ISO 9001 - 2015

ISSN 2349 - 4891

Monthly

IJRRAS

IF
4.665

Volume 4,Issue 5, May 2017

International Journal of
Recent Research and Applied Studies

SURRAGH PUBLICATIONS

ISSN: 2349 - 4891

International
Journal of Recent Research and Applied Studies
(Multidisciplinary Open Access Refereed e-Journal)

A Survey on Virtualization: Integration and Load Balancing in Data Centers

Usha Bellad[1] & Jalaja. G[2]

[1]*Student, M.Tech, CSE, B N M Institute of Technology, Bengaluru, Karnataka, India.*
[2]*Associate Professor, CSE, B N M Institute of Technology, Bengaluru, Karnataka, India.*

Received 4tht April 2017, Accepted 3rd May 2017

Abstract

Data centers became an optimal solution for business customers to maintain and promote their business needs to clients via Internet. Now days the virtual computing allows the business costumer to scale up and down their resource usage based on needs. In order to achieve resource multiplexing in virtual computing, recent researches were introduced dynamic resource allocation through virtual machines. Existing dynamic approaches followed unevenness procedures to allocate the available resources based on current workload of systems. Unexpected demand for huge amount of resources in future may cause allocation failure or system hang problem. In this paper we present a Survey on Virtualization: Integration and Load Balancing in Data Centers new systematic approach to predict the future resource demands of VMM from past usage. This approach uses the resource prediction algorithm to estimate future needs to avoid allocation failure problem in virtual machine management.

Keywords: Survey, Virtualization, Integration, Load Balancing, Data Centers.

Introduction

Virtual machine (VM) states that activities, availability and performance issues place great demand on a storage infrastructure. So administrators need to understand the most critical data center storage virtualization considerations are

Storage performance

Today, administrators can select from many different shared storage platforms, such as iSCSI, Fibre Channel, network-attached storage and Fibre Channel over Ethernet. Each of these platforms works fine in a virtual environment but offers radically different scalability, performance, availability and capacity characteristics [1]. Storage performance is usually considered among equals because multiple VMs residing on a physical host have considerable storage bandwidth requirements. A virtualized server hosting 10 VMs, for example, needs to load all 10 VMs from storage, to periodically snapshot the current state of each VM and to provide the VM data to users [1].

Integration with backup and disaster plans

Storage systems are backed up and protected with some manner of disaster recovery (DR) planning. Any new storage implementation should be fully compatible with existing backup and DR software, such as local snapshot and remote replication tools. If not, the new storage setup may force changes (and possibly introduce errors) to the existing data protection scheme or add additional tools that unnecessarily complicate data protection. Lab testing can usually confirm a storage system's compatibility [1].

Correspondence
Usha Bellad
E-mail: bkh.usha@gmail.com

Ensure redundant storage access:

Storage disruptions can have devastating consequences on a virtualized data center. When a traditional server is disrupted, usually one application is affected. But when a server with 10 or 20 virtualized workloads is disrupted, it affects far more business applications and users [1].

Energy-efficient storage:

The energy needed to run larger-capacity and higher-performance storage systems means a larger total cost of ownership. Consider more energy-efficient storage systems, which provide more input/output operations per second and bandwidth per watt of energy for active data. Energy-efficient storage should also provide more capacity per watt for inactive (e.g., archived) data. Achieving more energy-efficient storage is usually accomplished through a combination of controller designs and disk capacity/performance tradeoffs [1].

Virtualization and server management

A perpetual challenge with virtualization is the abstraction layer that separates a logical workload from its underlying hardware. It’s almost impossible to tell

which physical server is running each virtual workload, which makes it far more difficult to intuitively optimize and troubleshoot the virtual environment [1].

Related Work

Computing in VM is an emerging computing technology that is rapidly consolidating itself as the next big step in the development and deployment of an increasing number of distributed applications [1][2]. Cloud computing nowadays becomes quite popular among a community of cloud users by offering a variety of resources. Cloud computing platforms, such as those provided by Microsoft, Amazon, Google, IBM, and Hewlett-Packard, let developers deploy applications across computers hosted by a central organization. These applications can access a large network of computing resources that are deployed and managed by a cloud computing provider [1].

In cloud platforms, resource allocation (or load balancing) takes place at two levels. First, when an application is uploaded to the cloud, the load balancer assigns the requested instances to physical computers, attempting to balance the computational load of multiple applications across physical computers. Second, when an application receives multiple incoming requests, these requests should be each assigned to a specific application instance to balance the computational load across a set of instances of the same application. For example, Amazon EC2[5] uses elastic load balancing (ELB) to control how incoming requests are handled. Application designers can direct requests to instances in specific availability zones, to specific instances, or to instances demonstrating the shortest response times.

Elnozahy et al. [11] have investigated the problem of power efficient resource management in a single web-application environment with fixed SLAs (response time) and load balancing handled by the application. As in [3], two power saving techniques are applied: switching power of computing nodes on/off and Dynamic Voltage and Frequency Scaling (DVFS). The main idea of the policy is to estimate the total CPU frequency required to provide the necessary response time, determine the optimal number A. However, the transition time for switching the power of a node is not considered. Only a single application is assumed to be run in the system and, like in [10], the load balancing is supposed to be handled by an external system. The algorithm is centralized that creates an SPF and reduces the scalability. Despite the variable nature of the workload, unlike [11], the resource usage data are not approximated, which results in potentially inefficient decisions due to fluctuations. Nathuji and Schwan [4] have studied power management techniques in the context of virtualized data centers, which has not been done before.

Besides hardware scaling and VMs consolidation, the authors have introduced and applied a new power management technique called “soft resource scaling”. The idea is to emulate hardware scaling by providing less resource time for a VM using the Virtual Machine Monitor’s (VMM) scheduling capability. The authors found that a combination of “hard” and “soft” scaling may provide higher power savings due to the limited number of hardware scaling states. The authors have proposed an architecture where the resource management is divided into local and global policies. At the local level the system leverages the guest OS‟s power management strategies. However, such management may appear to be inefficient, as the guest OS may be legacy or power unaware.

The provision of resource may be made with various virtualization techniques. This may ensure a higher throughput and usage than the existing cloud resource services. The future work is required to deals with the evolutionary techniques that will further result in better resource allocation, leading to improve resource utilization. These resource allocation strategies have the following limitations.

a) Since users rent resources from remote servers for their purpose, they don’t have control over their resources.

b) Migration problem occurs, when the users wants to switch to some other provider for the better storage of their data. It’s not easy to transfer huge data from one provider to the other.

c) More and deeper knowledge is required for allocating and managing resources in cloud, since all knowledge about the working of the cloud mainly depends upon the cloud service provider.

Hence the existing systems are has the limitations as migration of resources, overloading at server and migrates only working set of an idle VM. To overcome from these limitations this paper presents skewness algorithm which uses green computing technologies and load prediction algorithm which uses past resource usage to predict the resources for present working environment.

1. System Design

A. System architecture

The proposed system presents the design and implementation of an automated resource management system that achieves a good balance. The proposed system makes the following three contributions:

a)Develops a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used.

b)Introduces the concept of “skewness” to measure the uneven utilization of a server. By minimizing skewness, thus we can improve the overall utilization of servers in the face of multi-dimensional resource constraints.

c)Designs a load prediction algorithm that can capture the future resource usages of applications accurately without looking inside the VMs. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement churn significantly.

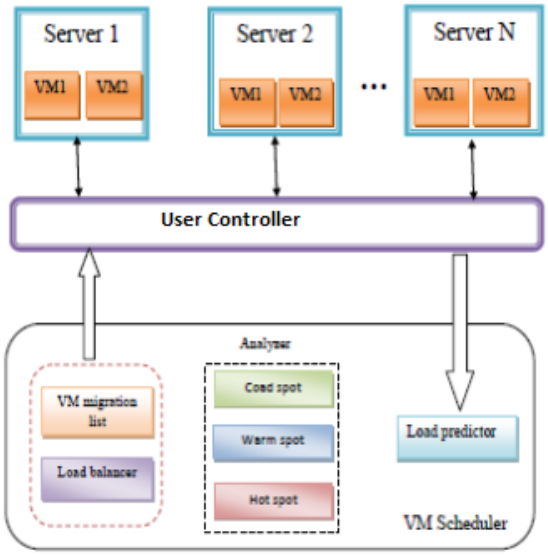

Figure I
System architecture

The figure I represents the architecture of the dynamic resource allocation for cloud computing environment, which consists of N servers each server consists of two virtual machines(VM) those are connected to the VM scheduler is connected to the internet to distribute the resources dynamically to the clients ,the clients are accessing resources through the internet. Virtual machine (VM) is a software implementation of computing environment in which operating system or program can be installed and run. The VM Scheduler is invoked periodically and receives the resource demand history of VMs, the capacity and the load history of server, and the current layout of VMs on servers.

B. **Sync Inventory Algorithm**

It is possible to design the sync inventory algorithm so that it can migrate away multiple VMs during each run. But this can add more load on the related servers during a period when they are already overloaded. It is decided to use this more conservative approach and leave the system some time to react before initiating additional migrations. In the first scenario, the VMs running in identified hot spots are migrated to warm spot servers which will not become hot by accommodating the VMs. In the second scenario, if sufficient warm spots are not available to accommodate the VMs in the hot spot, few loads are migrated to the nodes in the cold spot also to mitigate the hot spots.

Input: source, destination, flow, nodes_list, conn_matrix, costs_matrix, capacity_matrix
Output: path
T = all the nodes ∈ nodes_list
S = ∅
for i=0 to size of T **do**
  node = T.get(i)
  if node == source **then**
    f.put(source, 0)
    prec.put(source, 0);
  else
    if source and node are connected and there is enough capacity between them to accommodate flow **then**
      f.put(node, costs[source][node])
      pred.put(node,source)
    else
      f.put(node, ∞);
    end if
  end if
end for
for $i \in T$ **do**
  Find j in T such as $f(j) = min\ f(i)$ and $i \in T$
  Remove j from T
  Add j in S
  for k=0 to size of T **do**
    current = T.get(k)
    if j and current are connected AND the capacity between them is enough to accommodate flow AND $f.get(curr) \geq f.get(j) + costs[j][current]$ **then**
      f.put(current, f.get(j)+costs[j][current]);
      pred.put(current, j);
    end if
  end for
end for
path = Reconstruct_path(source, destination, prec)

Figure II
Sync Inventory Algorithm

C. Mapping Algorithm

```
Data: Set of VMs, requested_vm
Result: Mapping for the requested VM
mapping ← None
for vm in vms_in_node do
    if vm.score == -1 then
        simulate_migration(vm)
        vms_in_node.pop()
    end
end
mapping = map(requested_lease)
if mapping == None and vms_in_node remaining then
    optimal_vm = compute_optimal_vm_to_migrate()
    simulate_migration(optimal_vm)
    vms_in_node.pop()
end
mapping = map(requested_lease)
while mapping == None and vms_in_node remaining do
    for vm in vms_in_node do
        simulate_migration(vm)
        vms_in_node.pop()
        mapping = map(requested_lease)
        if mapping != None then
            break
        end
    end
end
```

Figure III
Mapping Algorithm

The paper introduces the concept of mapping algorithm in figure III to quantify the unevenness in the utilization of multiple resources on a server. By minimizing the sync inventory, we can combine different types of workloads nicely and improve the overall utilization of server resources.

D. Load Balancing Algorithm

Load balancing ensures that data stores in a data store cluster do not exceed their configured thresholds as space consumption and IO loads change during runtime. Unlike DRS, which minimize the resource usage deviation across hosts in a cluster, Storage DRS is driven by threshold trigger. Its load balancing mechanism moves VMs out of only those data stores, which exceed their configured threshold values. Figure IV gives the outline of load balancing as used by Storage DRS. For each pass of Storage DRS, the algorithm is invoked first for data stores that exceeded their space threshold and later for those violating IO threshold, effectively fixing space violations followed by IO violations in the data store cluster.

```
Input: Snapshot of entire cluster (hosts and VMs)
I_c ← σ(N_j) /* standard deviation over all hosts*/
NumMigrations ← 0
while I_c > T and NumMigrations < MaxMigrations do
    BestMigration ← NULL
    Max_δ ← 0
    foreach VM v in the cluster do
        foreach compatible destination host h do
            δ̂ ← improvement in imbalance I_c when v is migrated to h
            if benefit of migration > cost then
                if δ̂ > Max_δ then
                    BestMigration ← migrate v to h
                    Max_δ ← δ̂
    if BestMigration is NULL then
        break
    Apply BestMigration to the algorithm's internal cluster state and update I_c
    NumMigrations++
```

Figure IV
Load Balancing Algorithm

Feature Enhancement & Conclusion

Our work is still in the initial stage. In this paper we are done with our main goal of achieving synchronization, mapping and load balancing between virtual machines. In future our target will be achieving more performance result i.e. less time for creation of inverted load balance in data centers and for synchronizing virtual machines. In future we can work on the following aspects

-Discovery and mapping of storage arrays, Monitoring Storage.
- End-to-end discovery of VMs, ESX servers and their Storage.
- Storage Provisioning and management for VMFS and NFS.
- Fast clones of VMs with or without VMware View integration.
- VM Backup recovery.
- Automated virtual infrastructure reporting.
- Mass replication of VMs at a data store level.

- Integration with security software.

References

1. VMware, http://www.vmware.com/.
2. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of Symp. on Operating Systems Principles (SOSP)*, 2003.
3. M. Nelson, B. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *USENIX Annual Technical Conference*, 2005.
4. C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI*, 2005.
5. "IBM Storage Virtualization: Value to you," IBM Whitepaper, May 2006.
6. EMC Invista http://www.emc.com/products/software/invista/invista.jsp.
7. IDC, "Virtualization across the Enterprise," Nov 2006.
8. T. Clark, *Designing Storage Area Networks*. Addison-Wesley, 1999.
9. R. Goldberg, "Survey of virtual machine research," *IEEE Computer*, vol. 7, no. 6, pp. 34–45, June 1974.
10. Hewlett Packard Systems Insight Manager, http://h18002.www1.hp.com/ products/servers/management/hpsim/index.html
11. IBM TotalStorage Productivity Center, http://www-306.ibm.com/software/tivoli/products/totalstorage-data/.
12. DMTF Common Information Model Standards, http://www.dmtf.org/standards/cim.