training vectors $\mathrm{x}_{\mathrm{train}} = \{x_{ipc}, x_{mpc}, x_{power}\}$ be taken from randomly chosen application at runtime. For simplified analysis we assume just IPC and MPC as architectural metrics and a single bin, so all the training vectors are used together to generate the GMM. The resulting GMM consists of the Gaussian components with the following mean vectors: $\mathbf{m}_1 = \{7.8, 0.0134, 90W\}$, $\mathbf{m}_2 = \{14, 0.0027, 120W\}$, $\mathbf{m}_3 = \{9.8, 0.004, 103W\}$ (we omit covariance matrix and the component probability since it is not used for prediction). Note, these IPC and MPC values are at PM/VM level, i.e. they are a sum of the IPC and MPC of all the threads running within it. The resulting GMM is illustrated in Figure 3. Each point in the plot corresponds to a training vector instance (the color indicates their power, with brighter colors indicating higher power), and the ellipses indicate the Gaussian in the GMM. Now, let $\mathrm{x}_{\mathrm{test}} = \{15, 0.01\}$ be the test vector corresponding to a VM, for which we want to estimate power. Using QM distortion, we find g_2 to be the closest to the test vector. Thus, the power dissipated by the VM executing workload described by $\mathrm{x}_{\mathrm{test}}$ is predicted as 120W.

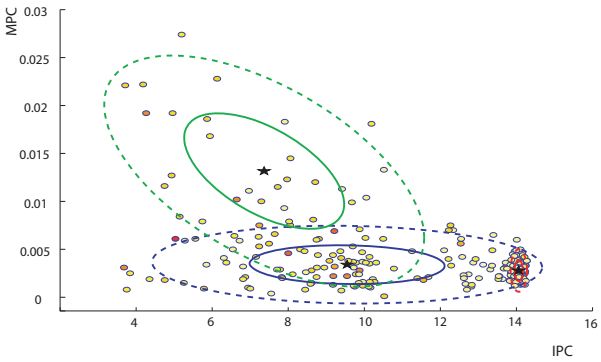


Figure 3: Example of training and prediction using GMVQ

5. EVALUATION

In this section we present the details of the implementation of our system, experimental methodology and the overall results. In the presentation of the results, we also provide a discussion on the results of our system compared to the state of the art regression based approaches and the runtime overhead of our system.

5.1 Implementation & Methodology

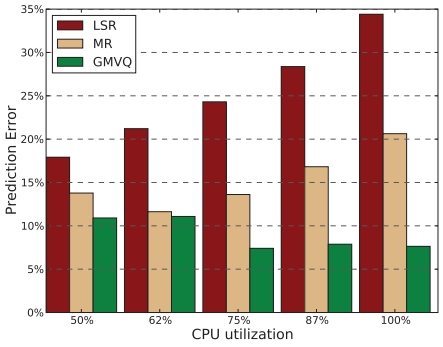
We use state of the art 45nm 2.93GHz Dual Intel Quad Core Xeon E5570 (16 CPUs) [1] based server machine with 8GB of DDR3 SDRAM as the client. It runs Xen 3.3.1 as the hypervisor, and uses Linux 2.6.30 for Dom-0. The *performance counter manager* (see Figure 2) is implemented as part of the Xen credit scheduler (the default scheduler) to record VM level metrics. In Dom-0, the VM characterization infrastructure is implemented in two parts:

(1) A Linux driver that interfaces with the performance counter manager to get the VM metrics and exposes it to the application layer.

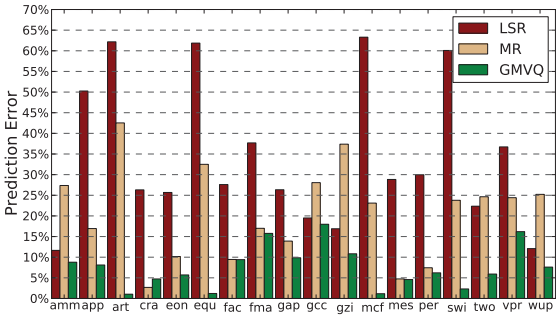
(2) An application client module that gets the VM metrics data from the driver and passes it on to the server as part of the metric vector for power prediction every 5s.

Our system requires no modifications to the applications and OS in the VMs running on the client, which makes the system easily deployable in real world setup. With each metric vector update we also log the corresponding active power consumption of the system using the setup described in Section 3. This is used to estimate the goodness of our predictions.

The GMM for the client system is trained using four benchmarks from the SPEC2000 suite, namely: mcf, gcc, mesa and gap. The benchmarks are selected based on their varying characteristics in terms of instruction throughput, and memory and cache behavior. We run each of them inside a VM with utilization varying from 10-100% and generate 200 training vectors for each run. We test our model with 18 benchmarks (including the four training benchmarks) from SPEC2000 suite running in single VM as well as multiple VMs with varying utilization levels. We generated 200 test vectors for each run of the benchmarks, and compared the measured power against the predicted power to estimate the error for each vector.



(a) Overall Comparison across different CPU utilization levels



(b) Individual benchmark predictions at 100% utilization level

Figure 4: Comparison of GMVQ with LSR and MR

We compare our model (which we refer to as GMVQ) against two other regression based power models representative of state of the art – (1) *Linear regression with a single metric (CPU utilization)*: This model is representative of coarse grained power model suggested in [9, 15]. We refer to it as LSR. (2) *Multivariate regression on utilization + architectural metrics*: This model uses exactly the same training vector as we do. We refer to it as MR. There is no prior work, which uses multi-variate regression for power prediction in virtualized environments. We include it to demonstrate the robustness and suitability of a GMM based approach over regression for power modeling.

5.2 Results

Homogeneous VMs. In the first set of experiments, we ran all the benchmarks as part of a single/multiple VM(s) for overall model verification, and varied the CPU utilization by varying the number of threads of the benchmark. Figure 4 shows the PM and VM level active power prediction errors for LSR, MR and GMVQ models. The per VM level predictions are verified against measured power for the benchmark divided by the utilization of that VM. This is reasonable since the same benchmark runs across all the VMs, and thus their power consumption must be proportional to the number of threads of the benchmarks they are running.

Figure 4a shows the average error across all the 18 benchmarks at different utilization levels. We include results for utilization $\geq$ 50%, since this is when the active power consumption becomes significant enough to require some dynamic power/thermal management decisions. We can observe that across all the utilization levels, GMVQ is consistently low ($<$10%) in terms of prediction error. In contrast, for LSR and MR, the error increases significantly with increasing utilization levels. At 100% utilization, GMVQ outperforms LSR by almost a factor of 5, and MR by a factor of 3. LSR performs poorly at high utilization levels due to the wide range in active power consumption across different benchmarks at the same utilization level (see Figure 1) that is difficult to capture using simple linear regression. MR does better than LSR, since it models architectural metrics of the workload, and is able to take into account the variations due to them at the same utilization level. However, it under-performs when compared to GMVQ; the reason for this is that regression is based on a single set of coefficients, which implies a single source for prediction. In contrast, GMM is based on multiple Gaussian components, each a possible source of prediction. As we observed in section 3, such an approach is more suitable for power prediction, where multiple sources are representative of different level of architectural interactions, that result in different levels of power consumption.

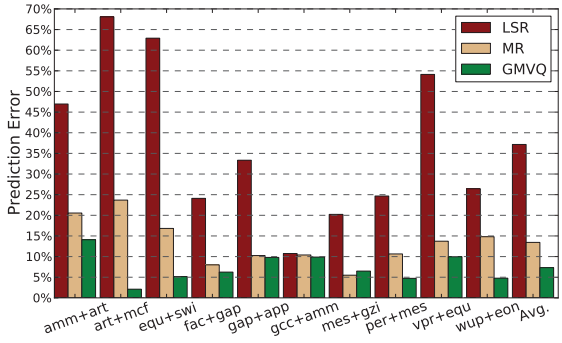


Figure 5: Heterogeneous VM combinations

Figure 4b gives a detailed breakdown of prediction error on a per benchmark basis at 100% utilization. We can observe that GMVQ is precise and stable across the whole set, achieving $<$10% error for around 85% of the benchmarks. LSR and MR are very inconsistent, with LSR incurring more than 60% prediction error for around 4 benchmarks. Thus, apart from higher prediction error, regression based models are also prone to inconsistency. One interesting observation is that LSR performs extremely poor on benchmarks from the training set (65% and 30% errors for mcf and mesa respectively), which further shows that CPU utilization alone is not enough to capture variation in power consumption.

Heterogeneous VMs. In the second set of experiments we ran two VMs on the client each running a different benchmark. Figure 5 shows the prediction results for PM level active power consumption for 10 such combinations at 100% utilization. We can observe that GMVQ on an average achieves an error of just 7%, and outperforms LSR and MR by a factor of 5 and 2 respectively. We can again observe the inconsistency of LSR prediction errors varying from 10% to as high as 68%. MR again performs better than LSR by the virtue of taking architectural metrics into account, but under-performs when compared to GMVQ. This experiment shows that even with VMs concurrently running benchmarks with varying characteristics, our model maintains its prediction consistency and precision.

Different machine configuration. To observe how our methodology scales if the machine configuration changes, we trained all the three models with the same training set (see section 5.1) for a similar machine equipped with 24GB of memory instead of 8GB. Figure 6 shows the average prediction errors for the three models

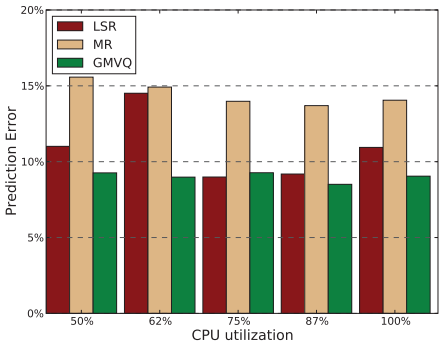


Figure 6: Overall comparison on machine with 24GB memory

across all the 18 benchmarks at different utilization levels. We can again observe, that GMVQ is extremely stable achieving $<10\%$ prediction error across all utilization levels. Interestingly LSR performs extremely well compared to the results achieved for 8GB machine (see Figure 4a), and even outperforms MR (though by small margin) across all utilization levels. This happens since the benchmarks exhibit different power characteristics for this machine due to its bigger memory size. Consequently, LSR is able to fit the training data better compared to the training data for the 8GB machine. However, at the same time this highlights the high dependence of the accuracy of just CPU utilization based regression approach on machine configuration, and the results might be different for machines with much bigger memory sizes. MR does fairly well and is more or less consistent in terms of its overall prediction when compared to the 8GB results, but is still outperformed by GMVQ by around factor of 2. Its consistency can be again attributed to it taking architectural metrics of the workloads into account.

In summary, our results highlight two important findings. First, it is important to take into account architectural metrics for server class machines for power prediction. Models based on just CPU utilization are highly inconsistent and are a function of the machine configuration. Second, the regression based models are inconsistent across varying workloads since they are unable to capture multiple architecture level interactions, which are representative of different levels of power consumption. These interactions are better captured by GMM based model.

Overhead. In our experiments we observed negligible runtime or power overhead due to our system. On the clients, the performance counter manager is implemented as a small module that does simple performance counter operations and VCPU and VM metric updates. The performance counters are hardware entities with no overhead on software execution and accessing them is just a simple register read/write operation. The Dom-0 application executes every 5s, and as explained in section 4 just reads and transmits the metrics vector updates to the server. In our experiments, we observed negligible difference in execution time or power consumption of the system ($<1\%$) across all the benchmarks with and without our system. On the server side, the power model performs online prediction based on simple GMVQ classification, which in our experiments was extremely quick (in the order of ms).

6. CONCLUSION

In this paper we present a system for online prediction of power consumption in virtualized environments. The system comprises of an infrastructure to dynamically characterize VMs based on their architectural metrics and CPU utilization, and a novel GMM based power prediction model that uses these metrics to estimate the active power consumption of the whole machine as well as its breakdown at per VM level. We implement the system on real life machines with different configurations and show that it is extremely stable and consistent achieving average prediction error of $<10\%$, while outperforming linear and multi-variate regression models by up to a factor of 5 and 3 respectively.

As part of our future work, we plan to expand our system to incorporate I/O related workload metrics, particularly disk and network, and test our system with a mix of VMs running CPU and I/O intensive workloads. We also plan to leverage our system to implement intelligent power and thermal management policies for VM allocation, scheduling and power management decisions at cluster level with the objective of reducing overall operational costs.

7. REFERENCES

[1] White paper: Intel®next generation microarchitecture (nehalem). `http://www.intel.com/pressroom/archive/reference/whitepaper_nehalem.pdf`, 2008.

[2] A. Aiyer, Y.-Z. Huang, D. B. O'Brien, and R. M. Gray. Lloyd clustering of gauss mixture models for image compression and classification. *Signal Processing: Image Communication*, 20(5):459–485, June 2005.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03*, pages 164–177, New York, NY, USA, 2003. ACM.

[4] F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 37–42. ACM, 2000.

[5] W. L. Bircher and L. K. John. Complete system power estimation: A trickle-down approach based on performance events. In *ISPASS*, pages 158–168, 2007.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI'05*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[7] G. Contreras and M. Martonosi. Power prediction for intel xscale®processors using performance monitoring unit events. In *ISLPED '05*, pages 221–226, New York, NY, USA, 2005. ACM.

[8] G. Dhiman, G. Marchetti, and T. Rosing. vGreen: A system for energy efficient computing in virtualized environments. In *ISLPED'09*. ACM, 2009.

[9] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07*, pages 13–23, New York, NY, USA, 2007. ACM.

[10] R. M. Gray. Gauss mixture vector quantization. In *Proceedings ICASSP*, pages 1769–1772, 2001.

[11] A. Lewis, S. Ghosh, and N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *HotPower*, 2008.

[12] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Perform. Eval. Rev.*, 31(1):160–171, 2003.

[13] D. Meisner, B. Gold, and W. Thomas. Powernap: Eliminating server idle power. In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, 2009.

[14] E. Pakbaznia and M. Pedram. Minimizing data center cooling and server power costs. In *ISLPED '09*, pages 145–150. ACM.

[15] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. *SIGOPS Oper. Syst. Rev.*, 2008.

[16] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *HotPower*, 2008.

[17] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering energy proportionality with non energy-proportional systems - optimizing the ensemble. In *HotPower '08*. ACM, December 2008.