Automated and On-Demand Provisioning of Virtual Machines for Database Applications

Piyush Shivam, Azbayar Demberel, Pradeep Gunda
David Irwin, Laura Grit, Aydan Yumerefendi
Shivnath Babu, and Jeff Chase

Dept. of Computer Science,
Duke University
Durham, NC, USA
{shivam,asic,pradeep,irwin,grit,aydan,shivnath,chase}@cs.duke.edu

ABSTRACT

Utility computing delivers compute and storage resources to applications as an 'on-demand utility', much like electricity, from a distributed collection of computing resources. There is great interest in running database applications on utility resources (e.g., Oracle's Grid initiative) due to reduced infrastructure and management costs, higher resource utilization, and the ability to handle sudden load surges. Virtual Machine (VM) technology offers powerful mechanisms to manage a utility resource infrastructure. However, provisioning VMs for applications to meet system performance goals, e.g., to meet service level agreements (SLAs), is an open problem. We are building two systems at Duke—Shirako and NIMO—that collectively address this problem.

Shirako is a toolkit for leasing VMs to an application from a utility resource infrastructure. NIMO learns application performance models using novel techniques based on active learning, and uses these models to guide VM provisioning in Shirako. We will demonstrate: (a) how NIMO learns performance models in an online and automatic fashion using active learning; and (b) how NIMO uses these models to do automated and on-demand provisioning of VMs in Shirako for two classes of database applications—multi-tier web services and computational science workflows.

Categories and Subject Descriptors: C.4 [Performance of Systems]: Modeling Techniques

General Terms: Measurement, Performance, Management

Keywords: Active Learning, Modeling, Virtual Machines

1. MOTIVATION

Utility computing delivers compute and storage resources to applications as an 'on-demand utility', similar to an electricity grid. The utility computing model is enabled by distributed collection of compute and storage resources spread over a local or a wide area network, i.e., networked utilities. Examples include scientific workflows running on computational grids and multi-tier web services (e.g., an ecommerce website) operating in data centers.

The utility setting offers several benefits for database applications: reduced infrastructure and management costs, higher resource utilization, and the ability to allocate resources on-demand to support dynamically changing demands. Oracle's Grid initiative is one recent example promoting utility resources for database applications.

Virtual machine (VM) technology offers powerful mechanisms for efficient management of networked utilities. After a decade of advances in VMs, robust and efficient VM systems are widely available and and are fast becoming ubiquitous. The leading VM systems (e.g., VMware, Xen) support live migration, checkpoint/restart, and fine-grained allocation of server resources as a measured and metered quantity [1]. These capabilities create a rich decision space for utility resource management: How should an intelligent infrastructure "turn the knobs" to map workload and resource requests onto a server network?

Intelligent provisioning of VMs is necessary to meet system performance goals such as meeting application SLAs, optimizing application execution time, and maximizing overall resource usage. However, provisioning VMs to meet such goals is challenging because application behavior is dictated by the interaction of several factors such as:

- **Resources.** The number of VMs assigned to the application, and the properties of the underlying resources bound to each VM, e.g., CPU, memory, storage, and network resources.
- **Data.** The characteristics of the data that the application processes such as the input data size, and its layout and partitioning on the storage server.
- **Workload.** The characteristics of the workload seen by the application, e.g., the request arrival rate, and the service demand of each request.

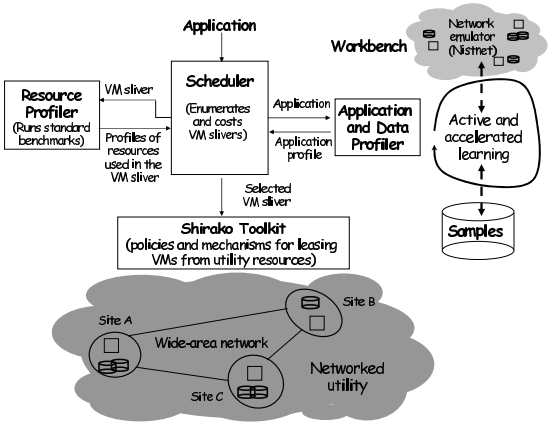
The NIMO system—focus of this demonstration—builds performance models that capture the interaction among such factors, and the resulting application performance [4, 6]. It uses these models to provision VMs in Shirako—a toolkit for leasing VMs from a shared utility resource infrastructure—to meet several performance goals in an automated and on-demand fashion.

2. SHIRAKO

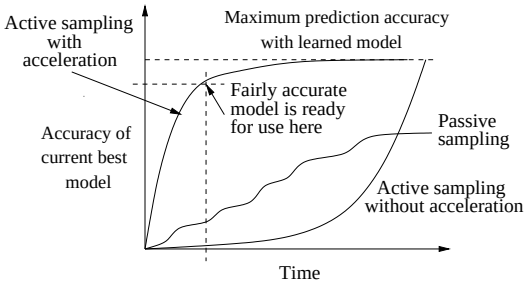
Shirako [1, 3] is a Java toolkit for secure, on-demand leasing of utility resources. Shirako provides a *guest* application with a *slice* of virtualized resources (VMs) from a collection of distributed physical resources.

Shirako has programmatic leasing primitives for dynamically partitioning a networked utility into Xen virtual machines. Each Xen VM is bound to a performance-isolated sliver of server resources; slivers are sized along multiple

SIGMOD'07, June 11–14, 2007, Beijing, China.
ACM 978-1-59593-686-8/07/0006.



1: Architecture of NIMO.



2: Active and accelerated learning.

dimensions (e.g., CPU cycles, memory, and network bandwidth), and can be resized on-the-fly to adapt to changing demands. Shirako, without NIMO, was demonstrated recently at OSDI'06 and SC'06 [2].

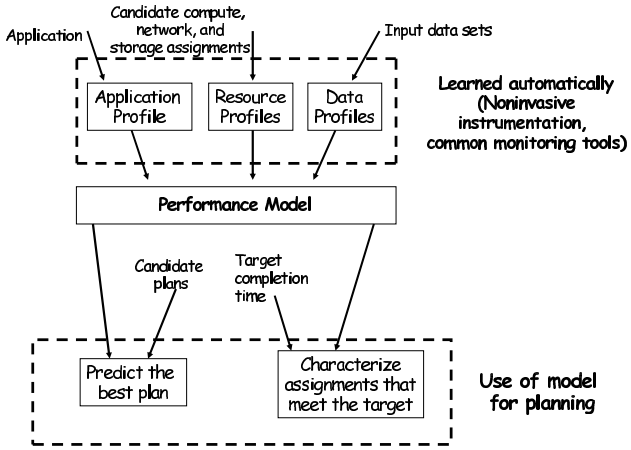
3. NIMO

NIMO [4, 5, 6] creates application performance models using active learning techniques automatically and quickly. NIMO has three objectives.

1. **End-to-End.** NIMO learns performance models that predict performance measures taking into account the interaction between an application's workload, the VM sliver(s) assigned to the application, and the data processed by it.
2. **Noninvasive.** NIMO gathers training data for models from passive instrumentation streams readily available with common tools, with no changes to application or system software.
3. **Active.** NIMO deploys and monitors applications on VMs with different sliver sizes to collect sufficient training data to learn accurate models.

4. NIMO OVERVIEW

Figure 1 shows NIMO's overall architecture. It consists of: (i) a *scheduler* that enumerates, selects, and requests VMs for applications from the Shirako toolkit; (ii) a modeling engine—consisting of a *resource profiler*, a *data profiler*, and an *application profiler*—that learns performance models for applications, and captures the behavior of the application resulting from the interaction of the relevant factors listed in Section 1; and (iii) a workbench where NIMO conducts proactive application runs to automatically collect samples for learning performance models. Active learning with *acceleration*, as shown in Figure 2, seeks to reduce the time



3: Model-guided resource planning.

before a reasonably accurate performance model is available. (The x-axis in Figure 2 shows the progress of time for collecting samples and learning models, and the y-axis shows the accuracy of the best model learned so far.)

We now summarize the components of NIMO in the context of a computational science workflow G [4]. Note that the demonstration will include computational science workflows and a multi-tier application.

4.1 Models

The scheduler uses a performance model $M(G, I, \vec{R})$ to estimate the performance of G with input dataset I on a resource assignment $\vec{R}$. Figure 3 gives an overview of our approach to estimate the performance of candidate plans. A *plan* consists of an assignment of a VM sliver(s) to an application by Shirako. NIMO builds profiles of resources and applications by analyzing instrumentation data gathered from previous runs of the application. A performance model M for an application G predicts the performance of a plan for G given three inputs: (i) G's *resource profile* of the VM assigned to the plan, (ii) *data profile* of the input data, and (iii) *application profile* of the application. Details on NIMO's performance models and profiles are in [6].

Intuitively, resource profiles specify attributes that characterize the function and power of the VM independent of applications. For example, a resource profile might represent a virtual machine with its share of CPU cycles, memory size, and network bandwidth. Similarly, storage resources can be approximated by attributes such as capacity, spindle count, seek time, and transfer speed. The data profile comprises the data characteristics of G's input data, e.g., the input data size. The application profile captures the behavior of the application resulting from the interaction of the application, the resources assigned to it, and its data profile.

4.2 Active Learning of Models

NIMO's modeling engine automatically learns the performance model for G from the instrumentation data samples obtained by deploying G on selected VMs with varying sliver sizes, either to serve a real request, or proactively to use idle or dedicated resources (a "workbench"; see Figure 1). Each sample point consists of G's application, resource, and data profile values for that run. NIMO's modeling engine actively initiates new runs of G on selected VMs in the workbench. The choice of VM slivers is guided by the active learning

algorithms. The goal is to obtain sufficient samples to learn an accurate performance model for G quickly; see Figure 2. Further details on instrumentation and the active learning in NIMO are in [4, 5].

Shirako provides programmatic leasing primitives to configure VMs on-the-fly to enable NIMO's active learning. For each application run on a VM, NIMO collects the instrumentation data during the run, then aggregates it to generate a sample data point as soon as the run completes. We continuously track the samples obtained for each application in a database of samples (Figure 1). We will use this feature to demonstrate the active learning of models in real-time.

4.3 Model-guided VM Provisioning

A key challenge in provisioning VMs for meeting performance goals is to determine the measure of physical resources or the "sliver size" bound to each VM. Once NIMO learns the application performance models, it can use them to determine the VM sliver sizes for applications by: (i) ranking the list of available VM slivers in order of application performance; (ii) predicting the VM slivers that meet an application's target performance; and (iii) doing what-if analysis (Section 5.3). Details of model use in NIMO are in [6].

5. SAMPLE DEMONSTRATION SESSION

Overall, we will demonstrate: (i) how NIMO builds models using active and accelerated learning; (ii) how NIMO guides the provisioning of VMs in Shirako in real-time to meet system performance goals.

5.1 Demonstration Setup

To demonstrate the NIMO system we need a utility resource infrastructure. Shirako provides mechanisms to create and manage *virtual data centers* from a utility consisting of a network of servers. We plan to create a network of servers using a small cluster of laptops. We will have applications drawn from computational science (CardioWave, NAMD, BLAST, and fMRI [6]), and RUBiS, an opensource multi-tier web service that implements the core functionality of an auction site like eBay: selling, browsing, bidding.

NIMO will use Shirako's programmatic leasing primitives to deploy and run the scientific workflows and RUBiS on the utility resources. Each run will involve NIMO requesting a VM from Shirako, and then dynamically instantiating and running the application on the VM. NIMO will build the model for each application, and use the learned model to meet system performance goals.

5.2 Model Learning

We will demonstrate the model learning process in real-time (as shown in Figure 2) using a database of samples as it is being populated by NIMO (see Figure 1). We keep track of all sample runs in the sample database to show: (i) online sample collection using active learning; (ii) model building using available samples; (iii) the current model's accuracy; (iv) selection of new samples based on the current model's accuracy; (iv) updating of the model and convergence of its accuracy as the system collects more samples.

5.3 Model-guided Resource Planning

We will demonstrate how a self-managing system or a system administrator can query the models to meet system performance goals as follows:

5.3.1 Ranking

An administrator can query the model to rank a list of available VM slivers in order of application performance. We will show how to query the model to determine the execution time of scientific applications, and the response time of the database and other tiers of RUBiS on available VM slivers.

5.3.2 Guaranteeing SLAs

We will demonstrate how an administrator can determine the VM sliver size for an application that meets an SLA specified along with the application. An example SLA may say: "the application must run in less than 10 minutes", or "the response time from the database tier should be less than 2 ms". Given a bound on the application performance, the administrator can query the performance model to determine the VM sliver that meets the SLA target. We will also show how well the NIMO-guided choice of VM meets the specified SLA.

5.3.3 What-if Analysis

We will show the querying of models with *what-if* queries, e.g., "how will the performance of RUBiS change if the CPU cycles or memory bound to the VM is doubled for the database tier?" Accurate and timely answers to such questions are useful for capacity planning as well as to estimate the potential impact of system failures and performance problems. We will demonstrate how NIMO answers such queries, and compare the answers that NIMO gives to the actual values by running the application on a VM.

6. SUMMARY

We demonstrate the NIMO system that uses active and accelerated learning techniques to learn performance models automatically for database applications. The learned models are used to provision VMs for these applications in a utility setting, including (i) ranking available VM sizes based on application performance, (ii) determining VM sizes that meet a performance target, and (iii) answering what-if queries. We use computational science applications as well as multi-tier web services in the demonstration.

7. REFERENCES

[1] D. Irwin, J. S. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum. Sharing Networked Resources with Brokered Leases. In *Proceedings of the USENIX Annual Technical Conference*, Jun 2006.

[2] *Network/Internet Computing Lab.* `http://www.cs.duke.edu/nicl`.

[3] L. Ramakrishnan, L. Grit, A. Iamnitchi, D. Irwin, A. Yumerefendi, and J. Chase. Toward a Doctrine of Containment: Grid Hosting with Adaptive Resource Control. In *Proceedings of ACM Conference on Supercomputing (SC)*, Nov 2006.

[4] P. Shivam, S. Babu, and J. Chase. Active and Accelerated Learning of Cost Models for Optimizing Scientific Applications. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, Sep 2006.

[5] P. Shivam, S. Babu, and J. Chase. Active Sampling for Accelerated Learning of Performance Models. In *Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, Jun 2006.

[6] P. Shivam, S. Babu, and J. Chase. Learning Application Models for Utility Resource Planning. In *Proceedings of International Conference on Autonomic Computing (ICAC)*, Jun 2006.