Towards Autonomic Workload Provisioning for Enterprise Grids and Clouds

Andres Quiroz, Hyunjoo Kim, Manish Parashar
NSF Center for Autonomic Computing
Department of Electrical & Computer Engineering
Rutgers, The State University of New Jersey
Email: {aquirozh,hyunjoo,parashar}@cac.rutgers.edu

Nathan Gnanasambandam, Naveen Sharma
Xerox Research
Xerox Corporation
Webster, NY, USA
Email: Fn.Ln@xerox.com

***Abstract*—This paper explores autonomic approaches for optimizing provisioning for heterogeneous workloads on enterprise Grids and clouds. Specifically, this paper presents a decentralized, robust online clustering approach that addresses the distributed nature of these environments, and can be used to detect patterns and trends, and use this information to optimize provisioning of virtual (VM) resources. It then presents a model-based approach for estimating application service time using long-term application performance monitoring, to provide feedback about the appropriateness of requested resources as well as the system's ability to meet QoS constraints and SLAs. Specifically for high-performance computing workloads, the use of a quadratic response surface model (QRSM) is justified with respect to traditional models, demonstrating the need for application-specific modeling. The proposed approaches are evaluated using a real computing center workload trace and the results demonstrate both their effectiveness and cost-efficiency.** [1]

I. Introduction

The emergence of cloud computing is leading to a paradigm shift in the way that research centers, enterprises and businesses are viewing their IT and computational needs, i.e., many organizations are considering using clouds to support a significant portion of these needs. This in turn is resulting in relatively long term, but highly varied and elastic (dynamically growing and shrinking) demands for cloud computational resources from various organizations of different types and sizes. At the cloud providers' end, consolidated and virtualized data centers attempt to exploit economies of scale and provide virtual machine (VM) containers to host a wide range of applications and provide users with an abstraction of unlimited computing resources - users can essentially "rent" VMs with required configurations to service their requests.

As scales, operating costs, and energy requirements of these data centers increase, maximizing efficiency, cost-effectiveness, and utilization becomes paramount. However, balancing QoS guarantees with efficiency and utilization becomes extremely challenging, especially when the workloads are highly dynamic and extremely varied. A typical workload in such a cloud infrastructure will consist of a dynamic mix of heterogeneous applications, such as long running computationally intensive jobs, bursty and response-time sensitive WS requests, and data and IO-intensive analytics tasks. Therefore, even though ensuring QoS and responsiveness can typically be accomplished using pre-allocated VMs and overprovisioning, the increasing difference between these resource allocations and application requirements magnifies resource under-utilization and leads to cost increases. Consequently, there is a growing need for providing more reactive on-demand provisioning, leading to better efficiency and utilization, but without an unacceptable impact on system responsiveness.

In a cloud environment, executing application requests on underlying Grid resources consists of two key steps. The first, which we call VM Provisioning, consists of creating VM instances to host each application request, matching the specific characteristics and requirements of the request. The second step is mapping and scheduling these requests onto distributed physical resources (Resource Provisioning). While much work to date has been dedicated to the resource provisioning problem, under-utilization of resources resulting from inappropriate VM provisioning will still occur. Most virtualized data centers currently provide a set of general-purpose VM classes with generic resource configurations, which quickly become insufficient to support the highly varied and interleaved workloads described above. For example, Amazon's EC2 only supports five basic VM types [1]. Furthermore, clients (enterprises and SMBs) can easily under or overestimate their needs because of a lack of understanding of application requirements due to application complexity and/or uncertainty.

The purpose of this paper is to present autonomic mechanisms for VM provisioning that addresses the unique challenges of enterprise Grids and cloud environments. The overall goal is to improve resource utilization, which is achieved by reducing overprovisioning at two levels. To reduce overprovisioning caused by the difference between the virtual resources allocated to VM instances and those requested by individual jobs, we develop a mechanism based on decentralized online clustering, which can efficiently characterize dynamic (rather than generic) classes of resource requirements and can be used for proactive VM provisioning. To address the inaccuracies in client resource requests that lead to overprovisioning, we

[1]The research presented in this paper is supported in part by National Science Foundation via grants numbers IIP 0758566, CCF-0833039, DMS-0835436, CNS 0426354, IIS 0430826, and CNS 0723594, by Department of Energy via the grant number DE-FG02-06ER54857, by a grant from UT Battelle, and by an IBM Faculty Award, and was conducted as part of the NSF Center for Autonomic Computing at Rutgers University.

explore the use of workload modeling techniques and their application to the highly varied workloads of cloud environments.

The specific contributions of this paper are the following: (1) the extension and application of a novel decentralized online clustering (DOC) mechanism for VM provisioning, and (2) the analysis of long-term, application-specific response time dynamics for high-performance computing tasks, aimed at demonstrating (a) the difference with commonly used workload models and therefore the need for application-specific workload modeling in mixed application environments and (b) the usefulness of workload modeling in providing feedback for VM provisioning. The paper is organized as follows. Section II first presents the specific characteristics and challenges of the autonomic mechanisms within the context of an end-to-end provisioning approach for enterprise Grids and clouds. Sections III and IV then describe, respectively, each of the proposed mechanisms in detail. Section V evaluates our approach using a real workload trace from a supercomputing center. Section VI discusses related work, after which Section VII presents our conclusions.

II. An End-to-End Provisioning Approach

Figure 1 illustrates an end-to-end provisioning approach, which identifies the different stages of the basic provisioning process along different spatial and temporal scales. First (1), broad workload trends and requirements across all application requests must be considered in the short term to perform provisioning of virtual resources, balancing overprovisioning and responsiveness; (2) provisioned VMs then need to be scheduled and mapped to physical resources according to data center wide availability, load-balancing, and long-term service-level constraints; (3) individual allocated VMs must then be managed at run-time within their local environments to ensure that their specific QoS and response time requirements are met, including possible re-scheduling and migration; (4) finally, the characteristics of the workloads of a specific application (client) over time must be monitored in order to construct models of the actual performance and response times achieved to provide feedback for input to the VM provisioning stage. Note that, although logically sequential for individual workloads, the aforementioned stages will be functioning in parallel and in different temporal and spatial scales within the data center. Also note that stages (2) and (3) both correspond to the resource provisioning problem and are consequently not addressed in this paper.

VM provisioning in stage (1) refers to the efficient allocation of virtual resources to application jobs as they arrive at service queues, through the creation and allocation of appropriately configured VM instances. Enterprise Grids hosting cloud services typically have multiple geographically distributed service providers and entry points, i.e., multiple queues may exist, with users submitting requests to their local application queues. The state of a system in terms of resource requirements is represented in the short-term by the job requests from different applications, as they wait in the multiple queues. A

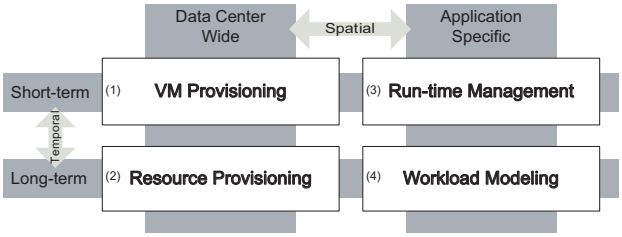

Fig. 1. Stages of data center provisioning

dynamic provisioning solution needs to create VM instances by closely approximating the specific resource requirements contained in this state. However, a reactive approach in which each individual request results in a custom VM allocation would be highly inefficient because of the significant wait times incurred in the instantiation of new VMs, in addition to the complexity of dynamically allocating physical resources for such a wide range of resource configurations. According to a recent study [2], the instantiation and startup times of VMs on Amazon's EC2 are in the order of 60–120 seconds. Thus, techniques are required to provide an online analysis and accurate characterization of the dynamic state in the form of a set of required resource configurations that can be efficiently provisioned and allocated in this time frame.

We claim that centralized analysis is not an appropriate solution for this purpose due to scalability, performance, and reliability concerns arising from the management of the multiple service queues and expected large volume of service requests. Thus, sustainable online approaches must be accomplished in a decentralized manner, preferably leveraging existing data center computing resources. We propose a mechanism for dynamic and decentralized VM provisioning, in which the flow of arriving jobs from different queues is analyzed in a decentralized manner during ongoing analysis windows of duration in the order of the startup time of new VMs. In each window, arriving jobs are analyzed, producing a number of VM classes that can be provisioned according to the volume of incoming requests of each class. At the same time, each job is assigned to an available VM class as it arrives, if one has been provisioned with sufficient resources to meet its requirements.

The decentralized, robust online clustering approach presented in this paper specifically addresses the distributed nature of enterprise Grids and clouds. The approach builds on a decentralized messaging and data analysis infrastructure that provides monitoring and density-based clustering capabilities. By clustering workload requests across data center job queues, the characterization of different resource classes can be accomplished online to provide autonomic VM provisioning. This approach has several advantages, including the capability of analyzing jobs across a dynamic set of distributed queues, the non-dependence on an a-priori knowledge of the number of clustering classes, and the amenity for online application and timely adaptation to changing workloads and resources.

The results of the analysis of resource requirements for VM provisioning can only be as good as the estimations of

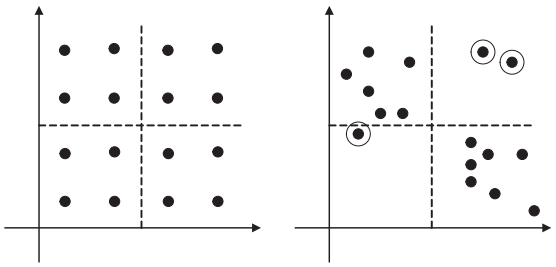

Fig. 2. (a) Uniform distribution of data points in the information space. (b) Point clustering and anomalies among regions; the circled points are potential anomalies.

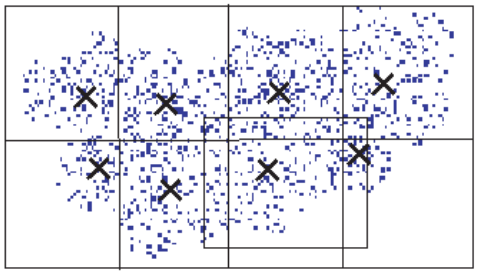

Fig. 3. Initial clustering result

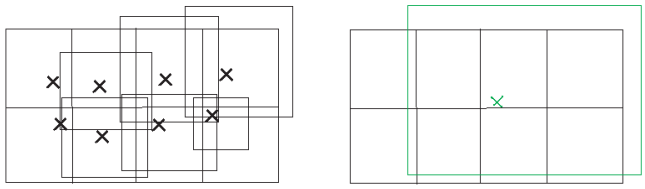

Fig. 4. Node 7 after one level of aggregation

these requirements provided by job requests, as they relate to performance and response times. Modeling the performance of the different individual types of applications (stage (4)) is therefore a good way of providing feedback for, as well as of autonomically transforming, client requests. For this approach, a broad range of analytical models and fitting techniques such as layered queuing models and regression time series can be used. These models in turn will be used to drive and possibly transform the input to VM provisioning, thus improving the efficacy of the decentralized mechanisms. Such improvements would better ensure the fulfillment of SLAs while reducing data center costs due to inaccurate resource requests. In this paper, we focus on models for high-performance workloads, which we have found to vary significantly from the traditional correlated workloads on one or more web-servers, as demonstrated in Section IV.

III. Decentralized Online Clustering (DOC)

A. Algorithm Description

Consider each job request represented as a *point* in a multidimensional space. Each dimension in this space, referred to as an *information space*, corresponds to one particular resource requirement, or attribute, and the location of a point within the space is determined by the values for each of the attributes within the particular request.

Our approach for cluster detection is based on evaluating the relative density of points (job requests) within the information space. In order to evaluate point density, the information space is divided into regions that are analyzed separately. If the total number of points in the information space is known, then a baseline density for a uniform distribution of points can be calculated and used to estimate an expected number of points per region. Clusters are recognized within a region if the region has a relatively larger point count than this expected value. Conversely, if the point count is smaller than expected, then these points may potentially be outliers or isolated points. These concepts are illustrated in Figure 2.

The approach described above lends itself to a decentralized implementation because each region is assigned to a particular *processing node*. Also, it can be done online because points are routed to processing nodes as they are produced from job requests, and processing nodes analyze the points within their region independently, performing a very lightweight computation (basically comparing point counts), and only communicating with nodes responsible for adjoining regions to deal with boundary conditions and for aggregating cluster data. We assume that the information space is predefined and globally known; however, the assignment of regions to nodes need not be static, and can thus be adapted to the arrival and departure of processing nodes in the network, as long as a mechanism exists to map data points to the node responsible for the region in which the point is located.

In previous work [3], we have been concerned with the accurate identification of clusters, and, more crucially, of outliers in system data. Therefore, the algorithm was designed to be capable of identifying cluster points, but not necessarily to consolidate the data from single clusters that may span multiple nodes in order to obtain a single set of descriptors for each cluster. Figure 3 shows the case in which a cluster lies across several nodes' regions, so that each node identifies the points within its region as clusters and calculates centroids separately. The problem is recognizing that all of the clusters in the separate regions are part of the same cluster, and to calculate a single centroid and range for it. Next, we explain the basic approach for tackling this specific problem.

Also shown in Figure 3 is the grid of processing node regions (referred to by node number 1 through 8). Each node that contains a local cluster issues a query to neighboring nodes, using a range determined by the size and density of its own cluster. Basically, the size of the range query is directly proportional to the size of the local cluster and inversely proportional to its density. The mechanism for mapping this query to neighboring nodes in the distributed network will be explained in Section III-B. Figure 3 shows the query issued by node 7.

The nodes whose regions are intersected by this query

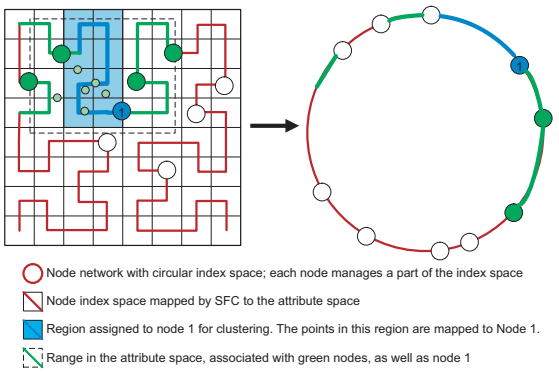

Fig. 5. Mapping from attribute space to node index space. Matching of cluster points and ranges can be done because intersecting regions are guaranteed to map to a non-empty set of common nodes, such as node 1 in the figure.

(nodes 2, 3, 4, 6, and 8 in the figure) respond with their own cluster centroids and cluster ranges, as shown in the left part of Figure 4. Upon receiving the replies to its query, each node will calculate a new centroid and query range by averaging the centroid values and by merging the cluster ranges together. This is shown on the right side of Figure 4 for node 7. Notice that the new centroid may not necessarily fall in the region of the node that issued the query. To continue the process, node 7 distributes the new cluster information using the new query range (received in this case by all remaining nodes). Nodes that receive this data will cache it to respond to queries by other nodes, and only the node that "owns" the new centroid (in whose region the centroid falls) will issue a new range query with the new box. This procedure is repeated until the expanding region covers the entire cluster.

B. Messaging Substrate

The exchange of job requests is supported by a robust content-based messaging substrate [4], [5], which also enables the partitioning of the information space into regions by implementing a dynamic mapping of points to processing nodes. The messaging substrate is responsible for getting the information used by the clustering analysis to the distributed processing nodes in a scalable fashion. The substrate essentially consists of a content-based Distributed Hashtable (DHT) that uses a Peano-Hilbert space-filling curve [6] as a locality preserving hash function. Content-based DHTs provide an interface that allows network nodes to be addressed by attribute-value pairs, so that, given a point with a value for each dimension (attribute), the DHT can find a route to the node responsible for the region containing that point. Additionally, our messaging substrate is able to resolve multidimensional range queries (in which a range of values is given for each dimension), guaranteeing the discovery of every node whose region intersects with the range of the query. Figure 5 shows how points, clusters, and ranges are mapped by the DHT to processing nodes connected by a ring overlay. A detailed analysis of the performance of content-based routing achieved by this substrate can be found in [4].

C. Cluster-based VM Provisioning

With the DOC algorithm and messaging infrastructure, the proposed mechanism for dynamic and decentralized VM provisioning is as follows. As with most predictive approaches, we divide the flow of arriving jobs into periods that we call analysis windows. In each window, an instance of the DOC algorithm is run on the jobs that arrive during that window, producing a number of clusters or VM classes. At the same time, each job is assigned to an available VM as it arrives, if one has been provisioned with sufficient resources to meet its requirements. The provisioning is done based on the most recent analysis results from the previous analysis window, as follows.

For the first window, the best option is to reactively create VMs for incoming jobs. However, the job descriptions are sent to the processing node network and by the end of the analysis window each node can quickly determine if a cluster exists in its particular region. If so, the node can locally trigger the creation of new VMs for the jobs in the next analysis window with similar resource requirements. According to [2], the time required to create batches of VMs in a cloud infrastructure does not differ significantly from the time for creating single VM instances. Thus, the VMs for each class can be provisioned within the given time window. Because the classes correspond to a characterization of incoming requests in the short term, it is expected that not all windows will require reprovisioning.

To clarify the above procedure, consider an example illustrated by Figure 6. The figure shows a two-dimensional information space with three initial clusters (squares). The top right-hand corner of these squares represents the resource values that characterize the class of jobs belonging to that cluster (because this amount of resources is sufficient for all of these jobs). The dotted lines represent the regions defined by each cluster that are mapped to the nodes that will contain the corresponding VM descriptors. Any job that falls within those regions (like the star in the figure) can be automatically assigned to the closest VM type (overlapping regions have more than one VM type description). The circle is a cluster that is discovered in a subsequent time window. This cluster can be used to create and provision a new VM type. The **x** is a request that falls outside existing regions (outlier) and represents a job whose requirements exceed the currently provisioned resources. In this case, the wait cost of reactively creating a new VM type would have to be incurred.

IV. Workload Modeling

The aforementioned VM provisioning model provides for each analysis window a set of VM classes, each with a given resource configuration, and a required number of VMs to be allocated to an expected job workload in the following window. However, this configuration does not guarantee an actual wall clock response time for this workload meeting existing requirements because the extent of achievable parallelism depends on the available resources. Frequently, overprovisioning will still occur from clients conservatively

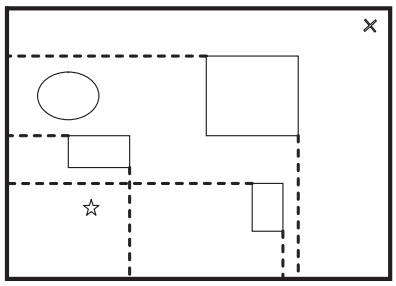

Fig. 6. Example information space and clustering results for dynamic workload provisioning.

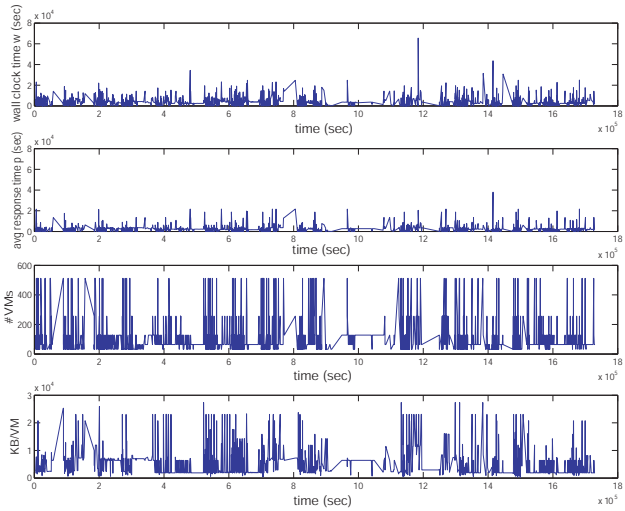

Fig. 7. Time Series plots

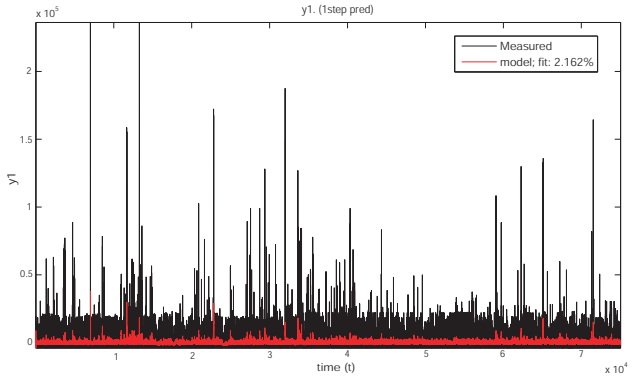

Fig. 8. Example ARMAX model exhibiting poor fit

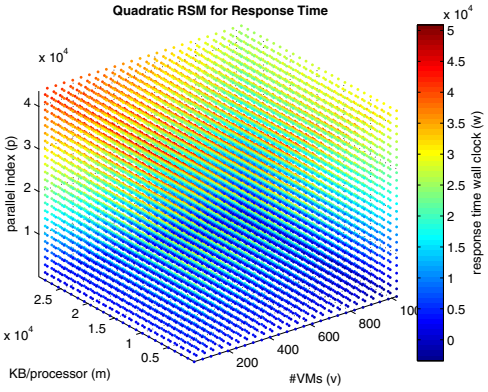

Fig. 9. Quadratic Response Surface Model

overestimating their needs, causing delays for subsequent loads as well as penalties for system underutilization. In order to model system response times with respect to workload, we examine the Los Alamos National Lab (LANL) CM-5 log, obtained from [7], which contains 201,387 jobs during two years, because computationally intensive applications are increasingly becoming part of enterprise datacenter and cloud workloads. We then aim to enhance the workload provisioning model by offering feedback to the requesters regarding achievable wall clock response times so that incoming job requirements can be refined. For example, provisioning can be performed using a higher or lower percentage of jobs' requested resources (memory or VMs) as input to avoid under or over system utilization, respectively.

The first step is to fit a model that best captures the behavior of response time from the actual resource allocations of incoming jobs for each type of application (e.g. transactional, computationally intensive etc.), using all of the data from the LANL data set. Figure 7 shows time series plots for the measured wall clock response times as a function of allocated parameters. We demonstrate this for a certain type of application workload which is computationally very intensive having high parallelization requirements. This workload varies significantly in terms of long-range correlations in the signals and non-linearity from the traditional workloads on one or more web-servers as seen in, for example, [8] or [9]. As a result, Box-Jenkins ARIMA or ARMAX [10] models provide poor fits when we consider workload provisioning across VMs for large run lengths and diverse sets of applications. From amongst various trials we show as an example in Figure 8 an ARMAX(5,4,3) that did not result in satisfactory estimation in terms of the goodness of fit. Therefore instead of adopting auto-regressive and/or moving average models, we take advantage of a quadratic response surface model (QRSM) [11], shown in Figure 9, to model the application specific dynamics in a given time window.

The data we analyze relates the achieved wall clock response time w with the number of virtual machines (v), the amount of memory alloted per VM in m (with units KB/VM) and the index of parallelism p (measured in terms of avg CPU time/VM). With this set of predictors, $\{v,\ m,\ p\}$ we aim to model the actual wall clock response time. In our non-linear model, we capture the achieved parallelization by accounting for the relationship between w and p in concert with other predictors. Formally the predicted response surface is,

$$\begin{aligned}\hat{w} &= b_0 + b_1 v + b_2 m + b_3 p + b_{12} vm + \\ &\quad b_{13} vp + b_{23} mp + b_{11} v^2 + b_{22} m^2 + b_{33} p^2. \quad (1)\end{aligned}$$

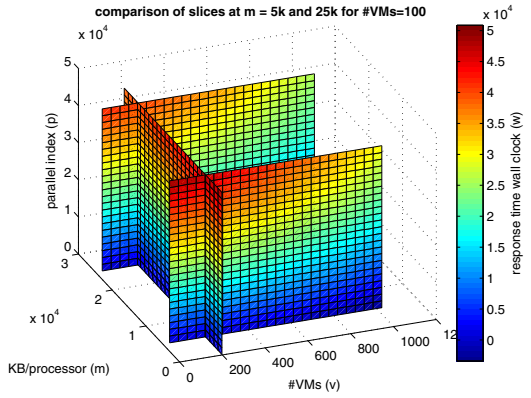

Fig. 10. Slices of QRSM model for alternative resource values

The parameters of the response surface obtained are (in order) 929.25, 2.832, -1.764e-4, 1.420, 1.272e-4, -3.666e-4, -6.989e-6, -6.903e-3, -1.087e-6 and -6.075e-6.

The estimated workload model is used in concert with cluster-based VM provisioning (see Section III) by applying it to each VM class. The model parameters v, m, and p are the number of VMs resulting in the analysis for the class and the memory and the CPU time requirements of the class, respectively. Given the total number of available CPUs and memory of the computing infrastructure, the model can be used to arrive at an actual response time $\hat{w}$ for the estimated workload that can be compared against response time requirements to adjust the class attributes. Given a target response time w^*, the model can also be used to suggest or automatically adjust job resource requirements. An example of such a recommendation is seen in Figure 10. The objective here is that we want to use the aforementioned model to remain within a region of a certain color (denoting response time) at a minimum cost (higher v, m, and p means greater cost). Following the color region along any dimension allows increasing or decreasing a resource without significantly impacting the response time (QoS) of the job. The particular example shows that, for an allocation of 200 VMs, the response time is not sensitive to the amount of memory used (the regions are approximately flat along that dimension), so that the fact that the memory requirements can be minimized without a significant impact on response time can be inferred.

V. Evaluation

We evaluate our approach with respect to two different types of cost corresponding to the tradeoff introduced in Section II. The first is the cost of assigning more resources to a VM than the job that runs on it requires, which we call overprovisioning cost. The second cost is the wait time between the moment that the job is received and when it starts running on a VM. If a VM is allocated specifically for each job (what we call a reactive approach), then every job will have to wait for the instantiation of its VM. We assume unsaturated resources for these experiments, so that enough resources can always be allocated to a VM for any job, and use the same LANL workload trace described in Section IV. Have in mind that using the same data does not invalidate our experiments, since the data contains both requested and observed values for each job, so that the latter is used for workload modeling, while clustering uses the former.

As described in Section II, we assume that jobs are submitted simultaneously to a number of service queues at a high rate, so that a considerable number of jobs can be queued within the system within each analysis window. The requests contain the jobs' resource requirements (memory, number of CPUs (N), and requested CPU time (T)). For clustering, we use memory and derive a value from the last two requirements that represents CPU speed (C). The value is calculated as:

$$C = \frac{N \times T}{100}$$

The value of 100 in the above calculation is a normalization factor that represents the duration in seconds of a generic or reference job. Multiplying the number of CPUs and requested CPU time gives a total CPU time requirement. The calculated CPU speed corresponds to the speed necessary to finish each job in the fixed reference time. The CPU time is used afterward with the clustering results to apply the response time model obtained in Section IV, the result of which is compared against the expected runtime of the set of jobs. We divide the dataset into several sequential subsets of job requests (of 1000 jobs each in our case, chosen as an extreme value), each of which represents the contents of the queues to be analyzed for provisioning within each analysis window.

The first experiment, which measures only overprovisioning cost, is meant to show the accuracy with which the different data analysis approaches can statically approximate the resource requirements of a job set. The DOC approach is compared to centralized k-means clustering, which is commonly used and is expected to provide a benchmark in terms of clustering accuracy. As a reference point, we also consider the overprovisioning approach. For the reactive approach, of course, no overprovisioning cost is ever incurred. The graphs in Figure 11 show the overprovisioning cost for resource R (CPU speed and memory) in each window i as the average difference between requested and provisioned resources for the set of jobs j in that window. This is:

$$O_i^R = \frac{1}{N} \sum_{j=1}^{N} \left(P_{ij}^R - Q_{ij}^R \right)$$

On average, both DOC and k-means approaches perform similarly, which is a good result, considering that decentralization and online execution must be traded for some accuracy in the clustering results. In fact, there is more variation in the results of k-means, which we attribute to the differences in the selection of k. In this experiment, we ran the k-means algorithm for a large range of k values and picked the best result for each run. This is a disadvantage with respect to the density-based approach, since this procedure for determining k values is usually done offline.

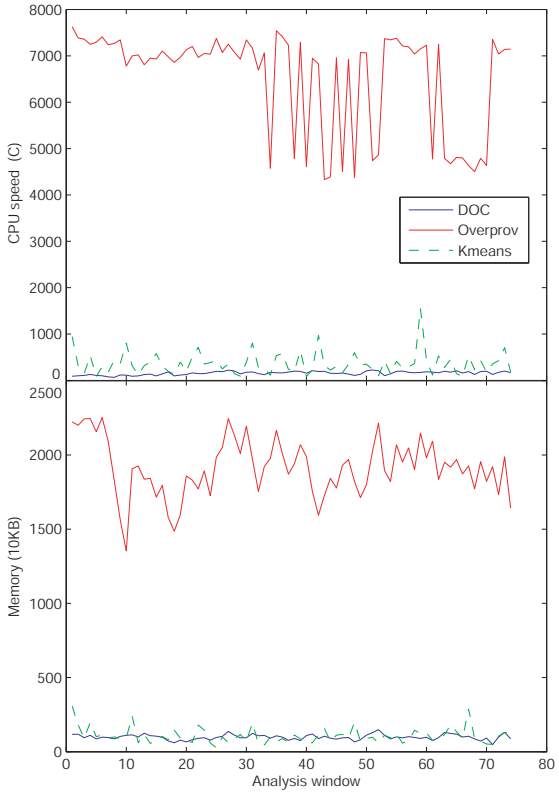

Fig. 11. Average difference between requested resource (CPU speed (top) and memory (bottom)) for jobs and the corresponding analysis result in each time window

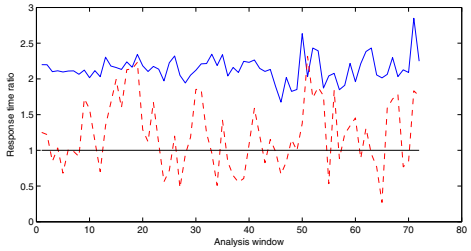

Fig. 12. Ratio of predicted vs expected runtime for job requests. The dashed and the steady line show, respectively, the ratio with and without the adjustment of clustering allocations based on the workload model.

Figure 12 shows the average ratio between the runtime predictions obtained for the different resource classes in each window and the requested runtime for the jobs in that window. The ratio oscillates around twice the requested runtime in each window considered, which indicates that the job requests should be adjusted with higher resources in order to meet the time constraints. When the model is applied to a higher allocation of VMs within each class, the runtime expectations move closer to the requirements, as shown by the dashed line.

The dynamic application of the different approaches with respect to provisioning response times is evaluated in the second experiment, in which the job sets form a sequence of incoming requests and the analysis results in each window are used for provisioning for the set of jobs in the next window.

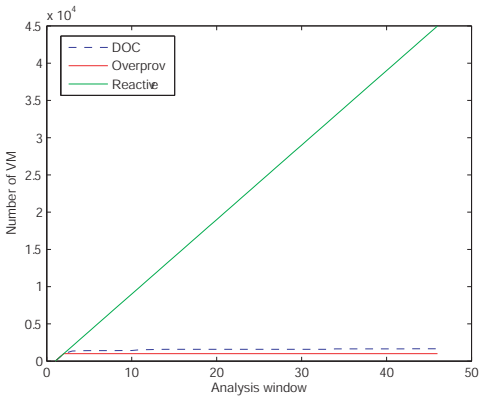

Fig. 13. VM creation cost vs. time for the different approaches

The impact on response time of each approach is measured by counting the number of VMs required by each job that could not be preallocated based on the predictions of that approach. This is because each such VM must be created after the job is received and not during the previous analysis window. Figure 13 compares the approaches that are amenable for online execution, which are the reactive, overprovisioning, and DOC approaches. In all of these, the VMs for the first set of jobs must be allocated reactively, so that all 1000 jobs count in the first analysis window. Afterward, however, the overprovisioning approach was always able to proactively allocate VMs for every job, so that its cost levels off at 1000. In contrast, all VMs created by the reactive approach are counted. The DOC approach performs very closely to the overprovisioning approach, incurring creation cost for outlying jobs or whenever the number of VMs provisioned for a resource class is not sufficient. The total count for the DOC approach is 1462 out of 45000 in the time frame shown.

VI. Related Work

Recent research has focused extensively on efficient and on-demand resource provisioning in response to dynamic workload changes. These techniques monitor workloads experienced by a set of servers or virtual machines and adjust the instantaneous resources availed by the individual servers or VMs. Menasce et al. [12] proposed an autonomic controller and showed how it can be used to dynamically allocate CPU resources in virtualized environments with varying workload levels by optimizing a global utility function using beam search. Soror et al. [13] introduced an offline virtualization design advisor that uses information about the anticipated workloads of a database system to recommend workload-specific configurations. ADVM [14] is an adaptive and dynamic scheme for adjusting resources (specifically, CPU and memory) between virtual machines on a single server.

Queuing and time-series models have been heavily employed in web-server environments with correlated arrivals. Chandra et al. [8] modeled resources using a time domain queuing model to capture the transient behavior of application workloads, and used prediction algorithms to estimate future

application workload parameters based on online measurements. This technique captures the transient behavior of application workloads when there are no long range interactions (i.e. higher order ARIMA processes). Kusic et al. [9] proposed a lookahead control algorithm and used it to identify optimal allocations of computing resources to multiple client QoS classes under correlated job arrivals modeled by ARIMA processes and service demand given by an exponentially weighted moving average filter. Xu et al. [15] also used a controller-based approach, in which a local controller is used to determine the resources required to satisfy application workloads on each virtual machine using fuzzy modeling and fuzzy prediction. Zhang et al. [16] defined a phase as a set of intervals within an application's execution that have similar system-level resource consumption behavior, and used k-means clustering to find the number of phases that gives minimal total costs. The number of phases is the best number of clusters.

For workload provisioning, several classic clustering algorithms such as k-means clustering and subtractive clustering [17] have been used. Since k is an input for the k-means algorithm, various k values must be tried to find the best k. Finally, Urgaonkar et al. [18] describe a dynamic provisioning approach for multi-tier Internet applications. It is similar to the present work in that it considers the dynamic assignment of a number of servers to the different application tiers, based on short and long term workload characteristics. The models used for long term provisioning in that work can be employed in this context for workload modeling of the particular set of applications for which they were devised. However, for short-term provisioning, a reactive approach is used, which is not able to account for the characteristics of the heterogeneous mix of applications in a cloud environment.

VII. Conclusion

This paper investigated an end-to-end autonomic approach for workload provisioning on enterprise Grids and clouds. It identified VM provisioning as a part of this problem that presents unique challenges for these environments because of a highly varied and dynamic mix of application types and workload demand. To deal with the underutilization of physical resources and increasing cost resulting from overprovisioning, it proposed the use of a decentralized, robust online clustering approach to drive VM provisioning. It also demonstrated a model-based approach for estimating the application service time given its provisioning in order to deal with inaccurate resource requirements provided by application job requests. The decentralized online clustering enabled the analysis of incoming jobs from multiple distributed queues and the characterization of workload classes from job resource requirements, which was used to dynamically provision virtual machines for future applications jobs. This approach was compared to overprovisioning and reactive approaches, as well as existing centralized approaches that use k-means clustering or model fitting, and was found to have comparable performance with the latter with a good tradeoff between overprovisioning cost and wait time overhead. The paper also presented a quadratic response surface model (QRSM), which was found to best capture the behavior of the workload used in the experiments, and demonstrated the need for application-specific and long-term modeling of application performance to be used in concert with short-term VM provisioning mechanisms. The comparison with other clustering and modeling techniques, besides k-means and QRSM, is the subject of ongoing work.

References

[1] "Amazon elastic compute cloud," http://aws.amazon.com/ec2/.

[2] S. Ostermann, A. Iosup, N. Yigibasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of cloud computing services for scientific computing," Delft University of Technology, Tech. Rep., December 2008.

[3] A. Quiroz, M. Parashar, N. Gnanasambandam, and N. Sharma, "Clustering analysis for the management of self-monitoring device networks," in *Proceedings of the International Conference on Autonomic Computing (ICAC 2008)*, Chicago, June 2008.

[4] C. Schmidt and M. Parashar, "Flexible information discovery in decentralized distributed systems," in *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12'03)*, 2003.

[5] A. Quiroz, N. Gnanasambandam, M. Parashar, and N. Sharma, "Robust clustering analysis for the management of self-monitoring distributed systems," *Journal of Cluster Computing*, vol. Online, pp. –, 2008.

[6] H. Sagan, *Space-Filling Curves*. Springer-Verlag, 1994.

[7] D. Feitelson, "The parallel workloads archive," http://www.cs.huji.ac.il/labs/parallel/workload/.

[8] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2003, pp. 300–301.

[9] D. Kusic and N. Kandasamy, "Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems," *Cluster Computing*, vol. 10, no. 4, pp. 395–408, 2007.

[10] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*, 3rd, Ed. Prentice Hall, 1994.

[11] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 3rd, Ed. John Wiley and Sons, 2009.

[12] D. A. Menasce and M. N. Bennani, "Autonomic virtualized environments," in *ICAS '06: Proceedings of the International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2006, p. 28.

[13] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, "Automatic virtual machine configuration for database workloads," in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2008, pp. 953–966.

[14] Y. Song, Y. Sun, H. Wang, and X. Song, "An adaptive resource flowing scheme amongst vms in a vm-based utility computing," *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, pp. 1053–1058, Oct. 2007.

[15] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, no. 3, pp. 213–227, 2008.

[16] J. Zhang, J. Kim, M. Yousif, R. Carpenter, and R. Figueiredo, "System-level performance phase characterization for on-demand resource provisioning," *Cluster Computing, 2007 IEEE International Conference on*, pp. 434–439, Sept. 2007.

[17] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent and Fuzzy System*, 1994.

[18] B. Urgaonkar and A. Chandra, "Dynamic provisioning of multi-tier internet applications," in *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 217–228.