Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Sandpiper: Black-box and gray-box resource management for virtual machines [☆]

Timothy Wood [a,*], Prashant Shenoy [a], Arun Venkataramani [a], Mazin Yousif [b]

[a] University of Massachusetts, Dept. of Computer Science, 140 Governor's Drive, Amherst, MA 01003, United States
[b] Avirtec, 1236 E. Grant Road, Tucson, AZ 85719, United States

A R T I C L E I N F O

Article history:
Available online 8 July 2009

Keywords:
Virtualization
Data centers
Migration
Dynamic provisioning

A B S T R A C T

Virtualization can provide significant benefits in data centers by enabling dynamic virtual machine resizing and migration to eliminate hotspots. We present Sandpiper, a system that automates the task of monitoring and detecting hotspots, determining a new mapping of physical to virtual resources, resizing virtual machines to their new allocations, and initiating any necessary migrations. Sandpiper implements a black-box approach that is fully OS- and application-agnostic and a gray-box approach that exploits OS- and application-level statistics. We implement our techniques in Xen and conduct a detailed evaluation using a mix of CPU, network and memory-intensive applications. Our results show that Sandpiper is able to resolve single server hotspots within 20 s and scales well to larger, data center environments. We also show that the gray-box approach can help Sandpiper make more informed decisions, particularly in response to memory pressure.

1. Introduction

Data centers—server farms that run networked applications—have become popular in a variety of domains such as web hosting, enterprise systems, and e-commerce sites. Server resources in a data center are multiplexed across multiple applications—each server runs one or more applications and application components may be distributed across multiple servers. Further, each application sees dynamic workload fluctuations caused by incremental growth, time-of-day effects, and flash crowds [1]. Since applications need to operate above a certain performance level specified in terms of a *service level agreement (SLA)*, effective management of data center resources while meeting SLAs is a complex task.

One possible approach for reducing management complexity is to employ *virtualization*. In this approach, applications run on virtual servers that are constructed using virtual machines, and one or more virtual servers are mapped onto each physical server in the system. Virtualization of data center resources provides numerous benefits. It enables application isolation since malicious or greedy applications can not impact other applications co-located on the same physical server. It enables server consolidation and provides better multiplexing of data center resources across applications. Perhaps the biggest advantage of employing virtualization is the ability to flexibly remap physical resources to virtual servers in order to handle workload dynamics. A workload increase can be handled by increasing the resources allocated to a virtual server if idle resources are available on the physical server, or by simply migrating the virtual server to a less loaded physical server. Migration is transparent to the applications and modern virtualization platforms support this capability [6,16]. How-

☆ A preliminary version of this paper, *Black-box and Gray-box Strategies for Virtual Machine Migration*, appeared in NSDI 2007.
* Corresponding author. Tel.: +1 4135454753.
E-mail addresses: twood@cs.umass.edu (T. Wood), shenoy@cs.umass.edu (P. Shenoy), arun@cs.umass.edu (A. Venkataramani), mazin@avirtec.net (M. Yousif).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.04.014

ever, detecting workload hotspots and initiating a migration is currently handled manually. Manually-initiated migration lacks the agility to respond to sudden workload changes; it is also error-prone since each reshuffle might require migrations or swaps of multiple virtual servers to rebalance system load. Migration is further complicated by the need to consider multiple resources—CPU, network, and memory—for each application and physical server.

To address this challenge, this paper studies automated black-box and gray-box strategies for virtual machine provisioning in large data centers. Our techniques automate the tasks of monitoring system resource usage, hotspot detection, allocating resources and initiating any necessary migrations. More importantly, our black-box techniques can make these decisions by simply observing each virtual machine from the outside and without any knowledge of the application resident within each VM. We also present a gray-box approach that assumes access to a small amount of OS-level statistics in addition to external observations to better inform the provisioning algorithm. Since a black-box approach is more general by virtue of being OS and application-agnostic, an important aspect of our research is to understand if a black-box approach alone is sufficient and effective for hotspot detection and mitigation. We have designed and implemented the Sandpiper system to support either black-box, gray-box, or combined techniques. We seek to identify specific limitations of the black-box approach and understand how a gray-box approach can address them.

Sandpiper implements a hotspot detection algorithm that determines *when to resize or migrate* virtual machines, and a hotspot mitigation algorithm that determines *what and where to migrate* and *how many resources to allocate*. The hotspot detection component employs a monitoring and profiling engine that gathers usage statistics on various virtual and physical servers and constructs profiles of resource usage. These profiles are used in conjunction with prediction techniques to detect hotspots in the system. Upon detection, Sandpiper grants additional resources to the overloaded servers if available. If necessary, Sandpiper's migration manager is invoked for further hotspot mitigation. The migration manager employs provisioning techniques to determine the resource needs of overloaded VMs and uses a greedy algorithm to determine a sequence of moves or swaps to migrate overloaded VMs to underloaded servers.

We have implemented our techniques using the Xen platform [3]. We conduct a detailed experimental evaluation on a testbed of two dozen servers using a mix of CPU-, network- and memory-intensive applications. Our results show that Sandpiper can alleviate single server hotspots in less than 20 s and more complex multi-server hotspots in a few minutes. Our results show that Sandpiper imposes negligible overheads and that gray-box statistics enable Sandpiper to make better migration decisions when alleviating memory hotspots.

The rest of this paper is structured as follows. Section 2 presents some background and Sections 3–6 present our design of Sandpiper. Section 7 presents our implementation and evaluation. Finally, Sections 8 and 9 present related work and our conclusions, respectively.

2. Background and system overview

Historically, approaches to dynamic provisioning have either focused on dynamic *replication*, where the number of servers allocated to an application is varied, or dynamic *slicing*, where the fraction of a server allocated to an application is varied. With the re-emergence of server virtualization, application *migration* has become an option for dynamic provisioning. Since migration is transparent to applications executing within virtual machines, our work considers this third approach—resource provisioning via dynamic migrations in virtualized data centers. We present *Sandpiper*,[1] a system for automated resource allocation and migration of virtual servers in a data center to meet application SLAs. Sandpiper assumes a large cluster of possibly heterogeneous servers. The hardware configuration of each server—its CPU, network interface, disk and memory characteristics—is assumed to be known to Sandpiper. Each physical server (also referred to as a physical machine or PM) runs a *virtual machine monitor* and one or more virtual machines. Each virtual server runs an application or an application component (the terms virtual servers and virtual machine are used interchangeably). Sandpiper currently uses Xen to implement such an architecture. Each virtual server is assumed to be allocated a certain slice of the physical server resources. In the case of CPU, this is achieved by assigning a subset of the host's CPUs to each virtual machine, along with a weight that the underlying Xen CPU scheduler uses to allocate CPU bandwidth. In case of the network interface, Xen is yet to implement a similar fair-share scheduler; a best-effort FIFO scheduler is currently used and Sandpiper is designed to work with this constraint. In case of memory, a slice is assigned by allocating a certain amount of RAM to each resident VM. All storage is assumed to be on a network file system or a storage area network, thereby eliminating the need to move disk state during VM migrations [6].

Sandpiper runs a component called the *nucleus* on each physical server; the nucleus runs inside a special virtual server (domain-0 in Xen) and is responsible for gathering resource usage statistics on that server (see Fig. 1). It employs a *monitoring engine* that gathers processor, network interface and memory swap statistics for each virtual server. For gray-box approaches, it implements a daemon within each virtual server to gather OS-level statistics and perhaps application logs.

The nuclei periodically relay these statistics to the Sandpiper *control plane*. The control plane runs on a distinguished node and implements much of the intelligence in Sandpiper. It comprises three components: a *profiling engine*, a *hotspot detector* and a *migration and resizing manager* (see Fig. 1). The profiling engine uses the statistics from the nuclei to construct resource usage profiles for each virtual server and aggregate profiles for each physical server. The hotspot detector continuously monitors these usage profiles to detect hotspots—informally, a hotspot is said to have occurred if the aggregate usage of any resource (processor, network or memory) exceeds a threshold or if SLA violations occur for a "sustained" period. Thus, the hotspot

[1] A migratory bird.