International Journal of Advanced Computer Research (ISSN (Print): 2249-7277 ISSN (Online): 2277-7970)
Volume-4 Number-4 Issue-17 December-2014

Affinity Aware VM Colocation Mechanism for Cloud

Nilesh Pachorkar[1*] and Rajesh Ingle[2]
Received: 24-December-2014; Revised: 12-January-2015; Accepted: 12-January-2015

Abstract

The most of the existing virtual machine (VM) placement algorithms do not consider the affinity i.e. dependency between the virtual machines. Also, in many existing virtualized systems it is found that the network bandwidth often becomes the bottleneck resource which can lead to the performance degradation of the applications deployed in the cloud. In this paper, we present an affinity aware VM colocation mechanism for cloud. The proposed mechanism find out the network affinity between the virtual machine pairs and colocate the virtual machine pairs having higher network affinity on the same physical host. The VM colocation helps to decrease the network overhead in the cloud and also helps to improve the performance of the communication intensive applications deployed in the cloud. The experimental results show that the runtime of the communication intensive applications, i.e. RUBiS and Twitter, is reduced by few seconds. Also, the network traffic between the virtual machines running the RUBiS application is reduced by 28% while the network traffic between the virtual machines running the Twitter application is reduced by 11%.

Keywords

Virtualization, page sharing, memory affinity, network affinity, VM placement, VM colocation.

1. Introduction

Cloud computing is the computing model in which the resources are provided to the cloud consumer on demand and for which the consumer pays as per the pay-per-use model. Virtualization is the key technology that enables the cloud computing. Virtualization allows the multiple virtual machines to run on the same physical machine and to share the resources of that physical machine.

The modern data centers are employing virtualization which helps to improve the resource utilization of the data centers. The data centers consist of large number of physical servers and the applications run inside the virtual machines which are deployed on these servers. The main objective of the cloud service providers is to optimize the resource utilization so that more virtual machines can be deployed while maintaining the promised SLA to the cloud consumers at the same time.

At the time of virtual machine placement in the cloud the virtual machines which belong to the same application or the virtual machines having dependency between them may get placed on the same host or on different hosts. When such virtual machines get placed on different hosts, the application's performance gets degraded to some extent due to the network overhead. Also, it restricts us from taking the full advantage of the content based page sharing (CBPS) mechanism provided by most of the hypervisor, which consolidates the memory pages having similar content of different virtual machines hosted on same physical machine. The colocation of the communicating virtual machines on the same physical host can help us to reduce the network overhead in the cloud and also to improve the performance of the applications deployed in the cloud.

2. Literature Review

The most of the existing VM placement algorithms consider only the physical characteristics of the physical machines such as CPU, RAM, disk space, etc. but do not consider the communication pattern between the virtual machine pairs. Such algorithms can be called as the non-affinity aware VM placement algorithms. Very little work has been done in the area of the affinity aware VM placement algorithms. The affinity aware algorithms considers

*Author for correspondence

Nilesh Pachorkar, Department of Computer Engineering, P.I.C.T. College, Pune, India.
Rajesh Ingle, Department of Computer Engineering, P.I.C.T. College, Pune, India.

different types of affinities such as network affinity, memory affinity and CPU affinity during the placement of virtual machines onto the physical machines.

A decentralized affinity-aware migration technique is proposed by Jason Sonnek et al. [1] that identify the network affinity between pairs of virtual machines. The VM placement is adjusted dynamically by migrating virtual machines with the help of bartering algorithm. Jianhai Chen et al. [2] proposed a VM placement algorithm called affinity aware grouping for allocation (AAGA) of VMs. This algorithm identifies the affinity relationship between the groups of virtual machines. The authors also defined some rules for performing affinity-aware grouping of VMs. The grouping method based on the heuristic bin packing algorithm is used to place the affine VM groups on the physical machines. Timothy Wood et al. [3] developed a system called Memory Buddies can identify the VMs with higher page sharing potential and colocate them onto the same physical host. The authors also developed a memory fingerprinting technique to identify VMs with high page sharing potential. The system uses a VM colocation algorithm to colocate the virtual machines with high page sharing potential onto the same host. Sujesha Sudevalayam et al. [4] build models to predict the resource usages when moving from non-colocated placements to colocated placements and vice-versa. These models can be used as an input for consolidation and splitting decisions. The authors also showed that how the VM colocation and VM dispersion affect the CPU usage of virtual machines. The authors argue that there is need to consider network affinity between the virtual machines during resource provisioning. Michael Sindelar et al. [5] developed sharing-aware algorithms to colocate virtual machines with identical content on the same physical host. The virtual machines that are colocated on the same physical host can share memory pages having similar content.

3. System Overview

The system architecture is shown in the figure 1. The system is tested on the Openstack cloud [9] which is deployed on three machines. Out of the three machines (nodes) one is the controller node and other two are compute nodes. The compute service is running on all the three nodes so that virtual machines can be spawned and run on all three nodes. Three nodes are connected through LAN. All three nodes are running Ubuntu 12.04. Different cloud applications such as RUBiS [6], TPC-W [7] and Twitter [8] are used to conduct the experiments.

- RUBiS is a free, open source web application that emulates an eBay like auction website. It implements the basic functionality of auction website such as selling, bidding and browsing [6]. We used the latest 1.4.3 version of RUBiS and implemented it with Apache, PHP and MySQL database.
- TPC-W is a transactional web e-commerce benchmark [7]. It emulates the Amazon like online book store. We have implemented the TPC-W benchmark with Java servlets, Tomcat web server and MySQL database.
- Twitter is one of the on-line transaction processing (OLTP) benchmark which is used for the benchmarking of the DBMS. This workload is inspired by the popular Twitter website [8]. We have used Apache and MySQL for the implementation of Twitter workload.

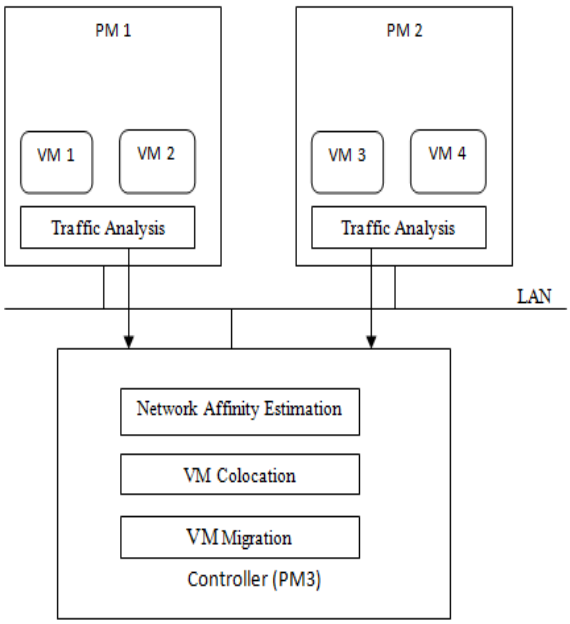


Figure 1: System Architecture

The virtual machines related to each application get placed on different physical machines by the default VM placement algorithm of the Openstack cloud. The traffic analysis module in each physical machine captures the network packets that are going out and

coming into that physical machine (PM). The network communication is captured with the help of tcpdump tool and the number of packets captured between the pair of virtual machines is sent to the network affinity estimation module which is located in the controller node. The network affinity estimation module computes the network affinity between every VM pair with the help of data send by a traffic analysis module of every compute node. It then sorts the VM pairs in descending order according to the network affinity. This sorted list is then sent to the VM colocation module. The VM pairs are then placed according to the proposed algorithm. Finally, the virtual machines that need to be migrated to achieve the desired colocation are identified and migrated.

The algorithms for the virtual machine (VM) colocation and for the placement of the virtual machines which don't have an affinity with other virtual machines are given below.

Algorithm 1: To colocate virtual machines

```
1:  k=1
2:  vm_pair_count = no. of VM pairs to be placed
3:      for i=1 to vm_pair_count do
4:          if vm_pair i is not placed then
5:              vm1=vm id of first vm in vm_pair i
6:              vm2= vm id of second vm in vm_pair i
7:                  if (resource_requirement(vm1) +
                        resource_requirement(vm2)) ≤
          (available_resource_capacity (pm k)) then
8:                      assign vm 1 to PM k
9:                      assign vm 2 to PM k
10:                 set flag to 1 to indicate vm_pair i
is placed
11:                 update the resource availability of
PM k
12:             else
13:                     k++
14:             end if
15:         end if
16:     end for
17:     place_remaining( )   //Place remaining virtual
machines.
```

Algorithm 2: To place remaining VMs those have no affinity with other VMs

```
1:  vm_count = total number of virtual machines
(VMs) present in the cloud.
2:  pm_count = total number of physical machines
(PMs) in the cloud.
3:  for i =1 to vm_count do
4:      for j=1 to pm_count
5:          if VM i is not placed then
6:              if (resource requirement of VM i) ≤
(available_resource_capacity of PM j) then
7:                  assign VM i to physical machine j
8:                  update resource availability of
physical machine j
9:              end if
10:         end if
11:     end for
12: end for
```

4. Experimental Evaluation

4.1 Experimental Setup

We conducted our experiments on multi-node Openstack Cloud which is deployed on three nodes. Two nodes are 2.9 GHz Intel i5 Processor with 8 GB Ram and 100 GB disk space and one node is 3.1 GHz Intel i5 Processor with 4 GB RAM and 100 GB disk space. All three nodes are connected in a LAN and running Ubuntu 12.04 operating system. The Openstack Icehouse version is used along with the KVM hypervisor. To generate the workload in the cloud three applications such as RUBiS, TPC-W and Twitter are used. The RUBiS benchmark is implemented in three virtual machines (Client VM, Web Server VM and DB Server VM) which are deployed on three different physical machines. The TPC-W benchmark is implemented as a standalone application in three virtual machines. The Twitter benchmark is implemented in three virtual machines (one Web Server VM and two Client VMs) which are also deployed on three different physical machines.

4.2 Experimental Results and Analysis

In this section we demonstrate the effectiveness of the VM colocation mechanism by conducting two sets of experiments. We measured two metrics, i.e. the runtime of applications and traffic between VM pairs before and after the colocation of communicating virtual machines in first and second experiment respectively.

The average runtime of benchmark applications before and after the VM colocation is shown in following graphs. From the graphs we can observe that the runtime of Twitter application is reduced by 2 seconds while the runtime of RUBiS application is reduced by 4 seconds.

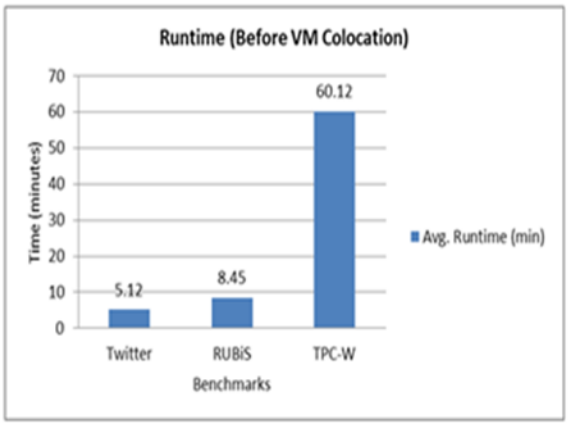


Figure 2: Average runtime of benchmark applications before VM colocation

The runtime of TPC-W application before and after the VM colocation remains the same, because the TPC-W benchmark application is installed as a standalone application in three different virtual machines. So, there is no communication between these three virtual machines and thus the colocation of communicating virtual machines will not affect the runtime of TPC-W application.

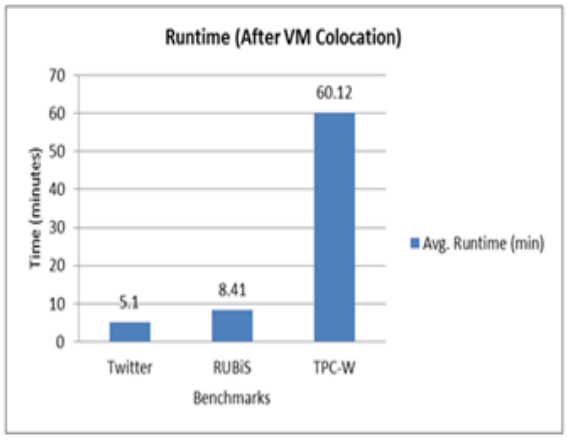
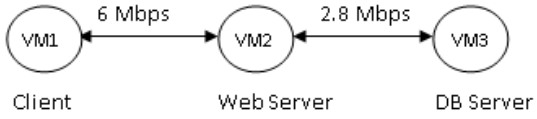
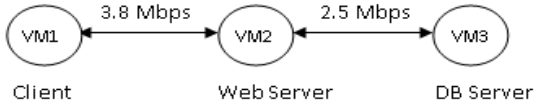


Figure 3: Average runtime of benchmark applications after VM colocation

The runtime of application decreases because the VM colocation mechanism colocates the communicating virtual machines on the same physical machine. Due to the colcoation of the communicating virtual machines the network traffic in the cloud decreases by the considerable amount which is evident from the following figures. The network traffic between the communicating virtual machines of the RUBiS and Twitter applications is shown in Figure 4 and Figure 5 respectively. As the TPC-W application is installed as a standalone application in each of the three VMs, there is no communication between these three virtual machines. In case of RUBiS application, the traffic rate between the client and web server is 6 Mbps and the traffic rate between the web server and DB server is 2.8 Mbps before the VM colocation. The VM colocation mechanism colocates these communicating VMs on the same physical machine which leads to the decreased traffic rate between the communicating VMs of RUBiS application. The traffic between the client and web server reduced to 3.8 Mbps while the traffic rate between web server and DB server reduced to 2.5 Mbps. Thus, the traffic between communicating virtual machines of RUBiS is reduced by 28%.



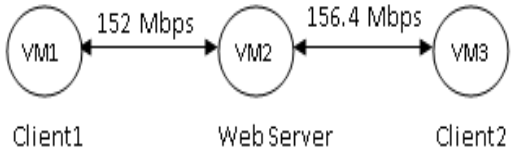
a) Before VM colocation



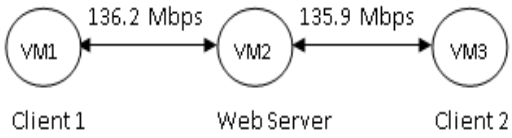
b) After VM colocation

Figure 4: Traffic rate between three VMs running RUBiS benchmark application

In case of Twitter application, there are two client virtual machines which are communicating with the web server virtual machine. The traffic rate between the client1 and web server is 152 Mbps and the traffic rate between the client2 and web server is 156.4 Mbps before the VM colocation. The VM colocation mechanism also colocates these virtual machines. The traffic rate between the client1 and web server reduced to 136.2 Mbps while the traffic rate between client2 and web server reduced to 135.9 Mbps. The Traffic between the communicating virtual machines of Twitter application is reduced by 11%.



a) Before VM colocation



b) After VM colocation

Figure 5: Traffic rate between three VMs running Twitter benchmark application

5. Conclusion

In this paper, we have proposed an affinity aware VM colocation mechanism which colocates the communicating virtual machines on the same physical machine. The main objective of the proposed mechanism is to reduce the network overhead in the cloud and improve the performance of the applications deployed in the cloud by minimizing the runtime of the applications. We conducted several experiments and found that the VM colocation mechanism can help us to reduce the network overhead in the cloud and also to improve the performance of the applications deployed in the cloud.

References

[1] Sonnek, Greensky, Reutiman, and Chandra. "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration." 39th Int. Conference on Parallel Processing (ICPP'10), 2010.

[2] Chen, Chiew, Ye, Zhu, and Chen. "AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines." IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 235-242, 2013.

[3] Wood, Tarasuk-Levin, Shenoy, Desnoyers, Cecchet, and Corner. "Memory buddies: exploiting page sharing for smart colocation in virtualized data centers." In Proceedings of the 2009 ACM SIGPLAN/ SIGOPS International conference on Virtual execution environments (VEE), 2009.

[4] Sudevalayam and Kulkarni. "Affinity-Aware Modeling of CPU Usage for Provisioning Virtualized Applications." In Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD), 2011.

[5] Sindelar, Sitaraman, and Shenoy. "Sharing-aware algorithms for virtual machine colocation," In Proceedings of the 23rd annual ACM symposium on Parallelism in algorithms and architectures (SPAA) 2011.

[6] RUBiS benchmark. http://rubis.ow2.org/. October, 2013.

[7] Smith. "TPC-W: Benchmarking an Ecommerce Solution." http://www.tpc.org/tpcw/. October, 2013.

[8] Twitter (OLTP benchmark). http://oltpbenchmark.com/wiki/index.php?title=Main_Page.

[9] Openstack cloud. http://www.openstack.org/. October, 2013.

Nilesh Pachorkar received the B.E. degree in computer engineering from Pune University, India. He is currently pursuing M.E. as PG Scholar in the department of computer engineering, PICT, Pune University, India. His research interest includes distributed systems and cloud computing.
Email: nbpachorkar@gmail.com

Rajesh Ingle is a professor of computer engineering, PICT, Pune. He received Ph.D. CSE from IIT Bombay, B.E. Computer from PICT, Pune University, and M.E. Computer from COEP, M.S. Software Systems from BITS, Pilani. His research interests include distributed system security, grid middleware, cloud computing, multimedia networks and spontaneously networked environments. He has more than 20 research publications in conferences and Journals.