Power-aware Consolidation of Scientific Workflows in Virtualized Environments

Qian Zhu Jiedan Zhu Gagan Agrawal
Department of Computer Science and Engineering
Ohio State University, Columbus OH 43210
{zhuq,zhujie,agrawal}@cse.ohio-state.edu

ABSTRACT

Use of virtualization technologies is becoming prevalent, including, but not limited to, the current commercial cloud environments. At the same time, besides resources costs and application performance, energy efficiency is becoming a dominant consideration while designing systems and executing applications on them. The premise of our work is that consolidation of scientific workflow tasks can be a promising approach for energy and resource cost optimization, without incurring substantial performance degradation. This is further supported by detailed analysis of resource requirements of scientific workflow tasks, and analysis of power consumption with use of virtualization and with consolidation of workloads.

Based on the trends observed from our experiments, we have developed *pSciMapper*, a power-aware consolidation framework for scientific workflow tasks. We view consolidation as a hierarchical clustering problem, and introduce a distance metric that is based on interference between resource requirements. We extract the key temporal features of the resource (CPU, memory, disk I/O, and network I/O) requirements of each workflow task, and use a dimensionality reduction method (KCCA) to relate the resource requirements to performance and power consumption. We have evaluated pSciMapper with both real-world and synthetic scientific workflows, and demonstrated that it is able to reduce power consumption by up to 56%, with less than 15% slowdown for the workflow. Our experiments also show that overhead of pSciMapper is very low, and it can be a practical approach even for workflows with hundreds of tasks.

1. INTRODUCTION

Business and scientific computing processes can often be modeled as workflows. Formally, a *"scientific workflow is a specification of a scientific process, which represents, streamlines, and automates the analytical and computational steps that a scientist needs to go through from dataset selection and integration, computation and analysis, to final data product presentation and visualization"*[1]. Large-scale scientific workflows are used to support research in areas like astronomy [1], bioinformatics [5], physics [12], and seismic research [2], and many others. Scientific workflow management systems like Pegasus [18] and Kepler [31] have been developed with goals of supporting specification, management, and execution of workflows, especially on large datasets and high-end resources like the TeraGrid [4].

Another recent trend has been towards *cloud computing*. Lately, there is a growing interest in the use of cloud computing for scientific applications, as evidenced by a large-scale funded project like Magellan [2], and related research efforts like the Nimbus project[3]. Several projects have experimented with scientific workflows on cloud environments [17, 26, 23]. One of the characteristics of cloud environments is the use of virtualization technologies, which enable applications to set up and deploy a customized virtual environment. Virtual Machines (VMs) are configured independent of the resources and can be easily migrated to other environments. Another feature of cloud computing is the *pay-as-you-go* model, or the support for *on-demand* computing. Applications have access to a seemingly unlimited set of resources, and users are charged based on the number and type of resources used.

Resource utilization and resource costs are important considerations for both cloud service providers and cloud users. From users' perspective, with the *pay-as-you-go* model, users need to maintain the tradeoff between resource costs and application performance. Cloud providers, on the other hand, will like to maintain high throughput, and meet the needs of a large number of users with a fixed set of resources. Power management is another critical issue for the clouds hosting thousands of computing servers. A single high-performance 300 W server consumes 2628 KWh of the energy per year, with an additional 748 KWh in cooling [10], which also causes a significant adverse impact on the environment in terms of $CO2$ emissions. Even in 2006, data centers accounted for 1.5% of the total U.S. electricity consumption, and this number is expected to grow to 3.0% by 2011[4]. Lately, there has been much interest in effective power management, and techniques like Dynamic Voltage and Frequency Scaling (DVFS) have been applied to reduce power consumption [22, 50, 40, 38].

This paper focuses on effective energy and resource costs management for scientific workflows. Our work is driven by the observation that tasks of a given workflow can have substantially different resource requirements, and even the resource requirements of a particular task can vary over time. Mapping each workflow task to a different server can be energy inefficient, as servers with a very low load also consume more than 50% of the peak power [8]. Thus, *server consolidation*, i.e. allowing workflow tasks to be consolidated onto a smaller number of servers, can be a promising approach for reducing resource and energy costs.

We apply consolidation to tasks of a scientific workflow, with the goal of minimizing the total power consumption and resource costs, without a substantial degradation in performance. Effective consolidation, however, poses several challenges. First, we must carefully decide which workloads can be combined together, as the workload resource usage, performance, and power consumption is not additive. Interference of combined workloads, particularly those hosted in virtualized machines, and the resulting power consumption and application performance needs to be carefully understood. Second, due to the time-varying resource requirements, resources should be provisioned at runtime among consolidated workloads.

We have developed *pSciMapper*, a power-aware consolidation framework to perform consolidation to scientific workflow tasks in virtualized environments. We first study how the resource usage impacts the total power consumption, particularly taking virtualization

[1] http://www.cs.wayne.edu/ shiyong/swf/swf2010.html
[2] Please see http://www.cloudbook.net/doe-gov
[3] http://workspace.globus.org
[4] http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency_study

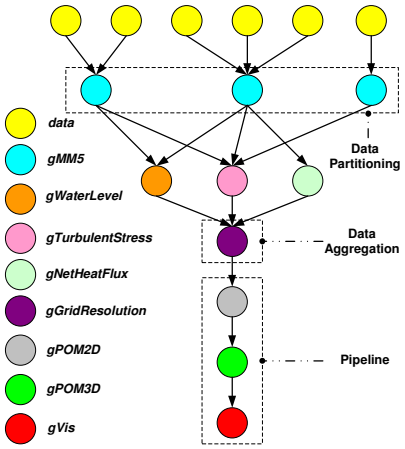


Figure 1: GLFS Workflow

into account. Then, we investigate the correlation between workloads with different resource usage profiles, and how power and performance is impacted by their interference. Our algorithm derives from the insights gained from these experiments. We first summarize the key temporal features of the CPU, memory, disk, and network activity associated with each workflow task. Based on this profile, consolidation is viewed as a *hierarchical clustering* problem, with a distance metric capturing the interference between the tasks. We also consider the possibility that the servers may be heterogeneous, and an optimization method is used to map each consolidated set (cluster) onto a server. As an enhancement to our static method, we also perform time varying resource provisioning at runtime.

We evaluated pSciMapper using several real and synthetic scientific workflows. The observations from our experiments show that our consolidation algorithm, with time-varying resource provisioning, is able to save up to 56% of the total consumed power, with only 10-15% performance slowdown, over the case where each workflow task is mapped to a different server. This is close to, or even better than, an optimal static approach that is based on exhaustive search. In addition, we also show that the overhead of pSciMapper is negligible and it is practical even for workflows with hundreds of tasks.

The rest of the paper is organized as follows. We motivate our work by analyzing the resource requirements of a scientific workflow application in Section 2. We study the impact of a workload's resource requirements, use of virtualization, and workload consolidation, on power consumption in Section 3. In Section 4, we describe our power-aware consolidation algorithm. Results from our experimental evaluation are reported in Section 5. We compare our work with related research efforts in Section 6 and conclude in Section 7.

2. MOTIVATING APPLICATIONS

The work presented in this paper is driven by characteristics of scientific workflows (or DAG-based computations), and the opportunities for optimizations they offer. We specifically study one application in details.

2.1 Great Lake Forecasting System (GLFS)

We now describe one such scientific application in details. Particularly, we first show its DAG structure. Then, we analyze the resource usage of the GLFS workflow and discuss the characteristics that make consolidation a promising solution to save power and resource costs in any virtualized environment, including the clouds.

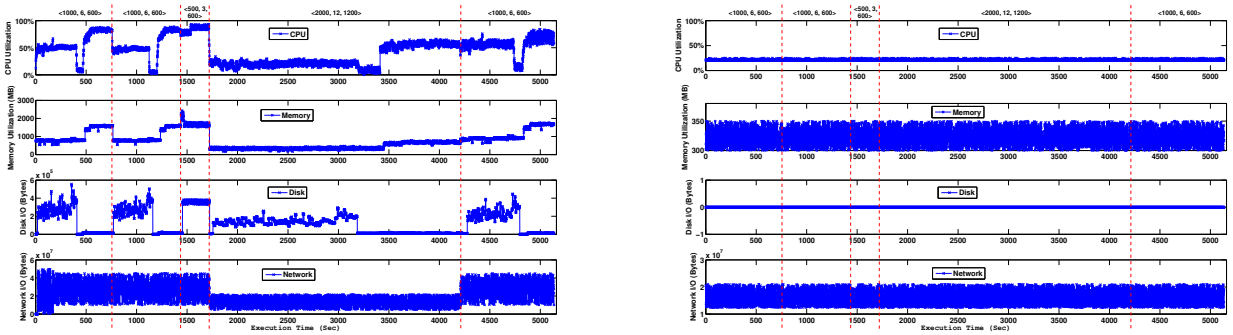
This application monitors meteorological conditions of the Lake Erie for nowcasting (for the next hour) and forecasting (for the next day). Every second, data comes into the system from the sensors planted along the coastal line, or from the satellites supervising this particular coastal district. Normally, the Lake Erie is divided into multiple coarse grids, each of which is assigned available resources for model calculation and prediction. We illustrate the GLFS workflow in Figure 1. The number of inputs processed by the workflow may increase over time as more regions of Lake Erie need to be covered. This will also translate to an increase in the number of computational tasks. The initial satellite sea surface temperature and other measured data, including the the wind speed, pressure and air humidity, are fed into the *gMM5* task. Then, the updated meteorological data go through a series of interaction models, including *gTurbulentStress*, *gNetHeatFlux* and *gWaterLevel*, as shown in the figure. These steps prepare the data that is required for predicting corresponding meteorological information using the *gPOM2D* or the *gPOM3D* tasks. Before starting the POM model, the *gGridResolution* task decides the resolution that will be used in the model. Finally, the grid resolution, along with the output from various interaction models, serve as inputs to the *gPOM2D* task (which applies a 2D ocean model) and/or the *gPOM3D* (which applies a 3D ocean model for more accurate prediction). Such POM models will predict meteorological information, including the water level, current velocity, and water temperature, for Lake Erie. The *gVis* task is then used to project the outputs onto images.

The runtime of most individual tasks in the GLFS workflow can be quite small (seconds to few minutes). Yet, GLFS is clearly a compute-intensive application. On a standard PC (Dual Opteron 254 (2.4GHz) processors, with 8 GB of main memory and 500 GB local disk space), predicting meteorological information for the next two days for an area of 1 square miles took nearly 2 hours. The execution time could grow rapidly with need for increasing spatial coverage (for example, the total area of Lake Erie is $9,940$ square miles), and/or better temporal granularity. Better response time can be achieved by using different servers for different workflow tasks. Obviously, this will also increase resource and power costs, and the right tradeoff between the execution time and resource and power costs needs to be maintained. In the next subsection, we perform a detailed analysis on the resource utilization of the GLFS workflow and motivate our work on using server consolidation to reduce power and resource costs.

2.2 Resource Usage of GLFS Workflow

We execute the GLFS application in a Linux cluster, which consists of 64 computing nodes. Each node has a dual Opteron 254 (2.4GHz) with 8 GB of main memory and 500 GB local disk space, and the nodes are interconnected with a switched 1 Gb/s Ethernet. During the execution of the GLFS workflow, we used *Sysstat* tools [3] to collect the following performance data every 1 second: CPU utilization, memory utilization, disk I/O (including reads and writes), and network I/O (including sends and receives). Like other scientific workflows, GLFS has the long-running, iterative nature. For a given certain input data, the steps between the transferring data into the first task, *gMM5* to the output from *gVis* are referred to as one invocation of the workflow. For different invocations, behavior of the application could be different, due to different data size, and/or dynamics exposed by the application. Figure 2 shows the CPU, memory, disk, and network utilization of two GLFS tasks, for 5 invocations of the workflow.

As illustrated in Figure 2, each resource usage data can be expressed as a *time series*. The resource usage of the first task varies across different invocations. In contrast, resources consumed by the second task remains roughly the same across invocations. We further analyze the characteristics of the time series we have obtained. Particularly, we have applied the auto-correlation function (ACF) [27] to the series and it shows that the resource utilization data from the scientific workflows is expressed as a periodic time series. Furthermore, a time series can be either *stationary*, with observations fluctuating



(a) (b)

Figure 2: The CPU, Memory, Disk and Network Usage of Two GLFS Components ($< 1000, 6, 600 >$, $< 500, 3, 600 >$ and $< 2000, 12, 200 >$ Represent Different Application Parameters

around a constant mean with constant variance, or *non-stationary*, where the underlying process has unfixed moments (i.e. mean and variance) that intend to increase or decrease over time. We claim that the resource utilization time series from one invocation at least meets the *weak stationary* requirements, which are that the first two moments (mean and variance) do not vary with time. The resource usage pattern of GLFS, in this respect, is similar to the observations made from other scientific workflow executions [13, 49].

Overall, we can make the following observations from the resource usage patterns we have seen. 1) The scientific workflow can be composed of a large number of tasks. The workflow tasks have a periodic behavior with respect to CPU, memory, disk, and network usage, with most of them incurring low resource utilization (e.g. Figure 2(b)). 2) The resource usage of a workflow task is significantly smaller than its peak value for more than 80% of the time (except that network activity, which remains constant over time). Therefore, provisioning resources based on peak requirements will likely lead to resource and power wastage. Furthermore, considering the time series of resource utilization of different tasks, we can see that their respective usage of the same resource does not peak at the same time. 3) The resource consumed by a workflow task can be dependent on the values of the application parameters for the particular invocation (Figure 2(a)), as well as characteristics of the host server. Comparing the 1st (and 2nd) to the 4th invocation, the resource usage pattern remains the same, with only a change in the intensity of resources consumption. However, even the pattern itself could change because of a different set of application parameters (i.e. consider the 3rd invocation). In the 5th invocation, the same parameter values were used with a change in the resource availability from the host server.

3. POWER ANALYSIS WITH VMS AND CONSOLIDATION

In the previous section, we demonstrated how the resource requirements of workflow tasks have a periodic pattern, with actual requirements being significantly lower than peak requirements most of the time. Our goal in this section is to understand the potential of using virtualization and consolidation of virtual machines (VMs) for optimizing resource and power requirements. To this end, we first analyze the relationship between resource usage and power consumption, particularly taking virtualization into account. Then, performance and power consumption with consolidation of different types of workloads is analyzed.

3.1 Power Consumption Analysis

In this subsection, we study how power consumption is related to different levels of resource activities. Particularly, we conducted the experiments on a single PC with Intel Dual Core processor at 3.0G Hz, 2 GB of memory and a 160 GB hard drive. This machine is connected to the department research network through a 1 GB Ethernet connection. We clutched a power meter to this PC to measure the power consumed as workloads are varied. Four distinct sets of synthetic workloads were used for our analysis. Note that a mix of such workloads could be representatives of the scientific workflow tasks. These sets are *CPU-bound*, *Memory-bound*, *Disk-bound*, and *Network-bound*, where CPU-bound denotes that only the CPU usage is varied, while other resource activities are fixed. Within each set, the workload is varied between *low*, *moderately low*, *moderately high*, and *high* levels. We summarize the resource needs for workloads in each set in Table 1.

Workload	CPU	Memory	Disk	Network
CPU-bound	Vary	2%	None	None
Memory-bound	70%	Vary	None	None
Disk-bound	50%	2%	Vary	None
Network-bound	50%	2%	18MB/s	Vary

Table 1: Resource Activities of Different Sets of Workloads

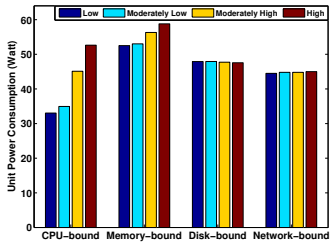
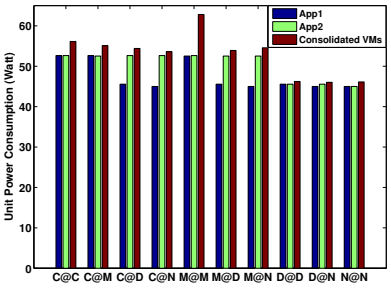


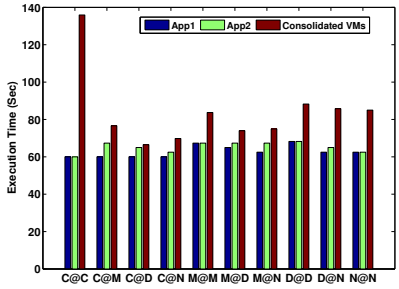
Figure 3: Power Consumption with Different Levels of CPU, Memory, Disk, and Network Usage

We present the experimental results in Figure 3. Unit power is reported here, and it denotes the power consumed every 1 second. When the system is idle, the power consumption is 32 watt. We varied the CPU usage from 20%, 40%, 80%, to 100% in the CPU-bound workload set. As we can see, CPU utilization has a significant impact on the consumed power. Between a workload where CPU activity is 20% and the workload where the CPU activity is 100%, power consumption increases by a factor of 64%. However, we can also see that lowering CPU activity does not result in a proportional decrease in power consumption. If the CPU activity is lowered to one fifth of the peak capacity of the system, the power savings are only close to 40% of the peak power.

In the memory-bound set, the memory utilization is 10%, 20%,

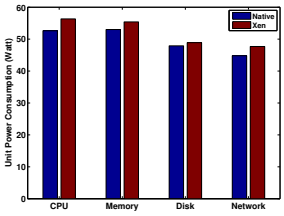


(a)

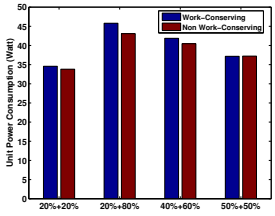


(b)

Figure 5: Impact of Consolidation of Workloads(C:CPU, M:memory, D:disk, N:network -intensive): (a) Power Consumption (b) Execution Time



(a)



(b)

Figure 4: (a) Unit Power Consumption Comparison: Native vs. Xen (b) Credit Scheduler: Work-Conserving vs. Non Work-Conserving

40%, and 80%, for the four workloads. The CPU activity is fixed at 70%, and thus, all four workloads in this set corresspond to a fairly high CPU load. As we can see, a larger memory footprints results in higher power consumption. Furthermore, during our experiments, we have also observed that varying cache activity levels, while keeping the same total memory footprint, impacts power consumption, and as expected, more power is consumed when cache miss rate is high.

Next, we consider disk activity of the workloads. Within the set, all workloads involved disk activity, but we varied the disk read rates. CPU usage is fixed at 50% and memory usage is less than 1%. We can observe that the consumed power remains constant as disk read rates are varied. More analysis demonstrated that power consumption is reduced if the disk is completely idle for long periods of time.

Finally, network-bound workload set is studied. CPU and memory usage is fixed at 50% and 10%, respectively. Disk activity is 18 MB/s. Similar to disk activity, presence of network I/O adds a constant overhead to the unit power consumption. However, the frequency of network I/O itself does not impact power consumption.

Next, we analyzed the power overheads of use of virtualization technologies (particularly, Xen). The results are shown in Figure 4(a). We chose one of the workloads from the four sets we had studied, i.e., CPU-bound, memory-bound, disk-bound, and Network-bound sets. We compared the unit power consumption by executing the workload on the same PC with and without Xen VM's enabled. Our results show that Xen VMs pose less than 7% of power overhead. This demonstrates that Xen VMs incur negligible overhead in terms of unit power consumption. Furthermore, we study the power consumption with the credit-scheduler of Xen [6]. We pined 2 VMs to a physical core, each of which has a different CPU requirement. In the *Work-Conserving* mode, the CPU requirement of each workload will be allowed whenever there is idle CPU. While using the *Non Work-Conserving* mode with the *cap* set, the workload cannot access more CPU than what has been limited by the cap value. As we can see from Figure 4(b), when there is a contention for CPU cycles, the power consumed by using the *Non Work-Conserving* mode is less than what is consumed in the *Work-Conserving* mode.

Overall, we can make several observations from our experiments. First, all of the resource activities impact power consumption. Variation in the CPU utilization has the largest impact on total power consumption, though memory footprint and cache activity also impact the consumed power. Enabling virtualization does incur some additional overheads, though these overheads are small. Finally, when there is CPU contention between Xen VMs, dynamic provisioning of CPU at runtime saves power.

3.2 Impact of Consolidation and Workload Correlation

We next study the impact of consolidating different types of workloads on a single physical machine, with the of virtualization technologies. We study both the execution time and the rate of power consumption (i.e., the unit power) after consolidation of workloads with different resource usage profiles.

We take four workloads, which could be considered as *compute-intensive* (C), *memory-intensive* (M), *disk-intensive* (D), and *network-intensive* (N), respectively. For each distinct pair from these four workloads (10 pairs in all), we compared two executions. The first execution involves running each workload individually on the research PC we used in Subsection 3.1. This is referred to as `App1` and `App2`. Then, we place each workload on a VM, and execute 2 VMs on the same core of the same machine. This execution is referred to as `Consolidated VMs`. The unit power consumption and the total execution time is reported in Figure 5, sub-figures (a) and (b), respectively.

From Figure 5 (a), we can see that consumed unit power from consolidation is up to 55% lower that the aggregate power used by executing workloads individually. Consolidated VMs can incur more power, but the extra power is negligible except in one case. When two memory-intensive workloads are co-located, frequent memory accesses from both of them increase the cache miss rates, resulting in higher unit power consumption.

Figure 5 (b) shows the execution times. The results show that when dissimilar workloads are consolidated, we are able to execute the application with only at most 10% slowdown as compared to the execution time of a single application. In other words, if we had only a single server for execution of the two workloads, consolidation of dissimilar workloads leads to a significant reduction in the total execution time. Another way of viewing these results is as follows. Compared to the case when we could use two independent servers for execution of these workloads, we are incurring only a small slowdown in the execution time, and large saving in resource and power costs.

As expected, consolidating workloads with similar requirements is not beneficial. For example, when we consolidate two CPU-intensive

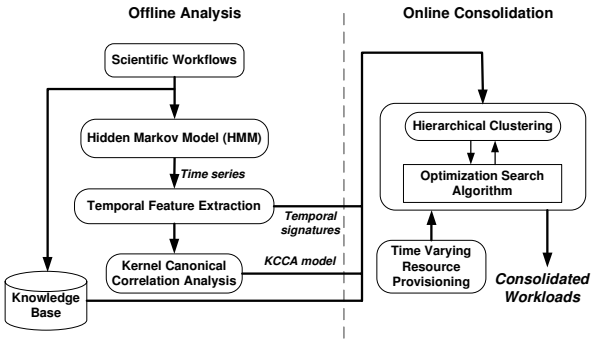


Figure 6: The pSciMapper Framework Design

workloads, the execution time doubled. Also, for a pair of memory-intensive workloads, the execution time increased by 41%. Consolidation of two workloads that have a high disk or network activity also leads to significant slowdowns. In these cases, even if the unit power does not increase, the total power consumption will increase because the same power is being consumed for a longer duration.

Overall, our results indicate that consolidation of workloads with dissimilar requirements can reduce the total power consumption and resource requirements significantly, without a substantial degradation in performance.

4. DESIGN OF PSCIMAPPER

This section presents the framework, *pSciMapper*, that we have developed to consolidate scientific workflows in virtualized environments. We first give an overview of our approach. Next, we discuss how we extract key temporal information from the given resource utilization profiles (time-series) for each task of the workflow, and further relate it with application performance and power consumption. Then, we present our power-aware consolidation algorithm, which places tasks of a scientific workflow on servers so that power consumption is minimized, without a substantial increase in the completion time. Finally, towards the end of this section, we describe how we learn a model capturing the resource usage profile's dependence on the application parameters and the characteristics of the resources on which the workflow is executed.

4.1 Overview

In order to reduce the total power in a computing environment, effective power management strategies have been developed [22, 50, 44, 39]. Dynamic Voltage and Frequency Scaling (DVFS) is a widely applied technique to minimize the processor power by scaling down CPU frequencies. We use server consolidation in our work, which is based on the characteristics of the workflows we have observed. Unlike the previous work on consolidating workloads [48, 43], we not only specifically focus on scientific workflows (DAGs), but also consider a more complex consolidation problem. We investigate server heterogeneity, with respect to CPU, memory, disk, and network bandwidths, as well as the application requirements for these resources. We also study the interference caused by resource contention when multiple workflow tasks are placed on a single server, and how it impacts the power consumption and application performance.

The overall design of pSciMapper is illustrated in Figure 6. It consists of two major components: *online consolidation* and *offline analysis*. The online consolidation algorithm assumes that for each task of the workflow to be scheduled, a set of time series data representing the CPU, memory, disk, and network utilization is available. In practice, such specific information may not be available for the given hardware and the application parameters without executing the workflow. We train a hidden Markov model (HMM) to predict such time series, which we describe towards the end of this section.

Now, for consolidation, we extract key *temporal features* from the time-series. This results in 52 features per workflow task , not all of which may be equally important for predicting the execution time and power consumption. We use a dimensionality reduction technique, Kernel Canonical Correlation Analysis (KCCA) [7], to relate the key features to the execution time and power consumption.

Next, the online consolidation algorithm uses hierarchical clustering [25] and the Nelder-Mead optimization algorithm [35], to search for the optimal consolidation, where power requirements are minimized, without a substantial impact on performance. As an enhancement to our static consolidation approach, we also perform *time-varying resource provisioning* based on the dynamic resource requirements of the workflow tasks.

4.2 Temporal Feature Extraction and Kernel Canonical Correlation Analysis

The relationships between resource usage time series and a workflow task's execution time and its power consumption are complex. To facilitate our analysis, we use a specific *temporal signature* to capture the key characteristics of a given time series.

Temporal Signature: Given a resource usage time series, which specifies the resource requirements with respect to a specific resource (CPU, memory, disk, or network) of a workflow task, we extract the following three sets of information: 1) *Peak value*: We define it as the max value of the time series data and denote it as max. 2) *Relative variance*: We compute the sample variance of the data, denoted as v^2. However, the sample variance depends on the scale of the measurement of a particular resource. Therefore, we transform the v^2 values to a normalized $[0, 1]$ space, with 0 being the lowest variance and 1 being the highest. This value serves as an indicator of fluctuation around the average resource utilization of the workflow. 3) *Pattern*: Besides peak and variance, we need to capture how the resource usage varies around an expected baseline. We take the repetitive pattern of the time series data and generate a sequence of samples, choosing the sampling rate is a way such that the information from the original time series is preserved as much as possible [27]. In this work, the pattern is represented by 10 numbers, $p_1, p_2, ..., p_{10}$.

Now the temporal signature for a resource utilization time series data (T_i) can be represented as

$$T_i =< max_i, v_i^2, p_{1,i}, p_{2,i}, ..., p_{10,i} > \quad (1)$$

As the last part of the offline analysis in pSciMapper, we need to relate the usage of resources with the workflow task's performance and power consumption when it is mapped onto a server. We train a model for this purpose. The input to such model is the temporal signatures we extracted from the CPU, memory, disk, and network utilization time series, plus resource variables representing the host server capacity. The model outputs time and consumed power for a task of the scientific workflow on this server. Thus, we have 52 features in the input space and 2 features in the output space. In our work, we have used Kernel Canonical Correlation Analysis (KCCA) [7] for modeling the *resource-time* and *resource-power* relationships.

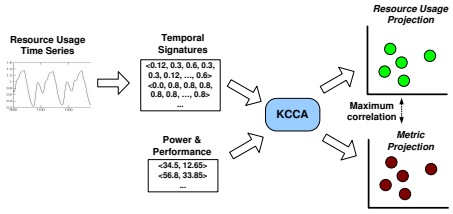


Figure 7: KCCA Training: From Vectors of Resource Usage Features to Vectors of Metric Features

Algorithm 4.1: WORKFLOWCONSOLIDATION(C, S)

INPUT C: the set of components
S: the set of servers
OUTPUT $\hat{\Theta}$: the consolidation plan
// initial assignment of components
$InitialCluster(C, S)$;
// generate resource usage profile
for each C_i and S_j
$GenerateTS(C_i, S_j)$;
$\Theta = OptimalSearch(C, S)$;
while $(true)$
for each $cluster_i$ and $cluster_j$
//calculate distance between two clusters
$linkage_{i,j} = CalculateDist(cluster_i, cluster_j)$;
if $linkage_{i,j} < merge_{threshold}$
// consolidate workloads in two clusters
$MergeCluster(cluster_i, cluster_j)$
$\Theta = OptimalSearch(C, S)$;
if (stopping criteria is met)
$\hat{\Theta} = \Theta$;
$break$;
$TimeVaryingResourceProvisioning(\hat{\Theta})$;

Figure 8: Workflow Consolidation Algorithm

KCCA: Canonical Correlation Analysis (CCA) is a dimensionality reduction technique that is useful to determine the correlation between two sets of multi-variate datasets. KCCA is a generalization of CCA using kernel functions to model nonlinear relationships [7]. Given N temporal signatures from resource usage profiles, we form an $N \times N$ matrix K_x, where the (i, j)th entry is the kernel evaluation $k_x(x_i, x_j)$. Specifically, we have used the Gaussian kernel [37]. Similarly, we also form an $N \times N$ matrix K_y, where the (i, j)th element is the kernel evaluation $k_y(y_i, y_j)$ with respect to application performance and power consumption. The KCCA algorithm takes the matrices K_x and K_y, and solves the following generalized eigenvector problem:

$$\begin{bmatrix} 0 & K_xK_y \\ K_yK_x & 0 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \lambda \begin{bmatrix} K_xK_x & 0 \\ 0 & K_yK_y \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \quad (2)$$

This procedure finds subspaces in the linear space spanned by the eigenfunctions of the kernel functions such that projections onto these subspaces are maximally correlated [7]. We refer to such projections as *resource usage projection* and *metric projection*, respectively. We illustrate such projections in Figure 7. If the linear space associated with the Gaussian kernel can be understood as clusters in the original feature space, then KCCA finds correlated pairs of clusters in the resource usage vector space and the performance/power vector space.

4.3 Power-Aware Consolidation

We now present the details of the online consolidation algorithm in our pSciMapper framework. The main algorithm is summarized in Figure 8. For generality, we assume that available servers may be heterogeneous. We first randomly assign each of the workflow tasks onto an underlying server. We assume for now that CPU, memory, disk, and network usage time-series for executing the task on this particular server is available. In practice, this is obtained using the HMM, which we will describe later.

Our consolidation problem can be viewed as a *clustering* problem. Normally, clustering groups similar objects together. Here, our goal is to group workflow tasks that are dissimilar in their resource requirements. This can be handled by defining a distance measure that focuses on the interference between the resource requirements of the workflow tasks.

Particularly, we apply the agglomerative hierarchical clustering algorithm [25] using the extracted temporal signature vectors. The consolidation procedure is iterative and it starts from the bottom of the hierarchical cluster structure, i.e., when each cluster is formed by a single workflow task. We decide whether two tasks, i and j, can be consolidated by calculating the *distance* between them. To be specific, we define the distance metric as the following.

$$Dist_{i,j} = \sum_{R_1,R_2} aff\_score(R_1, R_2) \times Corr(peak_i^{R_1}, peak_j^{R_1}) \times Corr(peak_i^{R_2}, peak_j^{R_2}) \quad (3)$$

, where R_1, R_2 denotes any type of resources (thus 10 pairs in total) and $Corr(peak_i^{R_1}, peak_j^{R_1})$ is the Pearson's correlation between two workloads with regards to the resource usage of R_1. We argue that it is more important to consider the correlation of the peak values rather than the whole temporal signature. The aff_score is pre-specified based on our correlation analysis in Section 3.2. It varies between 0 and 1, with 0 denoting the least interference when consuming two resources $R1$ and $R2$ together. The DAG structure has to be considered here. Although combining two workloads with active network activities could incur significant computing overhead, communication cost can be reduced if there is a direct dependency between them. Thus aff_score is small for two network-intensive workloads with a *parent-child* relationship. We merge two clusters if the distance between them is smaller than a threshold, i.e . $merge\_threshold$ in the pseudo-code.

Next, a well-known optimization algorithm, the Nelder-Mead [35], is applied to identify how the clusters obtained can be mapped to the given set of servers. The goal here is to minimize the power, without incurring a significant performance degradation. The consumed power and execution time for the one-to-one mapping of clusters to the servers can be estimated using the KCCA model trained offline. To be specific, the input vector, which includes the temporal signature of the resource usage profile along with the server capacity, is projected into the resource subspace. We then infer the coordinates in the metric subspace using the k nearest neighbors, where $k = 3$ in our implementation. Finally, we map the metric projection back to the metrics, which are the consumed power and the execution time. A weighted sum of the metric projections from the nearest k neighbors has been used, with the weight to be the reverse distance between coordinates of two projections in the subspace. The optimal point of this iteration with the minimum total power consumption is recorded.

Then, temporal signature of the new cluster is updated from the consolidated workloads. Such consolidation iterations stop when the clusters cannot be merged anymore since merging will incur significant interference, and/or the degradation in application performance will be intolerable.

As an enhancement to the static consolidation algorithm, we further take into account the dynamic resource requirements in the consolidated workloads to save power (Section 3.1). The reasons is that co-located workloads could peak together, or the resource usage from different workloads that we have co-located can vary. For example, the CPU utilization of a workload changes from 30% to 60% while its co-located workload drops from 50% to 30%. To support such dynamics, we propose a heuristic method for performing the time varying resource provisioning for each of the consolidated servers. Specifically, we make a reservation plan that simply allocates virtual CPU and memory (currently, Xen only supports runtime changes to CPU and memory) to individual workloads. Clearly, the total requirements from these two resources should not exceed what is available on the physical server. However, if we are unable to meet the peak requirements, resources are allocated to workloads in proportion to their peak values. Note that such resource reallocation

only takes action at the time instances where there are significant changes to the resource requirements, since dynamically changing the resource allocations frequently can involve significant overheads too.

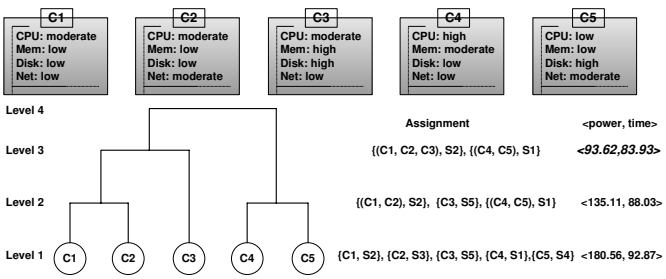


Figure 9: Consolidating Workflows: An Example

Example: We use an example to explain our consolidation algorithm, as illustrated in Figure 9. Assume that the workflow has 5 tasks, $C_1, C_2, ..., C_5$ and originally it is on 5 servers, $S_1, S_2, ...$, S_5. The resource usage profile for each task is known in advance. At the beginning of the consolidation algorithm, each C_i at level 1 is randomly assigned to a server S_j. Then the optimization algorithm generates an assignment where the power consumption is the minimum at the current level. Then we move up to level 2, where C_1 and C_2, and, C_4 and C_5 are merged as new clusters. This is consistent with what we expect by observing the resource utilization characteristics. The optimal search will find the servers for the consolidated workloads. Now the power consumption drops significantly as we now are using only 3 servers. This procedure repeats in level 3, where C_3 is merged with the cluster which contains C_1 and C_2. Note that due to the high memory footprint of C_3 and intensive I/O activities, serious interference would occur if it is co-located with C_4 or C_5. Now, 5 tasks of the workflow are consolidated on 2 servers and we are able to save half of the power with only a 9% performance degradation. Since no tasks can be consolidated anymore, our algorithm will generate C_1, C_2, and C_3 on S_2, C_4 and C_5 on S_1 as the output.

4.4 Hidden Markov Model for Resource Usage Estimate

The approach presented so far assumes that when a workflow is submitted, along with a set of application parameters, we know the time series of CPU, memory, disk I/O, and network I/O usage for each workflow task on any of the servers in our environment. In practice, however, it is impossible to know this information. We now describe how we train a hidden markov model (HMM) based on a representative set of application parameters and hardware specifications. Using this HMM, resource usage time series can be generated for given application parameters and hardware specification.

A hidden Markov model(HMM) is a statistical model that assumes that a sequence of observations $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, ...\mathcal{O}_T$ is generated from a set of N states ($X = x_1, x_2, ..., x_n$), with transition probabilities, $P(x_t|x_{t-1})$, between them, and emission probabilities, $P(\mathcal{O}_t|x_t)$, denoting the probability of an observation $\mathcal{O}_t$ being generated from the state t. The probability of a particular state in an HMM is only dependent on the previous state. The probability of an observation is dependent only on the state that produced this observation.

We generate our HMM for resource utilization prediction in the following way.

- *Hidden states*: They represent equivalence classes in resource usage. To be specific, a hidden state x_i is of the format of $x_i =< cpu_i, mem_i, disk_i, net_i >$, denoting the CPU, memory utilization, disk I/O, and network I/O. One issue we had to address is that, in practice, each resource usage variable is a continuous number, which leads an infinite number of hidden states. We discretize the variable and thus make the number of states finite.
- *Observations*: At time step t, the observation $\mathcal{O}_t$ is a set that comprises of the values of the application parameters and the current resource availability of the hosting node, denoted as $\mathcal{O}_t =< para_t, resource_t >$.
- *State transition probability*: The transition matrix characterizes the distribution over the states for CPU, memory, disk, and network usage. This is learned during the training of HMM.
- *Emission probability given an hidden state*: They are modeled using per-state multivariate Gaussian distributions.

The HMM is trained offline so that we are able to predict the CPU, memory, disk, and network usage without executing the workflow task on the server.

5. EXPERIMENTAL EVALUATION

This section presents results from a number of experiments we conducted to evaluate our power-aware consolidation algorithm.

5.1 Algorithms Compared, Metrics and Goals

As a baseline for evaluating our proposed algorithm, we implemented an *optimal* approach. In this approach, the best consolidation plan is obtained with an exhaustive search, i.e., we try every combination of placing workloads together and executing them on different servers. The consolidation plan resulting in a minimum power and less than 15% performance degradation is taken as the result. Furthermore, once the workloads are consolidated, we use *work-conserving* scheme during the application execution so that resource requirements of a workflow task are always satisfied, as long as total requirements of consolidated tasks does not exceed available resources. This consolidation plan thus is obtained and the resulting execution is referred to as `Optimal + Work Conserving`.

We also evaluated two versions of our consolidation algorithm. In the first version, only the static consolidation was performed. Then, we statically assigned resources among consolidated workloads proportional to the mean of their resource requirements. This is denoted as `pSciMapper + Static Allocation`. In the second version, we apply the time varying resource provisioning part of our algorithm. We refer this to as `pSciMapper + Dynamic Provisioning`. Finally, the three versions thus obtained through consolidation are compared with the case where each workflow task on an individual server, without virtualization. This is denoted as `Without Consolidation`.

To evaluate the performance of our approach against the optimal and the case where consolidation is not used, we use the following two metrics:

- *Normalized Total Power Consumption*: This shows the power that has been saved from our power-aware consolidation algorithm, as a percentage of the total power that is consumed by `Without Consolidation` version.
- *Execution Time*: It is defined as the makespan of the workflows.

Using the optimal approach and the above two metrics, we designed the experiments with the following goals: 1) Demonstrate that our power-aware consolidation algorithm can reduce total power significantly without incurring substantial slowdowns. 2) Demonstrate that the overhead of our algorithm is negligible, and the algorithm is scalable to workflows with a large number of tasks. 3) Demonstrate that our proposed resource prediction model is effective in estimating resource usage. Also, models trained on one type of hardware can still be effective on another type of hardware.

Application	CPU	Memory	Disk	Network
GLFS	High	Moderate	Moderate	Low
VR	Moderate	High	Moderate	Moderate
SynApp1	Low	Low	High	High
SynApp2	Moderate	High	Moderate	Low
SynApp3	High	Moderate	Low	Low

Table 2: Resource Usage of Scientific Workflows

5.2 Experimental Setup and Applications

Our experiments are conducted using 2 Linux clusters, each of which consists of 64 computing nodes. One cluster has dual Opteron 250 (2.4GHz) processors with 8 MB L2 cache and 8 GB main memory, while the other has Intel Xeon CPU E5345 (2.33 GHz) nodes, comprising two quad-core CPUs, with 8 MB L2 cache and 6 GB main memory. Computing nodes are interconnected with switched 1 Gb/s Ethernet within each cluster. We chose Xen as the virtualization technology and we used Xen-enabled 3.0 SMP Linux kernel in a stock Fedora 5 distribution. On a single consolidated server, hardware resources are shared between the virtual machines that host application service tasks and the management domain (`dom0` in Xen terminology). Throughout our experiments, we restrict the management domain to use one physical core, thus isolating it and avoiding performance interference. The virtual machines hosting application services share the remaining physical cores. The placement of VMs are decided at the start of application execution. In order to facilitate hosting the scientific workflows, the VMs are created and customized for each application.

The experiments we report were conducted using two real applications and three synthetic workflows. Two real applications are `GLFS` and `Volume Rendering`. The first of these applications was described earlier in Section 2. `Volume Rendering` interactively creates a 2D projection of a large time-varying 3D data set (volume data) [19]. The workflow we implemented reads 7.5 GB of image data, stores the spatial and temporal information in a tree structure, and then renders the final image using composed unit images. `VR` is considered as a memory-intensive application as it consumes more than 1GB of physical memory for 76% of its execution time. In order to cover a wide range of scientific workflows with different resource requirements and scales, we also evaluated our consolidation algorithm using three synthetic scientific workflows. These synthetic scientific workflows are from a set of benchmark scientific applications developed by Bharathi *et al.* [9]. `GLFS` and `VR` are executed on our clusters (with Xen virtual machines), whereas simulations were used for the synthetic applications. We used GridSim [42] as the grid environment simulator and its extension, CloudSim [14], to simulate a virtualized environment. Table 2 shows the relative resource usage of the two real workflows and three synthetic workflows, with respect to CPU, memory, disk I/O, and network I/O.

Since we did not have physical access to these clusters to be able to attach the power meter, we used *power modeling* for estimating power consumption. Particularly, we apply a full-system power model based on high-level system utilization metrics [20]. We validated this power model by actual measurements on the PC in our research lab, and found it to be very accurate. Furthermore, as we report only normalized power comparison for all of our experiments, limitations of the power model have a negligible impact on our results.

5.3 Performance Comparison

We now evaluate our proposed power-aware consolidation algorithm against the optimal approach and the case without consolidation, with respect to two metrics, i.e., normalized total power consumption and execution time.

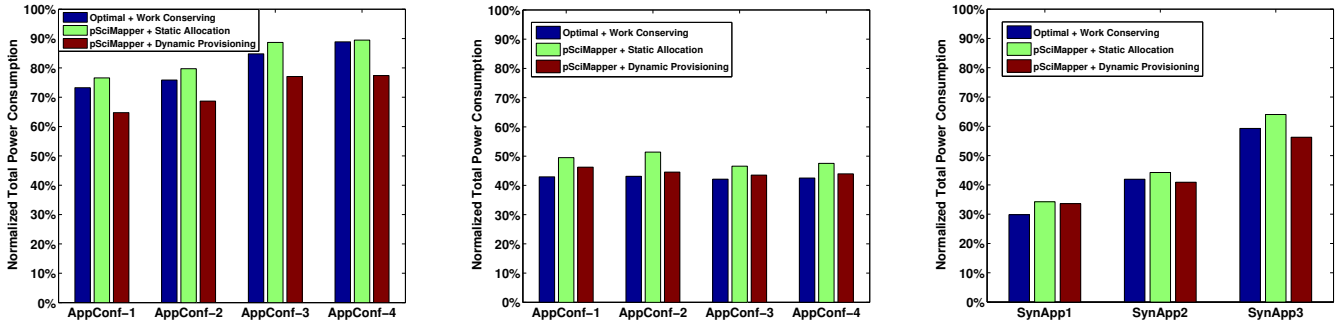
Normalized Total Power Consumption Comparison: We now evaluate our power-aware consolidation algorithm and demonstrate that the total power has been reduced significantly from the case no consolidation is applied. Moreover, we also show that power reductions are close to, or in some cases even better than, the optimal (but static) approach, as time varying resource provisioning is enabled through our approach.

First, we executed the `GLFS` workflow with four different combinations of application parameters, which are referred to as *AppConf-1*, *AppConf-2*, *AppConf-3*, and *AppConf-4*. For each of the parameter configurations, the workflow is invoked 5 times and the average is reported as the result. The normalized total power consumption, which is the ratio of consumed total power over that from the `Without Consolidation` version, is shown in Figure 10 (a). We can make the following observations from these results. First, the total power is reduced up to 27% by the `Optimal + Work Conserving` version. The CPU-bound characteristics of `GLFS` limit the consolidation that can be performed without significant performance degradation. In comparison, our `pSciMapper + Static Allocation` only performs 4% worse. Although the number of servers on which the `GLFS` workflow has been consolidated is the same as what is obtained from the optimal approach, the tasks that are merged into a cluster might not always be the optimal, due to the our distance metric definition.

Next, we consider `pSciMapper + Dynamic Provisioning`, where time varying resource provisioning heuristic is enabled. We are able to save up to 35% of the total consumed power, which is 8% better than the optimal (static) approach. This is consistent with our power analysis in Section 3.1, i.e., dynamically allocating CPU and memory to the consolidated tasks at runtime can help reduce the power, as compared to work conserving, particularly when the variance in resource requirement is large.

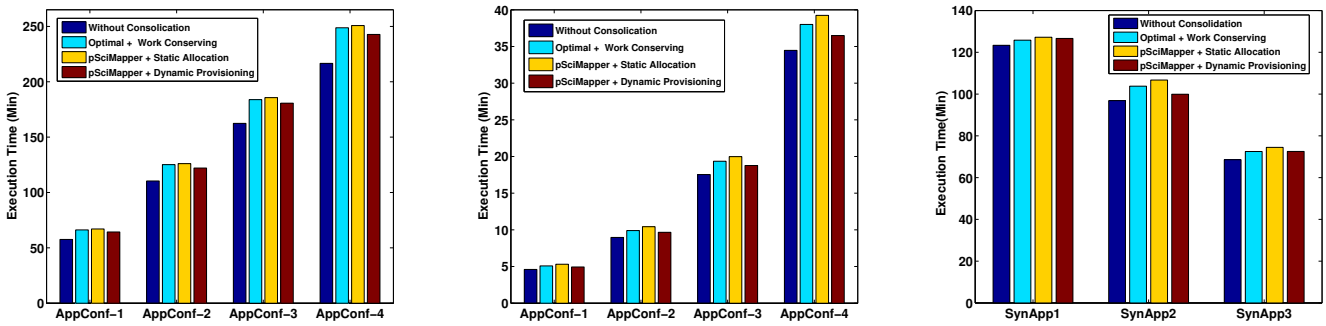
We repeated this experiment with the `VR` workflow and the three synthetic scientific workflows. The results are presented in Figures 10 (b) and (c), respectively. We executed `VR` with four different parameter configurations. As we can see, we are able to reduce 58% of the total power consumption in all cases, by applying the optimal consolidation approach. pSciMapper achieves 52% savings, and when time varying resource provisioning is enabled, pSciMapper performs only 2% worse than the optimal approach. Similar observations can be made from the three synthetic workflows. In the optimal case, more than 70% power is saved from `SynApp1` due to its large number of tasks with low CPU and memory usage. As `SynApp1`'s tasks have relatively constant resource requirements, dynamic resource provisioning does not help improve performance. pSciMapper, however, is still able to save 66% of the power. The other two synthetic workflows perform better with `pSciMapper + Dynamic Provisioning` when comparing to the optimal approach, and we are able to reduce the total power consumption by 59% and 44%, respectively.

Execution Time Comparison: Next, we compare the performance of pSciMapper with the other versions with respect of the workflow execution time. We first conducted the experiments using the `GLFS` workflow. Recall that we set performance degradation constraint to be 15%, i.e., consolidation stops when the estimated performance is still within 15% of the `Without Consolidation` case. As illustrated in Figure 11 (a), the execution time from the `Optimal + Work Conserving` case is within 15% of the case where each task of the workflow is mapped onto a single server. `pSciMapper + Static Allocation` also leads to performance that is close to the optimal case. This demonstrates the effectiveness of the temporal feature extraction and KCCA model in mapping resource usage to power and execution time. With time varying resource provisioning, pSciMapper leads to 3% better performance than the optimal case, and thus, the performance is within 12% of the `Without Consolidation` case. The experiment was repeated with the `VR` workflow and the other three synthetic workflows. The results are



(a) (b) (c)

Figure 10: Normalized Total Power Consumption Comparison: (a) `GLFS` (b) `VR` (c) Synthetic Workflows (Power Consumption of `Without Consolidation` is 100%)



(a) (b) (c)

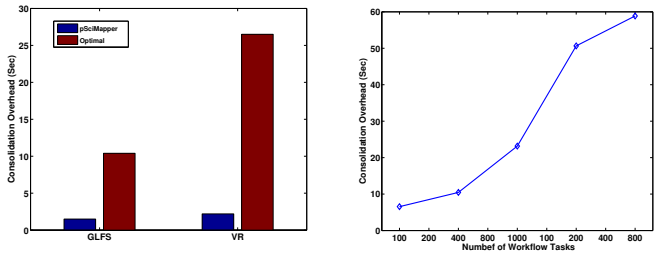
Figure 11: Execution Time Comparison: (a) `GLFS` (b) `VR` (c) Synthetic Workflows

demonstrated in Figure 11 (b) and (c). Similar observations can be made here. With time-varying resource allocation, pSciMapper results in only a 7% degradation.

5.4 Consolidation Overhead and Scalability

We now evaluate the execution time of our consolidation algorithm and compare it to the optimal approach. We used `GLFS` and the `VR` workflows, which have 16 and 24 tasks, respectively. The results are shown in Figure 12 (a). As shown in the Figure, our algorithm is able to consolidate 16 and 24 tasks onto 2 emulated grid sites, each with 64 nodes, with an algorithm execution time of 1.5 and 2.2 seconds, respectively. The overhead comes from the Nelder-Mead optimization algorithm at each level in the hierarchical cluster structure. In comparison, optimal algorithm involves an exhaustive search, and completes the consolidation in 10.4 and 26.5 seconds. The overhead of our algorithm is clearly very small, as compared to the actual execution time of any real scientific workflow. It may appear from these results that even the optimal algorithm is an acceptable choice for scheduling real workflows. However, the execution time of the optimal search will grow exponentially with respect to the number of tasks in the workflow and/or the number of servers. Therefore, its overhead will be unacceptable for consolidating large-scale scientific workflows, where hundreds or even thousands of tasks can be involved.

Furthermore, we evaluate the scalability of pSciMapper in Figure 12 (b). We used synthetic workflows and simulated 640 processing nodes for a grid computing environment. The number of tasks from the synthetic workflow is varied from 100 to 200, 400, 800, and 1000. We can see that the consolidation overhead only increases linearly as the number of workflow tasks increases, and it takes less than 1 minute to consolidate 1000 workflow tasks on 640 nodes.



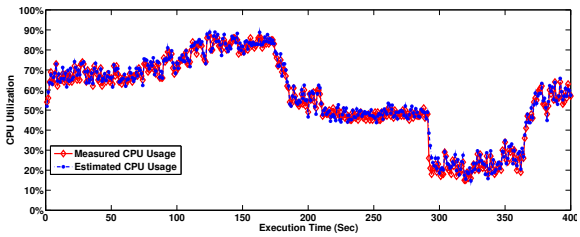
(a) (b)

Figure 12: (a) Consolidation Overhead Comparison of `pSciMapper` with `Optimal` (b)Scalability

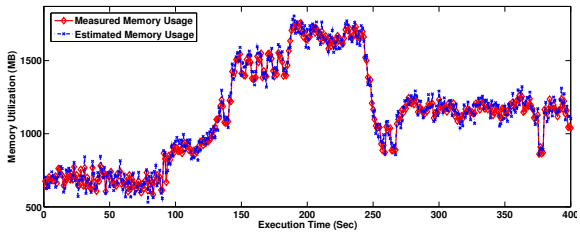
5.5 Model Evaluation

In this subsection, we evaluate the HMM-based model, and show that our model can accurately estimate the CPU, memory, disk and network usage, given the application parameters and the resource capacity of the host server. Furthermore, we show that the model can be trained on one server, and then used on a server with a different type of hardware effectively. This is an important requirement in cloud computing, as a service provider may use different types of hardware, with a user not having any control on what is made available to them.

We use tasks from both the `GLFS` and `VR` workflows. The data contains two sets of vectors. One includes the application parameter values and the resource capacity information of the server, while the other is the corresponding CPU, memory, disk and network utilization data. We performed *cross-validation* on the training data set



(a)



(b)

Figure 13: Resource Usage Measurement and Prediction Comparison: (a) CPU (b) Memory

and evaluate the model using new tasks from the two applications that have not been seen during the training. Note that our model is trained on a server with Intel Xeon quad-core CPU - 2.33 GHz, and then evaluated on a different server with AMD Opteron model 252 - 2.6 GHz. The results of CPU and memory predictions are presented in Figure 13 (a) and (b). As we can see, the average prediction error is 3.3%. Accurate resource usage prediction serves as an important step for our consolidation algorithm.

6. RELATED WORK

We now discuss the research efforts relevant to our work from the areas of scientific workflow scheduling and power management techniques.

Scientific Workflow Scheduling: Scientific workflows management and execution, particularly for heterogeneous and distributed environments like the computational grids, has been a popular topic of research in recent years. Workflow systems such as Pegasus [18], Kelper [31], ASKALON [51], Taverna [36], and Triana [46], are currently used in a number of scientific domains. As part of this research, several workflow scheduling algorithms have been developed, most of which focus on *makespan* minimization. Yan *et al.* proposed a workflow scheduler using data/network-aware execution planning and advanced reservation. VGrADS (previously GrADS) [32, 21, 41] first applies a ranking mechanism to assign each workflow job with a rank value, and then schedules these jobs based on their rank priorities.

None of these efforts have, however, considered consolidation of workflow tasks with virtualization technologies. In the future, our work can be combined together with workflow scheduling algorithm to achieve integrated scheduling and consolidation.

As scientific workflows are represented as a DAG structure, DAG scheduling is also related to our work. Comparison and evaluation of different DAG scheduling algorithms can be found in survey papers [15, 29]. Topcuouglu *et al.* proposed a Heterogeneous Earliest-Finish-Time (HEFT) algorithm, which minimizes the earliest finish time of priority-based tasks with an insertion-based approach [24]. As an improvement to the HEFT heuristic, Bittencourt *et al.* proposed to look ahead in the schedule and estimate a task by taking into account the impact on its children [11]. Again, our work is distinct in consolidating tasks in virtualized environments, and reducing total power requirements.

With the emergence of clouds, efforts have been initiated on executing scientific workflows on cloud environments, and evaluating the tradeoff between application performance and resource costs [17, 26, 23]. Our work focuses on virtualized environments, where either the users can control resource allocation for each VM, or a management layer knows the periodic resource requirements of a workflow. We expect that commercial cloud environments will have these characteristics in the near future.

Power Management: There have been two common mechanisms for power management, i.e., Dynamic Speed Scaling (DSS) and Dynamic Resource Sleeping (DRS). Dynamic Voltage and Frequency Scaling (DVFS), a typical example of DSS, is an effective technique to reduce processor power dissipation by either reducing the supply voltage or by slowing down the clock frequency [22, 50, 40, 38]. Wang *et al.* presented a cluster-level power controller based on multi-input-multi-output system model and predictive control theory, with the goal of shifting power among servers based on the performance needs of the applications [50]. DRS dynamically hibernates tasks to save energy and then wakes them up on demand [44, 39, 16].

Recently, power management for virtualized environments have received much attention [34, 47, 28, 45, 33]. Laszewski *et al.* propose a power-aware scheduling algorithm to schedule virtual machines in a DFVS-enabled cluster [30]. More closely related to our work are the efforts on power minimization using consolidation [48, 43]. Verma *et al.* propose two consolidation algorithms based on the detailed analysis of the workloads executed in data centers [48]. Particularly, they observed that 90-percentile of the CPU usage of the workloads should be used for consolidation, as it is less than half of the peak value. Furthermore, they also considered the correlation between workloads so that SLA violation could be avoided when such workloads are placed together. Our focus has been on scientific workflows, which have more complex structures and resource usage patterns. Besides CPU, we have taken memory, disk, and network usage into account. Furthermore, performance interference resulting from co-locating multiple workloads on the same server has been analyzed so that the consolidated workflows can achieve high performance with optimal power. Srikantaiah *et al.* proposed a multi-dimensional bin packing algorithm for energy-aware consolidation [43]. Our method, based on KCCA, is a more powerful model. Overall, we perform global optimization and can also consider time-varying provisioning of resources for meeting the requirements of consolidated workflow tasks.

7. CONCLUSION

Our work has been driven by two popular trends. First, scientific workflows have emerged as a key class of applications for enabling and accelerating scientific discoveries. Second, in executing any class of (high-end) applications, power consumption is becoming an increasingly important consideration.

In this paper, we consider the power management problem of executing scientific workflows in virtualized environments. We have developed a *pSciMapper*, a power-aware consolidation algorithm based on hierarchical clustering and a optimization search method. We have evaluated pSciMapper using 2 real and 3 synthetic scientific workflows. The experimental results demonstrate that we are able to reduce the total power consumption by up to 56% with at most a 15% slowdown for the workflows. Our consolidation algorithm also incurs low overheads, and is thus suitable for large-scale workflows.

8. REFERENCES

[1] Montage: An astronomical image engine. http://montage.ipac.caltech.edu/.

[2] Southern california earthquake center, community modeling environment (cme). http://www.scec.org/cme.

[3] Sysstat. http://pagesperso-orange.fr/sebastien.godard/.

[4] Teragrid. http://www.teragrid.org/.

[5] Usc epigenome center. http://epigenome.usc.edu.

[6] Xen credit-bsed cpu scheduler. http://wiki.xensource.com/xenwiki/CreditScheduler.

[7] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3, 2003.

[8] L.A. Barroso and U. Hölzle. The case for engergy-proportional computing. *IEEE Computer*, 14(12), 2007.

[9] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.H. Su, and K. Vahi. Characterization of scientific workflows. In *Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS08)*, pages 1–10, Nov. 2008.

[10] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*, 37(11), 2004.

[11] L.F. Bittencourt, R. Sakellariou, and E. Madeira. Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2010)*, Feb. 2010.

[12] D.A. Brown, P.R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb. A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. *Workflows for eScience*, 2006.

[13] E.S. Buneci and D.A. Reed. Analysis of application heartbeats: Learning structural and temporal features in time series data for identification of performance problems. In *Proceedings of the 22nd International Conference on High Performance Computing and Networking (SC08)*, pages 1–12, Nov. 2008.

[14] R. Buyya, R. Ranjan, and R.N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *Proceedings of the 7th International Conference on High Performance Computing & Simulation (HPCS09)*, June 2009.

[15] L.C. Canon, E. Jeannot, R. Sakellariou, and W. Zheng. Comparative evaluation of the robustness of dag scheduling heuristics. In *CoreGRID Technical Report TR-0120*, 2007.

[16] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP01)*, pages 103–116, Oct. 2001.

[17] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the montage example. In *Proceedings of the 22nd International Conference on High Performance Computing and Networking (SC08)*, pages 50–61, Nov. 2008.

[18] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, and D.S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3), 2005.

[19] R.A Drebin, L.Carpenter, and P.Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pages 65–74, 1988.

[20] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-system power analysis and modeling for server environments. In *Proceedings of the Workshop on Modeling Benchmarking and Simulation (MOBS06)*, June 2006.

[21] F.Berman, H.Casanova, A.Chien, K.Cooper, H.Dail, A.Dasgupta, W.Deng, J. Dongarra, L. Johnsson andK.Kennedy, C.Koelbel, B.Liu, X. Liu, A.Mandal, G.Marin, M.Mazina, J.Mellor-Crummey, C.Mendes, A. Olugbile, M.Patel, D.Reed, Z.Shi, O.Sievert, H.Xia, and A. YarKhan. New grid scheduling and rescheduling methods in the grads project. *International Journal of Parallel Programming*, 33(2), 2005.

[22] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. In *Proceedings of the 4th ACM European conference on Computer systems (EuroSys09)*, pages 317–330, April 2009.

[23] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the use of cloud computing for scientific workflows. In *Proceedings of the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES08)*, Dec. 2008.

[24] H.Topcuoglu, S.Hariri, and M.Y.Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13, 2002.

[25] S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3), 1966.

[26] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, and P. Maechling. Scientific workflow applications on amazon ec2. In *Workshop on Cloud-based Services and Applications in conjunction with 5th IEEE Internation Conference on e-Science (e-Science09)*, Dec. 2009.

[27] L.H. Koopsman. Sampling, aliasing and discrete-time models. *The Spectral Analysis of Time Series*, 1995.

[28] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, and G.F. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 5th International Conference on Autonomic Computing (ICAC08)*, pages 3–12, June 2008.

[29] Y.K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4), 1999.

[30] G.V. Laszewski, L.Z. Wang, A. Younge, and X. He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *Proceedings of the IEEE International Conference on Cluster Computing (Cluster09)*, pages 1–10, 2009.

[31] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice & Experience*, 18(10), 2006.

[32] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, and L. Johnsson. Scheduling strategies for mapping application workflows onto the grid. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing(HPDC05)*, pages 125–134, June 2005.

[33] R. Nathuji and K. Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP07)*, pages 265–278, Oct. 2007.

[34] R. Nathuji and K. Schwan. Vpm tokens: virtual machine-aware power budgeting in datacenters. In *Proceedings of the 17th IEEE International Symposium on High Performance Distributed Computing(HPDC08)*, pages 119–128, June 2008.

[35] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 1965.

[36] T. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18, 2006.

[37] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 3, 1962.

[38] P. Pillai and K.G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP01)*, pages 89–102, Oct. 2001.

[39] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. *ACM SIGMETRICS Performance Evaluation Review*, 34(1), 2006.

[40] C. Poellabauer, T. Zhang, S. Pande, and K. Schwan. An efficient frequency scaling approach for energy-aware embedded real-time systems. In *Proceedings of the 18th International Conference on Architecture of Computing Systems (ARCS05)*, pages 207–221, March 2005.

[41] L. Ramakrishnan, C. Koelbel, Y.S. Kee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T.M. Huang, K. Thyagaraja, and D. Zagorodnov. Vgrads: enabling e-science workflows on grids and clouds with fault tolerance. In *Proceedings of the 23rd International Conference on High Performance Computing and Networking (SC09)*, pages 1–12, Nov. 2009.

[42] R.Buyya and M.Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid compu ting. *Concurrency and Computation: Practice and Experience*,

14, 2002.

[43] S. Srikantaiah, A. Kansal, and F. Zhao. Energy. In *Workshop on Power Aware Computing and Systems (HotPower08)*, Dec. 2008.

[44] M. Steinder, I. Whalley, J.E. Hanson, and J.O. Kephart. Coordinated management of power usage and runtime performance. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS08)*, pages 387–394, April 2008.

[45] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *Proceedings of the 2007 USENIX Annual Technical Conference (USENIX07)*, pages 1–14, June 2007.

[46] I. Taylor, M. Shields, I. Wang, and A. Harrison. The triana workflow environment: Architecture and applications. *Workflows for e-Science*, 2007.

[47] A. Verma, P. Ahuja, and A. Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware08)*, pages 243–264, Sept. 2008.

[48] A. Verma, G. Dasgupta, T.K. Nayak, P. De, and R. Kothari. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 USENIX Annual Technical Conference (USENIX09)*, June 2009.

[49] F. Wang, Q. Xin, B. Hong, S. Brandt, E. Miller, D. Long, and T. McLarty. File system workload analysis for large scale scientific computing applications. In *Proceedings of the 21st IEEE / 12th NASA Goddard*, 2004.

[50] X.R. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA08)*, pages 101–110, Feb. 2008.

[51] M. Wieczorek, R. Prodan, and T. Fahringer. Scheduling of scientific workflows in the askalon grid environment. *ACM SIGMOD Record*, 34(3), 2005.