Technical journals online.com

Research Paper

A SURVEY OF EXISTING RESOURCE MANAGEMENT TECHNIQUES IN VIRTUAL ENVIRONMENTS

Dr. N. K. Cauvery[1] , Ms. Sandhya S[2] , Satvik N[3] , and Vaishakh B N[4]

Address for Correspondence

[1]Professor, [2]Assistant Professor, [3,4]7th semester BE, Department of Computer Science,
RV College of Engineering. Bangalore - 560 059

ABSTRACT

Virtualization is the buzz word in today´s IT. Virtualization provides high availability for critical applications, and streamlines application deployment and migrations. Virtualization has become an essential technology in the data center. Virtualization improves resource utilization through server consolidation, but it also makes resource management more complex. Virtualization can simplify IT operations and allow IT organizations to respond faster to changing business demands. This paper discusses the different resource management techniques recently developed and researched by researchers that are applied on Virtual machines to achieve maximum throughput, minimize response time, and avoid overload.

KEYWORDS resource management, virtualisation.

I. INTRODUCTION

Virtualization is the abstraction of IT resources that masks the physical nature and boundaries of those resources from resource users[1]. An IT resource can be a server, a client, storage, networks, applications or OSs. Essentially, any IT building block can potentially be abstracted from resource users. At one time, virtualization was considered just an academic interest. But now, it is used widely in the industry to provide cost effective and flexible applications. Allocation of resources in a virtual environment holds the key in the performance of virtual machines. The main resources of a virtual machine include CPU, memory, disk, and network resources. Each virtual machine consumes a portion of the CPU, memory, network bandwidth, and storage resources of the physical machine it belongs to. The host guarantees each virtual machine, its share of the underlying hardware resources based on a number of factors.

- Available resources for the physical machine
- Reservation, limit, or shares of the virtual machine.
- Number of virtual machines powered on, and resource utilization by those virtual machines.
- Overhead required to manage the virtualization.

Effective management of virtualized resources is a challenging task for providers, as it often involves selecting the best resource allocation out of a large number of alternatives. In the following sections we discuss various resource management techniques in virtual environment.

II. RESOURE MANAGEMENT TECHNIQUES

In this section, we talk about three important resource management techniques employed in virtual environment. A. Replication and Migration as Resource Management Mechanisms Virtualization refers to an abstract layer between the operating system and the hardware. The layer provides an interface to the actual hardware that allows for the support of a number of virtual machines. A Virtual machine running a server program is called a virtual server. Due to virtualization, the applications get smaller portion of the physical hardware. Thus virtualization benefits the data centers by allowing several applications to make use of a physical machine. This suggests that resource allocation should be dynamic to support the allocation of resources on- demand.

Dynamic resource management requires monitoring mechanisms and dynamic resource reallocation mechanisms. Examples of this mechanisms are migration and replication which are included under virtual server relocation. Migration consists of transferring a running virtual machine from one physical machine to another. Replication entails the creation of a replica of a virtual machine on another physical machine. Requests for the virtual server are balanced between the two instances. This should reduce the computing resources needed by a single physical machine by distributing requests to two different virtual machines on two different physical machines. This technique is useful when the combined resource needs of the virtual machines hosted in a physical machine exceed the resource availability. In this paper the authors use OpenVZ which provides operating system-level virtualization. OpenVZ is a Linux kernel modified to run multiple, isolated containers (i.e., virtual user-space environments) on a single physical machine or hardware node. OpenVZ supports the execution of multiple containers. The containers are isolated program execution environments, which appear as stand-alone servers to users. The host system runs inside a privileged container.

System Design

Golondrina was conceived as a multi-resource management system for data centers. Golondrina consists of three primary management entities: Client, Server, Gate.

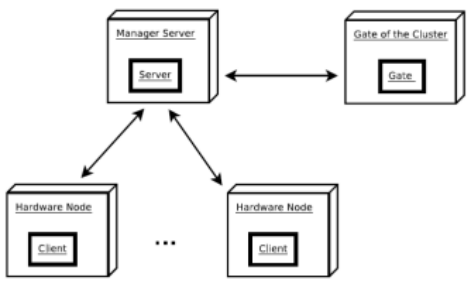

Fig. 1. Golondrina's system architecture

1) **Client -** Each hardware node has a Client instance that runs in the privileged container. The Client requires access to the operating systems configuration and accounting files, and the OpenVZ

management tools. The Client instance provides the following functionality:

a) The periodic collection of CPU utilization statistics from the containers and the hardware node. This is done by reading the hardware nodes operating systems accounting files through the proc file system. These statistics are sent to the Server.

b) Support for migration and replication of the containers. This is done upon a request from the Server.

2) **Server** Upon receiving the CPU utilization statistics sent by the Client, the Server stores the statistics. The data model used assumes that the hardware node is an aggregation of containers. The attributes of the hardware node and containers represent CPU utilization metrics.

a) **Analyzing CPU Utilization Statistics** The CPU utilization statistics sent by the client are used to create a CPU utilization profile of resource usage over time. A mathematical model uses the CPU utilization statistics collected at time t i for predicting the CPU utilization of a container or hardware node at time t i+1. This mathematical model relies on the last observation in the sequence of observations and two parameters: μ, the mean of the values in the sequence, and θ, which accounts for the variations of the values in the sequence.

Given a sliding window of CPU utilization statistics W = [u x , ..., u t] with maximum size w and where x = max(0, t-w + 1). The parameters μ and θ at time t are calculated as follows:

$$\mu_t = \frac{\sum_{i=x}^{t} u_i}{t - x + 1}$$

$$\theta_t = \frac{\sum_{i=x}^{t-1}(u_i - \mu_t) * (u_{i+1} - \mu_t)}{max(1, \sum_{i=x}^{t-1}(u_i - \mu_t)^2)}$$

Having the values $u_t, \mu_t and \theta_t$, it is possible to predict the CPU utilization at time t + 1:

$$u_{t+1} = \mu_t + \theta_t * (u_t - \mu_t)$$

The profiling process uses a historicalpolicy to calculate the containers profiled CPU utilization.

b) **Resource Stress Check** The Server executes a resource stress check on every hardware node that is not currently involved in a relocation. The resource stress check consists of two steps. First, it verifies whether the predicted CPU utilization of the hardware node exceeds the CPU utilization threshold. Then, if the latter is true, it checks whether k out of the previous n resource stress checks also exceeded the threshold, in which case the hardware node is considered to be under a resource stress situation.

$$(u_{t+1} > threshold) \wedge (\sum_{i=1}^{n}(u_i > threshold) \geq k)$$

c) **Reloation Search:** After the resource stress check round is completed the hardware nodes are classified as stressed or non-stressed hardware nodes. If both sets are non-empty, then the Server searches for a sequence of relocations to solve the resource stress situations. This decision making process consists of determining which containers hosted in stressed hardware nodes will be relocated and which non-stressed hardware nodes will serve as a target for those relocations.

3) **Gate** The Gate component runs in a non-virtualized physical server, which is used as the gate of the cluster (i.e., all service requests come through this physical server). Its responsibility is to update the load balancers configuration after a replication occurs.

B. Autonomic resource management using fuzzy logic-based approaches

In this paper the author proposes a two-level resource management system to dynamically allocate resources to individual virtual containers. It uses local controllers at the virtual container level and a global controller at the resource-pool level. An important advantage of this two-level control architecture is that it allows independent controller designs for separately optimizing the performance of applications and the use of resources. Autonomic resource allocation is realized through the interaction of the local and global controllers. Local controller uses fuzzy logic-based approaches to efficiently and robustly deal with the complexity and uncertainties of dynamically changing workloads and resource usage. The global controller determines the resource allocation based on a proposed profit model, with the goal of maximizing the total profit of the data center.

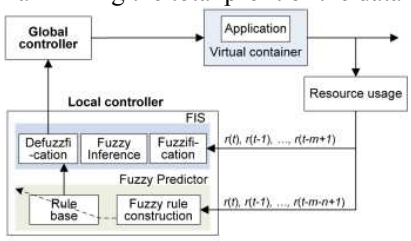

Fig. 2. Resource management based on fuzzy logic systems.

This resource management system can significantly reduce resource consumption while still achieving application performance targets. Resource management in a data center is decoupled at two levels: virtual containers and resource pools. The key to cost effective resource allocation is the ability to efficiently find the minimum amount of resources that an application needs to meet the desired QoS.

In each virtual container hosting an application, a local controller is responsible for determining the amount of resources needed by the application and making resource requests accordingly. A global controller responds to the local controllers requests by dynamically allocating resources across multiple virtual containers hosted on the same physical resources. It controls the resource allocations in a way that maximizes the data centers profit.

Two fuzzy-logic-based approaches- fuzzy modeling and fuzzy prediction are proposed for use by local controllers in automatically learning of runtime behavior of virtual containers under dynamically changing workloads. Two-level resource control system is preferred over the more obvious centralized approach in which all the control functions are implemented at one centralized location. Since local containers are independent of

each other, heterogeneous local controllers implementations are possible The system handles two different types of optimizations independently. Interaction between the local and global controllers enables a virtual container to augment its resources in response to increased workload, and to reduce its resources when they are no longer needed. The main task of the local controller is to estimate the set of resources needed by an application running in the container. Each individual local controller tries to minimize the resource cost by only requesting the resources necessary for meeting the application SLA.

The global controller receives requests for physical resources from the local controllers and allocates the resources among them as required. It seeks to make allocations that maximize the data centers profit, which is the revenue received by allocating the physical resources among virtual containers minus the penalties due to violations of resource SLA.

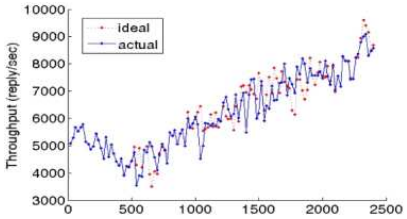

Fig. 3. Experimental Results

A real world experiment was conducted using the Fuzzy Approach. The results were promising. Figure 3 shows that the applications throughput achieved by using the local controller is close to the ideal throughput obtainable for the workload (the difference is within 5%), indicating the effectiveness of the fuzzy-modeling approach under real-world workloads

Advantages of Fuzzy Based Method: One of the advantages of fuzzy approaches is that they do not require prior knowledge or a mathematical model of the system being managed. They typically do not need time-consuming training, which makes them suitable for real-time control. Moreover, the approaches are robust with respect to noisy data and have the ability to adapt to changes very quickly

C. Sandpiper-Black Box and Gray box resource management technique

This paper studies automated black-box and gray-box strategies for virtual machine provisioning in large data centers. These techniques automate the tasks of monitoring system resource usage, hotspot detection, allocating resources and initiating any necessary migrations. More importantly, the black-box techniques can make decisions by simply observing each virtual machine from the outside and without any knowledge of the application resident within each VM. The authors 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. The authors have designed and implemented the Sandpiper system to support either black-box, gray-box, or combined techniques.

They have identified specific limitations of the black-box approach and understood 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, Sandpipers 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.

Sandpiper supports both black- and gray-box monitoring techniques that are combined with profile generation tools to detect hotspots and predict VM resource requirements.

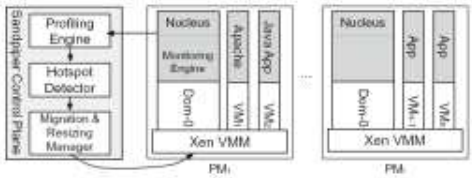

Fig. 4. The Sandpiper architecture

1) Unobtrusive black-box monitoring The monitoring engine is responsible for tracking the processor, network and memory usage of each virtual server. It also tracks the total resource usage on each physical server by aggregating the usages of resident VMs. In a pure black-box approach, all usages must be inferred solely from external observations and without relying on OS-level support inside the VM
2) CPU Monitoring By instrumenting a hypervisor, its possible to gain access to CPU scheduling events which indicate when a VM is scheduled and when it relinquishes the CPU. These events are tracked to determine the duration for which each virtual machine is scheduled within each measurement interval.
3) Network Monitoring The monitoring engine can conveniently monitor each VMs network usage. The monitoring engines can use the Linux / proc interface to monitor the number of bytes sent and received on each interface. These statistics are gathered over interval I and returne to the control plane.
4) Memory Monitoring Black-box monitoring of memory is challenging since Xen allocates a user-specified amount of memory to each VM and requires the OS within the VM to manage that memory; as a result, the memory utilization is only known to the OS within each VM. It is possible to instrument Xen to observe memory accesses within each VM through the use of shadow page tables, which is used by Xens migration mechanism to determine which pages are dirtied during migration.

However, trapping each memory access results in a significant application slowdown and is only enabled during migrations[6]. Thus, memory usage statistics are not directly available and must be inferred.

Gray-box monitoring

Sandpiper supports gray-box monitoring, when feasible, using a light-weight monitoring daemon that is installed inside each virtual server. In Linux, the monitoring daemon uses the /proc interface to gather OS-level statistics of CPU, network, and memory usage. The memory usage monitoring, in particular, enables proactive detection and mitigation of memory hotspots. The monitoring daemon also can process logs of applications such as web and database servers to derive statistics such as request rate, request drops and service times. Direct monitoring of such application-level statistics enables explicit detection of SLA violations, in contrast to the blackbox approach that uses resource utilizations as a proxy metric for SLA monitoring.

Profile generation

The profiling engine receives periodic reports of resource usage from each nucleus. It maintains a usage history for each server, which is then used to compute a profile for each virtual and physical server. A profile is a compact description of that servers resource usage over a sliding time window W. Three black-box profiles are maintained per virtual server: CPU utilization, network bandwidth utilization, and swap rate (i.e., page fault rate). If gray-box monitoring is permitted, four additional profiles are maintained: memory utilization, service time, request drop rate and incoming request rate. Similar profiles are also maintained for each physical server, which indicate the aggregate usage of resident VMs.

Hotspot detection

The hotspot detection algorithm is responsible for signaling a need for VM resizing whenever SLA violations are detected implicitly by the black-box approach or explicitly by the gray-box approach. Hotspot detection is performed on a perphysical server basis in the black-box approach.

Resource provisioning

A hotspot indicates a resource deficit on the underlying physical server to service the collective workloads of resident VMs. Before the hotspot can be resolved, Sandpiper must first estimate how much additional resources are needed by the overloaded VMs to fulfill their SLAs.

Hotspot mitigation

Once a hotspot has been detected, Sandpiper must determine if the hotspots can be resolved with local resource adjustments, or if migrations are required to balance load between hosts

III. CONCLUSION

The study in [2] proposes a resource management system for operating system level virtualized environments. In addition,this is the first c study that uses replication as an alternative to migration and compares both mechanisms. There are many ways in which the Golondrina could be extended. One of them is adding memory and network as managed resources. The study in [3] proposes the use of a two-level resource management system to dynamically allocate resources to individual virtual containers. An important advantage of this two-level control architecture is that it allows independent controller designs for separately optimizing the performance of applications and the use of resources. Autonomic resource allocation is realized through the interaction of the local and global controllers. The paper [4] argued that virtualization provides significant benefits in data centers by enabling virtual machine migration to eliminate hot spots. The authors introduced Sandpiper, system that automates the task of monitoring and detecting hotspots, determining a new mapping of physical to virtual resources, and resizing or migrating VMs to eliminate the hotspots. They discussed a black-box strategy that is fully OS- and application-agnostic as well as a graybox approach that can exploit OS- and application-level statistics. An evaluation of their Xen-based prototype showed that VM migration is a viable technique for rapid hotspot elimination in data center environments. Using solely black-box methods, Sandpiper was capable of eliminating simultaneous hotspots involving multiple resources. They found that utilizing gray-box information can improve the responsiveness of our system, particularly by allowing for proactive memory allocations and better inferences about resource requirements.

REFERENCES

1. Gartner Definition Of Virtualization Online article available at http://www.gartner.com/it-glossary/virtualization/
2. Lutfiyya; Hanan, Keller; Gaston. Replication and Migration as Resource Management Mechanisms for Virtualized Environments. 2010, Sixth Internation Conference on Autonomic and Autonomous Systems.
3. Jing Xu, Ming Zhao,Jos Fortes, Robert Carpenter, Mazin Yousif. Autonomic resource management in virtualized data centers using fuzzy logic-based approaches Cluster Comput (2008) 11: 213227 DOI 10.1007/s10586-008-0060-0
4. T. Wood et al., Sandpiper. Black-box and gray-box resource management for virtual machines, Computer Network (2009), doi:10.1016/j.comnet.2009.04.014