pVOCL: Power-Aware Dynamic Placement and Migration in Virtualized GPU Environments

Palden Lama,* Yan Li,‡ Ashwin M. Aji,§ Pavan Balaji,† James Dinan,† Shucai Xiao,¶
Yunquan Zhang,‡ Wu-chun Feng,§ Rajeev Thakur,† Xiaobo Zhou*

*Department of Computer Science, University of Colorado, Colorado Springs, USA. {plama,xzhou}@uccs.edu
†Math. and Comp. Sci. Div., Argonne National Laboratory, USA. {balaji,dinan,thakur}@mcs.anl.gov
‡Institute of Software, Chinese Academy of Sciences, China. {liyan08}@iscas.ac.cn, {zyq}@mail.rdcps.ac.cn
§Department of Computer Science, Virginia Tech, USA. {aaji,feng}@cs.vt.edu
¶Advanced Micro Devices, USA. {shucai.xiao}@amd.com

Abstract—**Power-hungry Graphics processing unit (GPU) accelerators are ubiquitous in high performance computing data centers today. GPU virtualization frameworks introduce new opportunities for effective management of GPU resources by decoupling them from application execution. However, power management of GPU-enabled server clusters faces significant challenges. The underlying system infrastructure shows complex power consumption characteristics depending on the placement of GPU workloads across various compute nodes, power-phases and cabinets in a datacenter. GPU resources need to be scheduled dynamically in the face of time-varying resource demand and peak power constraints. We propose and develop a power-aware virtual OpenCL (pVOCL) framework that controls the peak power consumption and improves the energy efficiency of the underlying server system through dynamic consolidation and power-phase topology aware placement of GPU workloads. Experimental results show that pVOCL achieves significant energy savings compared to existing power management techniques for GPU-enabled server clusters, while incurring negligible impact on performance. It drives the system towards energy-efficient configurations by taking an optimal sequence of adaptation actions in a virtualized GPU environment and meanwhile keeps the power consumption below the peak power budget.**

I. Introduction

Graphics processing units (GPUs) are ubiquitous accelerators in high performance computing data centers today [1], [6], [17], [19], [20], mainly due to their excellent performance-to-power ratio and the availability of general-purpose programming models, such as CUDA and OpenCL. While GPUs can deliver much higher performance than CPUs, it comes at the cost of significantly higher power consumption. The thermal design power (TDP) of a high-end GPU, e.g 512-core NVIDIA Fermi, is as large as 295 watts(W), while a high-end quad-core x86-64 CPU has a TDP of 125 watts. Hence, the usage of power-hungry GPUs in the already power-consuming high performance computing systems must be carefully evaluated with respect to the impacts on overall system power efficiency.

Traditionally, GPU-accelerated application executions are tightly coupled to the physical GPU hardware, requiring each computational node to be equipped with one or more local GPUs. Recent efforts on GPU virtualization such as VOCL [21] and rCUDA [2] expose physical GPUs as decoupled virtual resources. Virtualization enables better management of GPU resources for improved resource utilization and fault tolerance. However, the potential use of GPU virtualization for power management in GPU-enabled server clusters is not well-explored. In this paper, we investigate and enable dynamic scheduling of GPU resources for online power management in virtualized GPU environments.

There are significant challenges in achieving online power management of GPU-enabled server clusters in a data center environment. Today, most data center cabinets are equipped with 3-Phase Cabinet Power Distribution Units (CDUs) to cater for increased power demands, greater equipment densities and cost reduction initiatives. However, as an artifact, the underlying system infrastructure shows complex power consumption characteristics depending on the placement of GPU workloads across various compute nodes, power-phases and cabinets. In addition, the power drawn across the three phases in the same cabinet needs to be balanced for better power efficiency and equipment reliability. This is demonstrated by our motivational case study using NVIDIA Tesla GPU-enabled server clusters and Switched CW-24VD/VY 3-Phase CDU. Furthermore, power delivery and cooling limitations in data centers impose peak power constraints at various levels. For instance, server racks are typically provisioned for 60 Amps of current. This could become a bottleneck for high density configurations, specially when power-hungry GPUs are used.

In this paper, we present a power-aware virtual OpenCL (pVOCL) framework that controls the peak power consumption and improves the energy efficiency of GPU-enabled server clusters. It automates power management in data center cabinets by performing dynamic consolidation and power-phase topology aware placement of GPU workloads. For GPU virtualization and virtual GPU migration capability, it utilizes the VOCL framework. However, our power-aware dynamic placement and migration approach is applicable to other GPU virtualization frameworks as well.

We study the effects of load imbalance across various power-phase circuits of a CDU on the power-efficiency of a GPU-enabled server cluster. We develop an automated power manager that utilizes power-phase topology awareness and dynamically drives the cluster towards more power-efficient configurations. The pVOCL runtime system periodically checks the configuration of server clusters in the face of dynamically varying GPU resource demands. Then, it performs an optimal sequence of adaptation actions that will drive the system towards the most power-efficient configuration, without violating the given power budget. The adaptations involve changing the power states of compute nodes and performing live migration virtual GPUs. We evaluate the efficacy of the pVOCL power manager framework by measuring the power and energy usage of various application kernel benchmarks under the influence of pVOCL, in comparison with the power-phase unaware static power management solution.

The rest of the paper is organized as follows: Section II presents an overview of the VOCL framework. A motivational case study is discussed in section III. The pVOCL framework and its design is described in details in section IV. We evaluate the pVOCL framework in section V. The related work is discussed in section VI. Section VII concludes our paper.

II. Overview of VOCL Framework

VOCL [21] is based on the OpenCL programming model and uses Message Passing Interface (MPI) for data communication across different computing nodes when remote GPUs are used. VOCL exposes the same API as the OpenCL programming model. Figure 1 shows VOCL components.

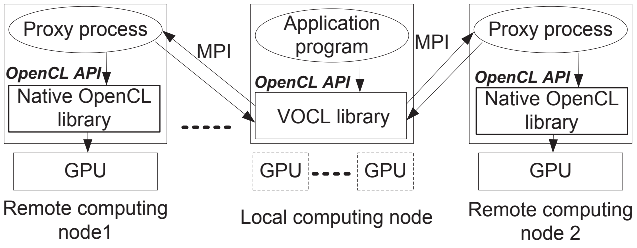


Fig. 1: The Virtual OpenCL (VOCL) framework.

The VOCL library is located on the local node. It is responsible for forwarding the OpenCL function calls to the corresponding GPUs and returning the GPU outputs to the application. It calls the native OpenCL functions to perform GPU computation on local GPUs. For using a remote GPU, it wraps up the inputs of the OpenCL function and sends them to the remote computing node using MPI communication.

The VOCL proxy is located on the remote node. It is a service provider for applications to use GPUs in the node. It receives MPI connection requests from the VOCL libraries and establishes communication channels with them. It is responsible for decoding the messages received from VOCL libraries, calling the native OpenCL functions to perform computation and sending back the function output to the VOCL libraries.

VOCL also supports live task migration across different computing nodes. Such migration is achieved by changing the mapping relationship between *virtual GPUs* (or *VGPU*) in the VOCL library and the VOCL proxy. Specifically, a VGPU represents the resources used by an application on each physical GPU. It includes the OpenCL resources such as the context, device memory, and kernels. VGPUs exist in both the VOCL library and the VOCL proxy, which are referred to as *VOCL VGPU* and *OpenCL GPU*, respectively. The VOCL framework has a one-to-one mapping between the VOCL VGPU and the OpenCL VGPU.

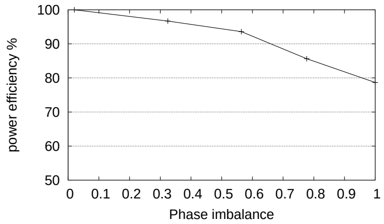


Fig. 2: Impact of phase imbalance on power efficiency. Efficiency is measured relative to the perfectly loaded configuration.

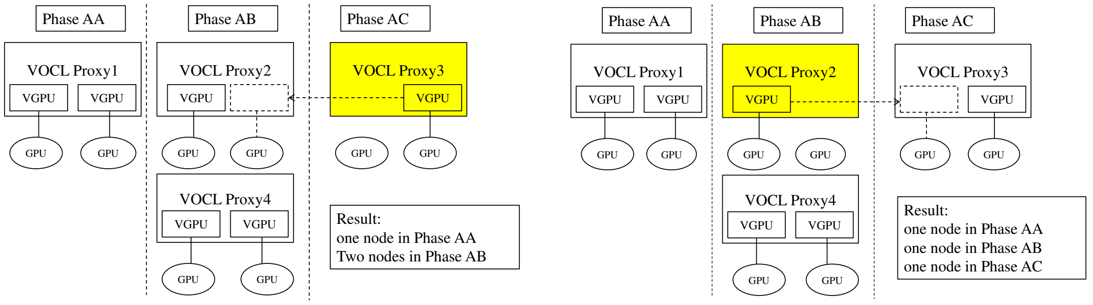
III. Motivational Case Study

The most common power generation and distribution system in use today is the balanced, three-phase system. This system is comprised of three equal-amplitude, sinusoidal voltages, that are offset from one another by 120° phase. As a consequence of this configuration, the instantaneous voltage across the phases sums to zero, which translates into mechanical balance and greater efficiency for rotating power generation machinery.

A critical factor in the efficiency of the three-phase system is the balance of the load across phases. An imbalance in current across phases results in lost power through neutral line current. We measure the efficiency with respect to three-phase load configuration in Figure 2, using the experimental platform described in Section V. We compare the power consumption of several node configurations with a configuration where the load is balanced equally across all three phases. Six compute nodes are used and we measure the power consumed over several permutations of these nodes with respect to power phase. The phase imbalance of a three-phase system is expressed as a percentage value, often defined as the maximum deviation from the average of the three-phase voltages or currents, divided by the average of the three-phase voltages or currents. We use the power consumption measurement to quantify the phase imbalance in place of voltage or current.

Given the impact of balancing the workload across phases, we explore two possible configurations for a job executing on our cluster system, shown in Figure 3. In the original configuration, two compute nodes (i.e., Proxies 2 and 3) both have two physical GPUs, but only one is currently mapped to a virtual GPU. We consolidate these two virtual GPUs to a single node, leaving one node idle and allowing it to be powered down to save power. In Figure 3(a), one way of GPU consolidation results in an unbalanced configuration with a phase imbalance of 4. In Figure 3(b), another way of GPU consolidation results in a balanced configuration with a phase imbalance of 0.

Figure 4 shows the total energy and peak power consumption for the original configuration and each configuration after consolidation. This data was measured using power



(a) An unbalanced configuration.

(b) A balanced configuration.

Fig. 3: The original system configuration and two possible resulting configurations after VGPU migration and consolidation.

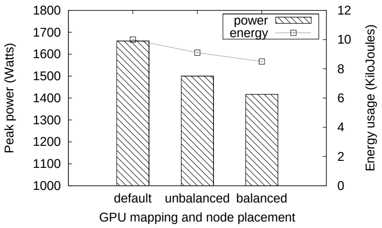


Fig. 4: Impact of consolidation and node placement.

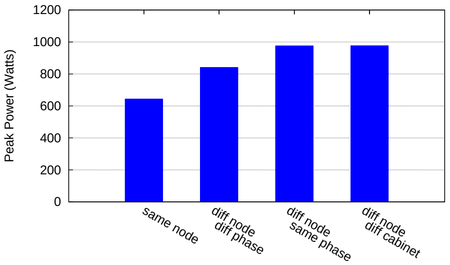


Fig. 5: Power usage for various node configurations using two GPUs.

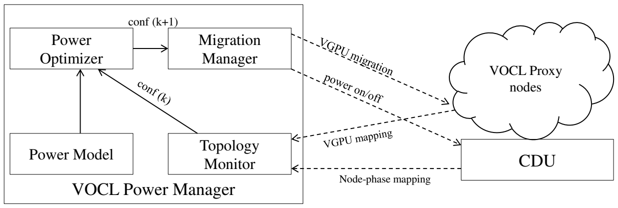


Fig. 6: The Power-aware VOCL (pVOCL) framework.

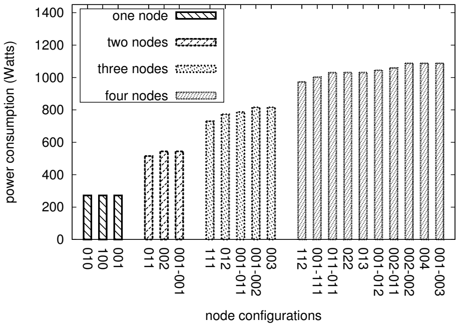


Fig. 7: Power profile of various node configurations.

monitoring equipment on our experimental cluster. We observe that consolidation yields a significant improvement in total energy, as well as a corresponding reduction in the peak power consumption. In addition, the balanced configuration achieves higher efficiency, with a 7.6% reduction in energy and 5.5% reduction in the peak power.

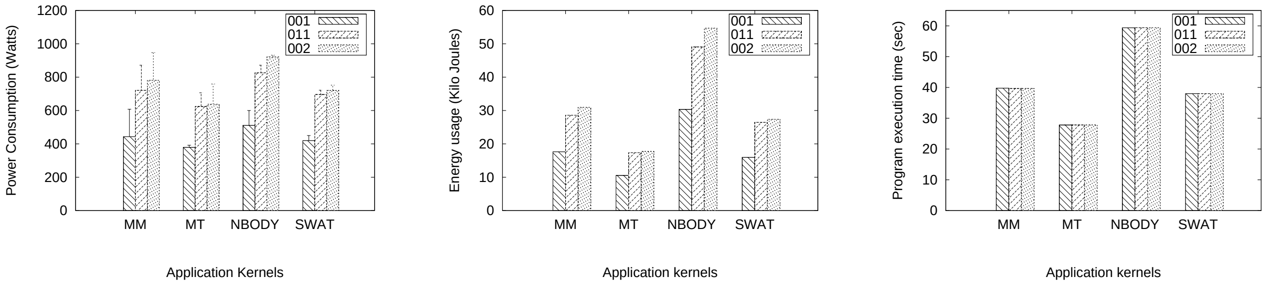
Figure 5 demonstrates the impact of GPU workload placement on the power consumption. Assuming a resource demand for two GPUs, the power consumption varies significantly depending on how the GPU workloads are distributed across compute nodes and CDU power phases.

IV. The Power-Aware VOCL (pVOCL) Framework

Figure 6 presents an architectural overview of the management components used in our power-aware VOCL framework. It consists of a power model, a topology monitor, a power optimizer, and a migration manager. The power model captures the power consumption trends for various configurations of VOCL proxy nodes in datacenter cabinets. The topology monitor periodically collects the information regarding the configuration, $conf(k)$, at the current control interval k. The power optimizer determines the optimal configuration that minimizes the total power consumption of the system while considering the overheads involved in the adaptation from the current configuration to the new configuration. The migration manager performs node reconfiguration of VOCL proxy nodes and live migration of existing VGPUs. Together these components form a control loop that dynamically reconfigures the mapping of existing VGPUs to VOCL proxy nodes and the placement of VOCL proxy nodes in the data center cabinets in order to minimize the power consumption.

A. Power Modeling

1) Power-Phase Awareness: In order to make the most power-efficient GPU consolidation and node placement decision, a model is needed that captures the impact of various node configurations on the overall power consumption. A node configuration encapsulates the mapping of VGPUs to VOCL proxy nodes and the placement of nodes at various power-phases of datacenter cabinets. For this purpose, we

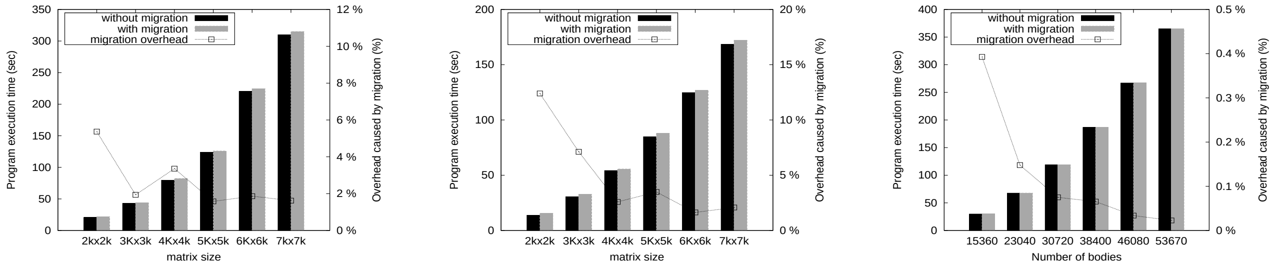


(a) Power Consumption.

(b) Energy Consumption.

(c) Performance.

Fig. 8: Power profile of various application kernels using 2 GPUs and various node configurations.



(a) Matrix Multiplication.

(b) Matrix Transpose.

(c) N-body

Fig. 9: Total execution time for each kernel over a range of input sizes with and without migration.

conducted an extensive power profiling of various possible node configurations in our testbed of GPU enabled server clusters described in Section V.

First, we measured the total power consumption by turning on idle nodes at various power phases of the two cabinets as shown in Figure 7. The notations $n_{11}n_{12}n_{13}$ and $n_{11}n_{12}n_{13}$-$n_{21}n_{22}n_{23}$ represent node configurations in one cabinet and two cabinets respectively. n_{ij} denotes the number of nodes turned on in the i_{th} cabinet and the j_{th} power-phase. The power consumption increases as more nodes are turned on, as expected. For a given number of nodes, we observe a significant variation in the power consumption for different node placements. A node configuration that is more balanced in terms of the number of nodes turned on at each power-phase is more power-efficient than other configurations.

Next, we examined whether the power consumption trend can be generalized to active nodes that execute GPU workloads. We analyzed the power profiles of four application kernel benchmarks: (matrix-multiplication (MM), n-body (NB), matrix-transpose (MT), and Smith-Waterman (SWAT). The first two kernels are compute-intensive; the other two require more data movement between host memory and device memory. In this experiment, each node configuration used two GPUs in total. As indicated in Figures 8(a) and 8(b), all four application kernels show similar variations in the power consumption and energy usage for different node configurations. Figure 8(c) shows that the application performance is independent of the node configuration.

We note that a generalized model that captures the power consumption trends for various node configurations provides sufficient information to find the most power-efficient GPU consolidation and node placement. The power minimization problem does not need actual power numbers corresponding to various application kernels. Hence, we design pVOCL to use a simple lookup table-based power model. The lookup table is generated based on the power usage data for various node configurations as provided by the datacenter administrator.

2) Analysis of Reconfiguration Overhead: A node reconfiguration of a virtualized GPU environment involves three types of actions. They are to turn on a new compute node, migrate existing VGPUs across different nodes and to turn off a compute node. The time required by a newly powered on compute node to be ready for computation may take a few seconds to a few minutes depending on the computing platform. However, it does not impact the application performance in our pVOCL framework. It is due to the fact that the GPU workloads continue to execute in the existing nodes at the time of node reconfiguration. The workloads are only migrated when the new compute nodes become available.

We analyze the performance overhead caused by VGPU migration. In Figure 9 shows the total execution time for various application kernel benchmarks with no migration and when a single migration is performed. We observe that, overall, as the problem size increases, the relative performance overhead due to migration decreases. It is due to the fact that the execution time increases faster than the migration overhead with regard to the problem size. In addition, the migration overhead is a few hundred milliseconds for compute-intensive kernels and up to a few seconds for memory-intensive kernels. We conclude that the performance degradation caused by

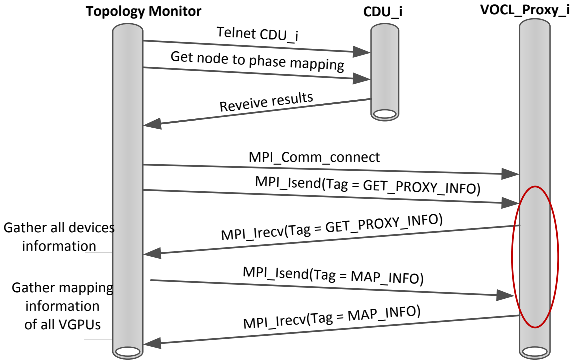

Topology Monitor
CDU_i
VOCL_Proxy_i
Telnet CDU_i
Get node to phase mapping
Reveive results
MPI_Comm_connect
MPI_Isend(Tag = GET_PROXY_INFO)
Gather all devices information
MPI_Irecv(Tag = GET_PROXY_INFO)
Gather mapping information of all VGPUs
MPI_Isend(Tag = MAP_INFO)
MPI_Irecv(Tag = MAP_INFO)

Fig. 10: The topology monitor.

migration can be negligible for programs running a reasonably long time (e.g., a few tens of seconds). Furthermore, note that we use Ethernet connected compute nodes in our experiments. The use of high speed InfiniBand can further reduce the migration overhead drastically.

B. pVOCL Components Design

GPU-enabled server clusters in a data center face dynamically varying GPU resource demands from multiple users. As a result, the number of GPUs used in various nodes changes over time. Furthermore, the mapping of nodes to different power phases in a cabinet may also change as new nodes are installed or old nodes are removed for system upgrade and maintenance. The topology monitor periodically communicates with the CDUs and the VOCL proxy nodes to collect the information about the current node configuration. Note that the VOCL proxy does not distinguish between the communication messages from the topology monitor and an application host. It merely receives the MPI messages and replies back after performing certain actions according to the information provided in the MPI message tag. In this paper, we implemented the interfaces necessary in the VOCL proxy module to enable topology monitoring.

1) Topology Monitor: As Figure 10 shows, the topology monitor performs the following operations.

1) Utilize the remote monitoring capability provided by the CDUs to get the number of nodes that are turned on at various power phases of the data center cabinets. The communication is performed by using a telnet interface to each CDU.
2) Call the MPI_Comm_connect() function to establish communication channel with each VOCL proxy node. The information regarding existing VOCL proxy nodes are stored and updated locally by the topology monitor.
3) Send control messages to the VOCL proxy nodes to request for all the GPU device information, including the number of available GPUs and the device Ids. For this purpose, it calls the MPI_Isend() function with the MPI message tag GET_PROXY_INFO.
4) Call MPI_Irecv() to receive GPU device information from the VOCL proxy nodes.
5) Call MPI_Isend() with the MPI message tag MAP_INFO to request the information about various VGPU to physical GPU mappings. Note that a physical GPU that is not mapped to any VGPU indicates that it is not used by any application. We assume that at most one VGPU is mapped to a physical GPU.
6) Call MPI_Irecv() to receive VGPU mapping information from the VOCL proxy nodes.

2) Power Optimizer: GPU consolidation and placement in a virtualized GPU environment may involve a sequence of adaptation steps that include turning on new compute nodes, migrating VGPUs and turning off compute nodes across different power-phases of data center cabinets. A direct transition between two node configurations may not be possible due to potential power budget violations. We formulate dynamic GPU consolidation and placement as the following optimization problems.

Optimization 1:

$$\text{Minimize} \quad p(c_n) \tag{1}$$

Subject to Constraints:

$$\text{for all} \quad c_i \epsilon \ [c_1, c_2, ..., c_n] \tag{2}$$

$$p(c_i) < P \tag{3}$$

$$g(c_i) = g(c_0) \tag{4}$$

Here, $p(c_i)$ is the total power consumption of all the nodes used in configuration c_i. $c_1, ..., c_{n-1}$ are the intermediate configurations generated by applying GPU consolidation and node placement actions, starting from the initial configuration c_0. Let $g(c_i)$ denote the total number of GPUs used for the configuration c_i. The objective Eq. (1) is to find a set of target node configuration, c_n, that minimizes the total power consumption while satisfying some constraints. Eq. (2) and Eq. (3) state that the power consumption due to an intermediate configuration must not violate the given power budget P. Eq. (4) states that each intermediate configuration must satisfy the current GPU resource demand, $g(c_0)$.

Optimization 2:

$$\text{Minimize} \quad \sum_{a_i \epsilon A_n} d(a_i) \tag{5}$$

Subject to Constraints:

$$\text{for all} \quad c_i \epsilon \ [c_1, c_2, ..., c_n] \tag{6}$$

$$p(c_i) < P \tag{7}$$

$$g(c_i) = g(c_0) \tag{8}$$

Let $A_n = a_0, a_1, ..., a_{n-1}$ be the sequence of adaptation actions required to transform the node configuration from c_0 to c_n. Let $d(a_i)$ denote the length of each adaptation action. The objective Eq. (5) is to find the optimal sequence of adaptation actions that minimizes the time required to reach one of the target configurations. We represent the optimization problem as a single source shortest path problem of graph theory. Here, each node configuration is a vertex in the graph and the

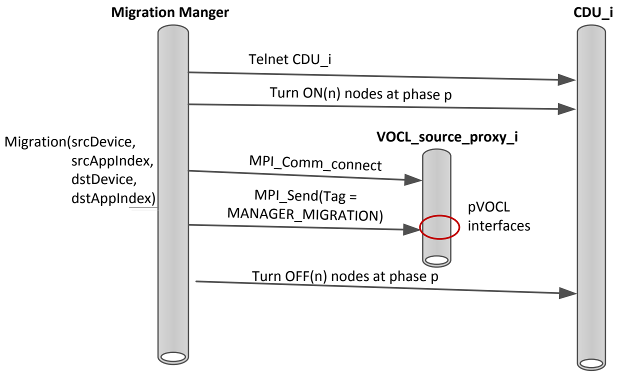


Fig. 11: The migration manager.

TABLE I: The workload mix.

Node 1	Kernel	Input Size	Kernel instances
Node 1 GPU 1	N-body	15360	20
Node 1 GPU 2	Matrix Multiplication	4kx4k	60
Node 2 GPU 1	Matrix Transpose	3kx3k	20
Node 2 GPU 2	Matrix Multiplication	3kx3k	60
Node 3 GPU 1	N-body	23040	20
Node 3 GPU 2	Matrix Transpose	4kx4k	60

adaptation actions become the edges between various vertices. The edges are weighted by the adaptation delay, $d(a_i)$. pVOCL power optimizer applies the Dijkstra’s algorithm to solve the optimization problem.

3) Migration Manager: Figure 11 shows that based on the optimal configuration suggested by the power optimizer, the migration manager performs node reconfiguration as follows.

1) Check whether the target configuration requires turning on new nodes.
2) Turn on the required number of nodes at various power-phases of the datacenter cabinets according to the target configuration. Telnet interfaces to various CDUs are used to perform this operation.
3) Based on the VGPU mapping information of the existing configuration and that of the target configuration, identify the VGPUs that need to be migrated from one proxy node to another.
4) Call the MPI_Comm_connect() function to establish communication channel with each source proxy node.
5) Call MPI_Isend() with the MPI message tag MAP_MIGRATION along with the required information to trigger VGPU migration from the source proxy nodes to the destination proxy nodes.
6) Turn off the nodes that do not have any VGPU mapping in the target configuration. Telnet interfaces to the corresponding CDUs are used for this purpose.

V. Evaluation

We evaluate the pVOCL framework in a testbed of GPU-enabled server cluster. Each computing node is equipped with Dual Intel Xeon Quad Core CPUs, 16 GB of memory, and two NVIDIA Tesla C1060 GPUs. They are installed with the Ubuntu Linux operating system and the CUDA 4.2 toolkit. The servers residing on data center cabinets are powered by the switched CW-24VD/VY 3-Phase CDU (Cabinet Power Distribution Unit). The CDUs provide power distribution and remote power monitoring capability. We use the MPICH2 MPI implementation for the compute nodes, which are connected with Ethernet.

A. Impact of GPU Consolidation

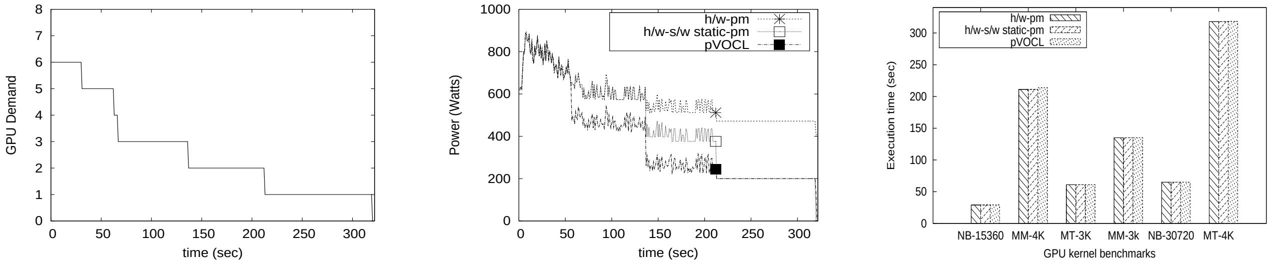
We now demonstrate the impact of GPU consolidation performed by pVOCL on the power consumption, energy usage and application performance. In this experiment, we use three GPU-enabled compute nodes located in the same power-phase circuit of a CDU. Power consumption is measured at time intervals of one second, in the particular power-phase circuit only. Table I shows the initial placement of a workload mix using application kernel benchmarks with various input sizes. Figure 12(a) shows the time-varying overall demand for GPU resources as various application kernels finish execution.

We compare the instantaneous power consumption of the system under the influence of three power management techniques. First, the hardware technique (h/w-pm) refers to the in-built idle power management of GPU hardware that causes GPUs to save power when they are idle. Second, the hardware/software static power management (h/w-s/w static-pm) is a combination of h/w-pm and a software based technique that turns off a compute node when all its GPU resources are idle. Finally, pVOCL essentially combines GPU consolidation with the first two techniques. In this experiment, pVOCL is configured to find optimal configurations within the same power-phase circuit since we are measuring the impact of GPU consolidation alone without power-phase awareness.

As shown in Figure 12(b), pVOCL reduces the power consumption more significantly than the other two techniques. It is due to the fact that pVOCL drives the system towards more power-efficient configurations by pro-actively consolidating GPUs into fewer compute nodes and turning off the unused nodes at time 63 sec and 137 sec. On the other hand, the h/w-s/w static-pm technique waits until time 137 sec to turn off node 2 when all of its GPUs become idle. Then, it turns off node 1 at time 213 sec. The relatively high power consumption in case of h/w-pm technique is due to the fact that the compute nodes are never turned off.

Figure 12(c) shows that the application performance under the influence of pVOCL is similar to that of the other two techniques. It is due to the fact that pVOCL is able to consolidate GPUs by performing VGPU migration with negligible performance overhead. We measure the overall energy usage of the system under the influence of the three power management techniques. pVOCL improves the energy efficiency by 43% compared to the h/w-pm technique and by 18% compared to the h/w-s/w static-pm technique.

Next, we study the impact of workload mix variations on the energy efficiency of the three power management techniques as shown in Figures 13 and 14. We measure the total energy usage of the system for a fixed period of time using eight variations of the workload mix shown in Table I.



(a) the workload.

(b) power consumption.

(c) performance.

Fig. 12: Power and performance comparison with different power management techniques.

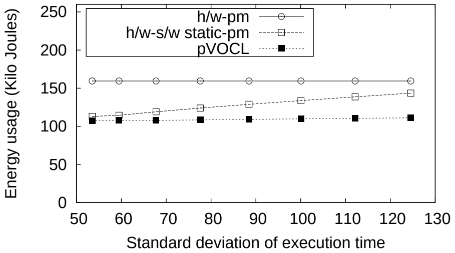


Fig. 13: Energy usage of various workload mixes.

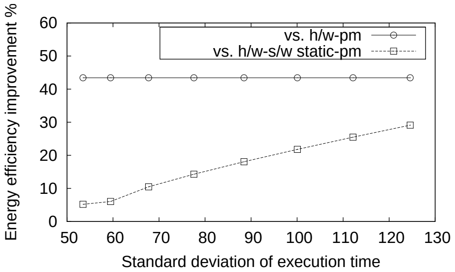


Fig. 14: Impact of GPU consolidation on energy efficiency.

In each variation, we change the number of kernel instances in such a way that the average program execution time remains unchanged while the standard deviation varies. In case of h/w-s/w static-pm technique, a higher deviation of the program execution time results in longer waiting times to turn off a node until all of its GPUs become idle. Hence, the total energy usage increases with increase in the deviation of execution time. For the h/w-pm technique, the energy usage remains unchanged since the average program execution time is the same for all workload variations. Similarly, the energy usage for pVOCL is largely unaffected by the increase in the deviation of execution time for the workload mixes used in this experiment. As shown in Figure 14, the improvement in energy efficiency by pVOCL reaches up to 29% compared to the h/w-s/w static-pm technique due to variations in the workload mix.

B. Power-Phase Topology Aware GPU Consolidation and Placement

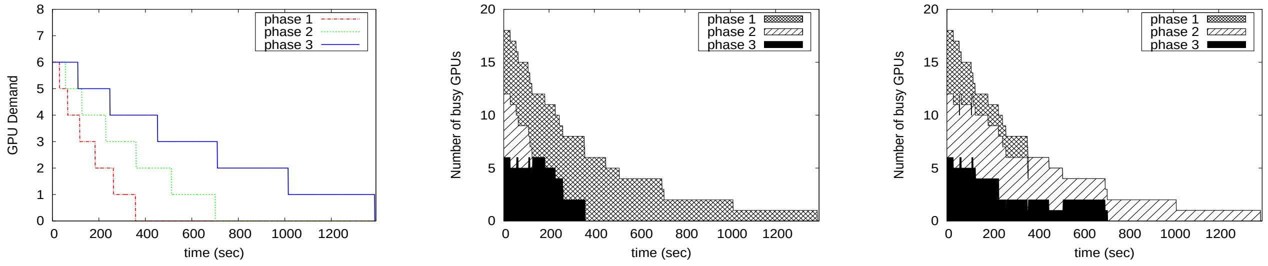
We evaluate the effectiveness of pVOCL's power-phase topology aware GPU consolidation and placement. We use three compute nodes in each power-phase circuit of the 3-phase CDU. Since each compute node has two GPUs, there are six GPUs in each power-phase. As a case study, we execute N-body kernel benchmarks with input sizes of 15360, 23040, 30720, 38400, 46080, and 53760 bodies respectively on the six GPUs. The number of kernel instances being executed are 20, 40 and 80 respectively on the GPUs residing in the three power-phase circuits. Figure 15(a) shows the time-varying demand for GPU resources in the three power-phases as various application kernels finish execution. We measure the power consumption of the entire CDU at time intervals of one second. The peak power budget of the CDU is set to be 4800 Watts in this experiment.

Figures 15(b) and 15(c) compare the dynamic distribution of GPU workloads across the three power-phases due to pVOCL's power-phase topology aware consolidation and a consolidation approach without power-phase awareness. Figures 16(b) and 16(c) show the number of compute nodes that are powered on change accordingly. In case of power-phase aware consolidation, the number of busy GPUs and the powered-on compute nodes are more evenly distributed across the three power-phases. As a result, as shown in Figure 16(a), it is able to maintain a much lower power-phase imbalance in the face of time-varying GPU resource demand. Note that there is a gradual increase in power-phase imbalance in both cases of consolidation. It is due to the fact that there are a lower number of GPU workloads available to balance the power-phases as the benchmark applications finish their execution.

Figure 17 shows the impact of power-phase awareness on the instantaneous power consumption of the system. Overall, there is an improvement of 14% in energy efficiency due to the power-phase aware GPU consolidation that drives the system towards more power-efficient configurations with negligible performance overheads. We observed similar results with various other application benchmarks, which are omitted here due to space constraints.

C. Peak Power Management and Energy Efficiency

We evaluate pVOCL's ability to manage the peak power consumption while improving the energy efficiency of GPU-enabled server clusters. In this experiment, we start with three compute nodes in one of the power-phase circuits in the 3-phase CDU. We execute 100 instances of N-body kernel

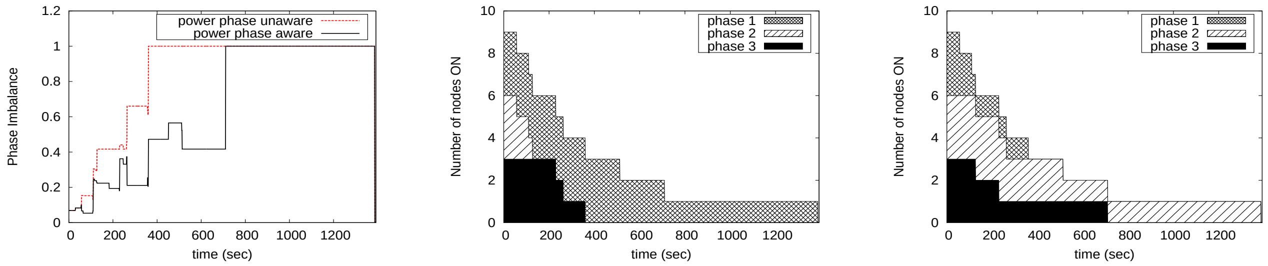


(a) the workload.

(b) power-phase unaware.

(c) power-phase aware.

Fig. 15: Impact of GPU consolidation on workload distribution across various power phases.



(a) phase imbalance comparison.

(b) power-phase unaware.

(c) power-phase aware.

Fig. 16: Impact of GPU consolidation on phase imbalance and node configurations.

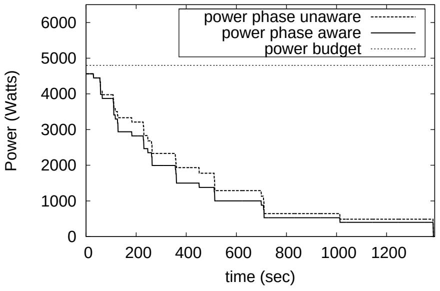


Fig. 17: Impact of power-phase topology aware consolidation and node placement on power consumption.

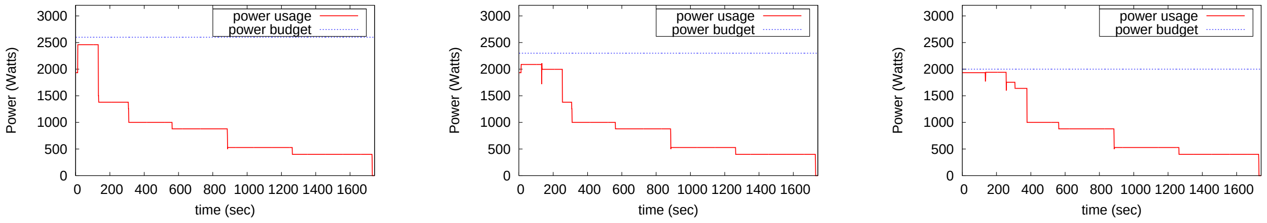
benchmarks with input sizes of 15360, 23040, 30720, 38400, 46080, and 53760 bodies respectively on the available GPUs.

Figures 18(a), (b) and (c) compare the instantaneous power consumption of the entire CDU under the influence of pVOCL's GPU consolidation and placement actions, when different power budgets are imposed. In each case, pVOCL is able to keep the peak power consumption below the given power budget. However, the decrease in power consumption due to pVOCL occurs much earlier when the power budget is higher. It is due to the fact that pVOCL performs different adaptation actions under different power constraints as shown in Figures 19 and 20. A higher power budget provides more node reconfiguration options. Hence, pVOCL is able to find the best sequence of adaptation actions to reach a power-efficient configuration under the given power constraint.

Figure 19(a) shows that pVOCL initially turns on two compute nodes in power-phase 2 and one compute node in power-phase 1 without violating the power budget of 2600 Watts. This explains the initial increase in power consumption shown in Figure 18(a). On the other hand, When the power budget is 2300 Watts, it can turn on only one compute node in power-phase 2 initially as shown in Figure 18(b). When the power budget is only 2000 Watts, it is not able to turn on any new compute node until one of the GPUs become idle around time 134 seconds. As a result, a higher power budget allows pVOCL to distribute the GPU workloads across different power-phases and reach more power-efficient configurations much earlier compared to the cases when the power budget is low. This is illustrated in Figures 20(a), (b) and (c).

Figures 21(a), (b) and (c) show the performance overhead caused by pVOCL's adaptation actions under various power constraints. We measure the total execution time of each application under the influence of pVOCL and compare it with its default execution time. Note that the applications may undergo different number of migrations or may not be migrated at all due to various GPU consolidation actions taken by pVOCL. This explains the variations in the performance overhead for different applications and power constraints. Furthermore, the performance overhead also depends on the input size of each application. We observe a negligible performance overhead of less than 0.15% in all three cases. It is due to the fact that the amount of time required to migrate VGPUs among various compute nodes is quite small compared to the computational cost. Furthermore, turning on new compute nodes does not impact application performance since the applications continue to execute in the existing compute nodes until the new nodes become available for migration.

Finally, we compare the energy efficiency of pVOCL with

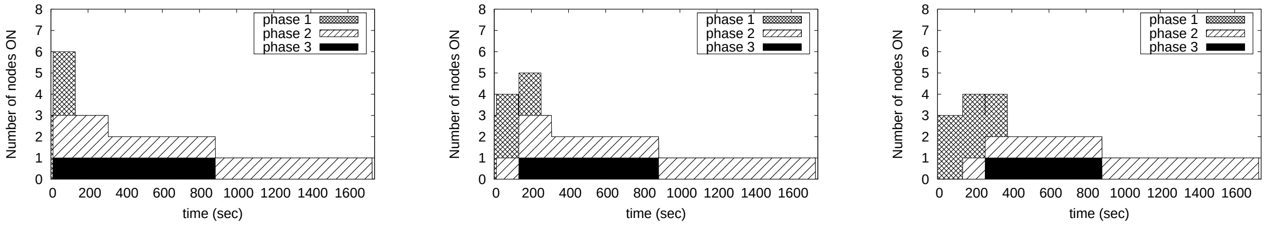


(a) with power budget 2600 W.

(b) with power budget 2300 W.

(c) with power budget 2000 W.

Fig. 18: Power consumption trends under various peak power constraints.

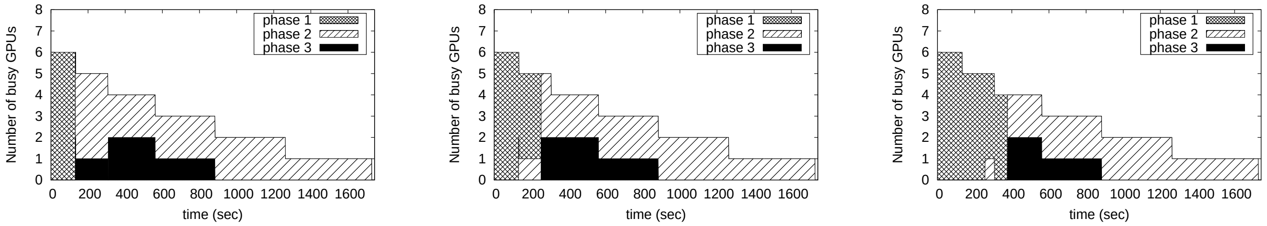


(a) with power budget 2600 W.

(b) with power budget 2300 W.

(c) with power budget 2000 W.

Fig. 19: Node configurations under various peak power constraints.



(a) with power budget 2600 W.

(b) with power budget 2300 W.

(c) with power budget 2000 W.

Fig. 20: GPU workload distribution under various peak power constraints.

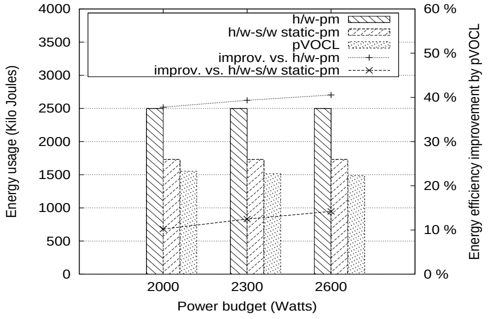


Fig. 22: Energy efficiency improvement due to pVOCL.

the h/w-pm and h/w-s/w static-pm techniques. As shown in Figure 22, the improvement in energy efficiency due to pVOCL increases with increasing power budgets.

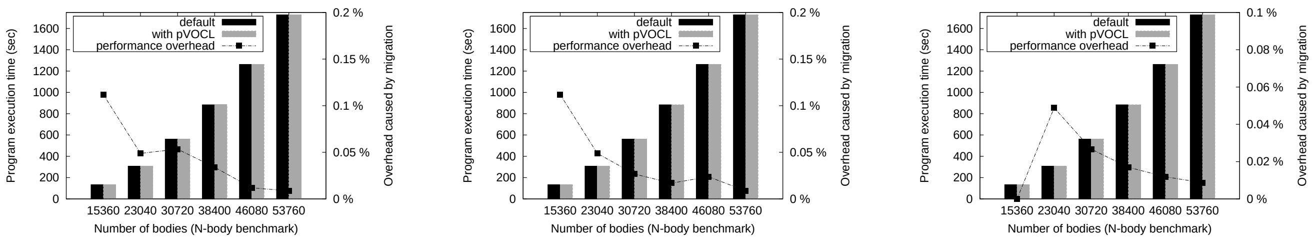
VI. Related Work

In general, GPUs deliver more performance-to-power ratio than traditional processors. GPUs are also significantly more power-hungry. Thus, analyzing and reducing the power consumption in GPU computing has become a top research topic in recent years [10], [11].

Sheaffer et al. [18] proposed a GPU simulator, Qsilver, to model GPU power consumption. However, this simulator is primarily limited to GPU architectures, particularly at ISA and micro-architectural definition design stages. Like the work of Ma et al., this framework is focused on improving power consumption of GPUs at the hardware level, which is orthogonal to our work to save energy via task migration.

Li et al. [15] developed a runtime framework to dynamically consolidate instances from different workloads from multiple user processes into a single GPU workload. To be able to consistently achieve better energy efficiency, they propose a GPU performance and power models to make predictions for potential workload consolidation alternatives and identify useful consolidations. Their work achieves energy efficiency via predictions beforehand. Once an instance is assigned on a GPU, it will be executed to completion regardless of whether there is system state change. In contrast, pVOCL dynamically adjusts the task across different GPUs via live task migration and reflect real-time state of the system.

There are significant research efforts focused on power, energy and thermal management in enterprise data centers [3]–



(a) with power budget 2600 W.

(b) with power budget 2300 W.

(c) with power budget 2000 W.

Fig. 21: Performance overhead due to pVOCL under various power constraints.

[5], [7]–[9], [12]–[14], [16]. However, existing techniques are not easily applicable to GPU-accelerated high performance computing systems. GPU virtualization itself is an emerging field of research [2], [21]. In this paper, we explore and enable the use of GPU virtualization for improving energy efficiency and managing the peak power of GPU-enabled server clusters.

VII. Conclusion

The challenge of power management in virtulized GPU environments mainly lie in the dynamic scheduling of power-hungry GPU resources in the face of complex power consumption characteristics of the underlying system infrastructure and the ever-changing GPU resource demand from multiple users. pVOCL supports dynamic placement and consolidation of GPU workloads in a power aware manner. It controls the peak power consumption and improves the energy efficiency of the underlying server system by migration of existing virtual GPUs and power-phase-aware placement of GPU-enabled server nodes in datacenter cabinets. It drives the system towards energy-efficient configuration by taking an optimal sequence of adaptation actions. We have implemented pVOCL in a cluster of GPU-enabled server nodes using four application kernels. Experimental results demonstrate that pVOCL achieves significant savings in energy consumption through dynamic consolidation of GPU workloads. It drives the system toward optimal energy-efficient configurations because of its awareness of the power characteristics of the widely used three-phase power supply in data center cabinets. Furthermore, it optimizes the power consumption while considering the transient costs incurred by various adaptations.

Acknowledgement

This research was supported in part by U.S. DOE grant DE-AC02-06CH11357, NSF CAREER Award CNS-0844983 and research grant CNS-1217979.

References

[1] M. Anderson, D. Sheffield, and K. Keutzer. A Predictive Model for Solving Small Linear Algebra Problems in GPU Registers. In *Proc. of IEEE Int'l Parallel and Distributed Processing (IPDPS)*, 2012.

[2] J. Duato, A. J. Pena, F. Silla, R. Mayo, and E. S. Quintana-Orti. rCUDA: Reducing the Number of GPU-Based Accelerators in High Performance Clusters. In *Proc. of Int'l Conference on High Performance Computing and Simulation (HPCS)*, pages 224–231, June 2010.

[3] X. Fu, X. Wang, and C. Lefurgy. How much power oversubscript is safe and allowed in data centers? In *Proc. IEEE Int'l Conference on Autonomic computing (ICAC)*, 2011.

[4] Y. Fu, C. Lu, and H. Wang. Robust control-theoretic thermal balancing for server clusters. In *Proc. of IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, 2010.

[5] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *Proc. ACM SIGMETRICS*, 2009.

[6] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron. Energy-efficient mechanisms for managing thread context in throughput processors. In *Proc. Int'l Symposium on Computer Architecture (ISCA)*, 2011.

[7] I. Goiri, K. Le, M. Haque, R. Beauchea, T. Nguyen, J. Guitart, J. Torres, and R. Bianchini. GreenSlot: Scheduling energy consumption in green datacenters. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.

[8] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. In *Proc. the EuroSys Conference*, 2009.

[9] Y. Guo and X. Zhou. Coordinated vm resizing and server tuning: Throughput, power efficiency and scalability. In *Proc. IEEE Int'lSymposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 289–297, 2012.

[10] S. Hong and H. Kim. An Integrated GPU Power and Performance Model. In *Proc. of Internation Symposium of Computer Architecture (ISCA)*, June 2010.

[11] S. Huang, S. Xiao, and W. Feng. On the Energy Efficiency of Graphics Processing Units for Scientific Computing. In *Proc. IEEE Workshop on High-Performance, Power-Aware Computing*, May 2009.

[12] P. Lama and X. Zhou. PERFUME: Power and performance guarantee with fuzzy mimo control in virtualized servers. In *Proc. IEEE Int'l Workshop on Quality of Service (IWQoS)*, pages 1–9, 2011.

[13] P. Lama and X. Zhou. NINEPIN: Non-invasive and energy efficient performance isolation in virtualized servers. In *Proc. IEEE/IFIP Int'l Conference on Dependable Systems and Networks (DSN)*, 2012.

[14] C. Li, R. Zhou, and T. Li. Enabling distributed generation powered sustainable high-performance data center. In *Proc. IEEE Int'l Symposium on High-Performance Computer Architecture (HPCA)*, 2013.

[15] D. Li, S. Byna, and S. Chakradhar. Energy-Aware Workload Consolidation on GPU. In *Proc. of the Int'l Conference on Parallel Processing Workshops*, 2011.

[16] H. Lim, A. Kansal, and J. Liu. Power budgeting for virtualized data centers. In *Proc. USENIX Annual Technical Conference*, 2011.

[17] V. T. Ravi, M. Becchi, G. Agrawal, and S. Chakradhar. Supporting gpu sharing in cloud environments with a transparent runtime consolidation framework. In *Proc. of the Int'l Symposium on High Performance Distributed Computing (HPDC)*, 2011.

[18] J. W. Sheaffer, D. Luebke, and K. Skadron. A Flexible Simulation Framework for Graphics Architectures. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2004.

[19] H. Shojania and B. Li. Cache miss analysis for gpu programs based on stack distance profile. In *Proc. IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, 2009.

[20] T. Tang, X. Yang, and Y. Lin. Cache miss analysis for gpu programs based on stack distance profile. In *Proc. IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, 2011.

[21] S. Xiao, P. Balaji, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, and W. Feng. VOCL: An Optimized Environment for Transparent Virtualization of Graphics Processing Units. In *Proc. of the 1st Innovative Parallel Computing (InPar)*, May 2012.