www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 7 July 2015, Page No. 13230-13236

Enhance Distribution of Load in Cloud

Rashi Saxena[1], Tarun Gupta[2]

[1]PG Scholar, Jayoti Vidyapeeth Women's University,
Jaipur, Rajasthan, India
rashi.saxena@yahoo.co.in

[2]Associate Level 2, Sapient Nitro Corporation,
Bangalore, Karnatka, India
dtarun2009@gmail.com

Abstract: ***This paper proposes and evaluates an approach to the parallelization, deployment and management of applications that integrates several emerging technologies for distributed computing. The proposed approach uses the Map Reduce paradigm to parallelize tools and manage their execution, machine virtualization to encapsulate their execution environments and commonly used data sets into flexibly deployable virtual machines. Multi node environment in which one node will act as a gateway client machine can access the cluster through the gateway via REST API. Using this concept we propose a virtual infrastructure gateway that lifts this restriction. Through gateway cloud consumers provide deployment hints on the possible mapping of VMs to physical nodes. Such hints include the collocation and ant collocation of VMs, the existence of potential performance bottlenecks, the presence of underlying hardware features (e.g., high availability), the proximity of certain VMs to data repositories, or any other information that would contribute in a more effective placement of VMs to physical hosting nodes. Oozier will allow REST access for submitting and monitoring jobs. Cloud Computing allows cloud consumers to scale up and down their resource usage based on demand using the Apache Hadoop, using this prototype we are analyzing various techniques for scalability in cloud. It also demonstrates how power-aware policies may reduce the energy consumption of the physical installation.***

Keywords: IaaS Cloud, Cloud Computing, Resource management, Distributed Processing, virtualization, Map Reduce.

1. Introduction

When computer was just invented, data and computer resource were centralized, computer users used terminal to access them, and with the implementation of hardware, personal computer comes into our life. But now it shows a trend that data and computer resources are centralized again which called cloud computing. Now a days, the most frequently used programs are those Internet based services, such as engines, social network services and electronic business, which have millions of users. Every moment those services emit large amounts of data, which brings a problem: how to deal with immense data set. Search engine Google uses a programming model called Map Reduce can process 20PB data per day. Hadoop is an open source implantation of Map Reduce, which is sponsored by Yahoo. As free and open source software, Hadoop is developing fast. Meanwhile the majority of virtual machine technology. VM-based computing infrastructure has coming up such as Amazon EC2 (Elastic cloud computing).In this paper, we propose the design, implementation and evaluation of a cloud gateway, that performs intelligent placement of VMs onto physical node by exploiting user provided deployment hints. Hint realize the placement preference based on the knowledge only by the cloud consumer has regarding the intended usage of the requested VMs. by modeling workload as pattern of data flows , computations, control/synchronizations points, and necessary network connection , users can identify favorable VMs layouts

.these layouts translate to deployment hints, such hints articulate:

1) Resource consumption pattern among VMs

2) VMs that may become a performance bottleneck

And the portion of the requested virtual infrastructure that can be assisted by the existence of special hardware support. For instance, the fact that two VMs in a virtual infrastructure will hold mirrors of a database is only known to cloud consumer. This information should be communicated to the cloud as a deployment hint so that the respective VMs will not be deployed on the same host.

2. Background

2.1 Laod Balancing

Cloud computing is designed to provide on demand resources or services over the Internet, usually at the scale and with the reliability level of a data centre. MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. It is a style of parallel programming that is supported by some capacity-on-demand-style clouds such as Google's BigTable, Hadoop, and Sector. In this paper, a load-balancing algorithm that follows the approach of the Randomized Hydrodynamic Load Balancing technique (more on that in the following sections) is used. Virtualization is used to reduce the actual number of physical servers and cost; more importantly, virtualization is used to achieve efficient CPU utilization of a physical machine. We have implemented MapReduce algorithm on a system using:

- Hadoop 0.20.1.

- Eclipse IDE 3.0 or above (or Rational application Devel oper 7.1).
- Ubuntu 8.2 or above.

Before diving into the MapReduce algorithm, we'll set up the basics of the cloud architecture, load balancing, MapReduce, and parallel programming.

2.1.1Cloud Architecture

The basic Figure 1 shows a detailed picture of the complete system, platforms, software, and how they are used to achieve the goal set.

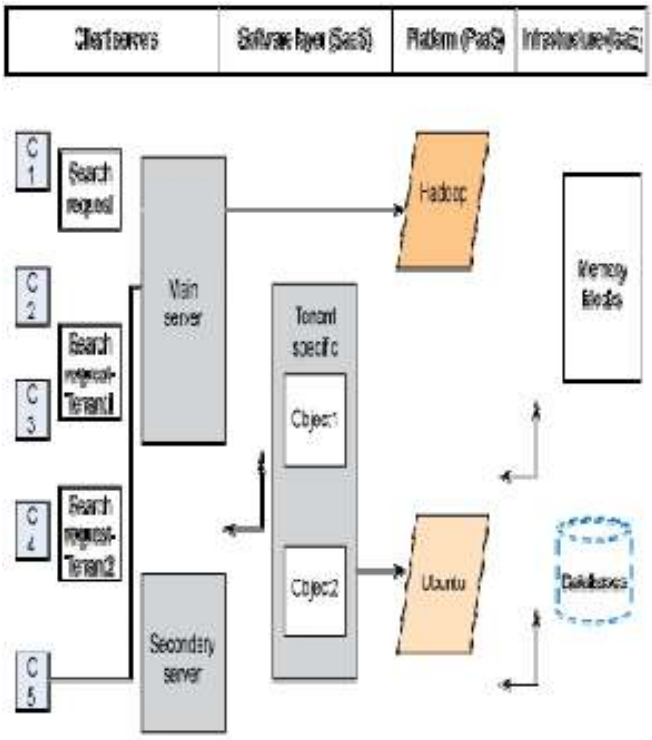

Figure 1: Cloud Architecture

Ubuntu 9.04 and 8.2 is used for the operating systems. Hadoop 0.20.1, Eclipse 3.3.1, and Sun Java 6 for the platforms, the Java language for programming; and HTML, JSP, and XML as the scripting languages. This cloud architecture has both master and slave nodes. In this implementation, a main server is maintained that gets client requests and handles them depending on the type of request. The master node is present in main server and the slave nodes in secondary server. Search requests are forwarded to the NameNode of Hadoop, present in main server as you can see in Figure 2. The Hadoop NameNode then takes care of the searching and indexing operation by initiating a large number of Map and Reduce processes. Once the MapReduce operation for a particular search key is completed, the NameNode returns the output value to the server and in turn to the client.

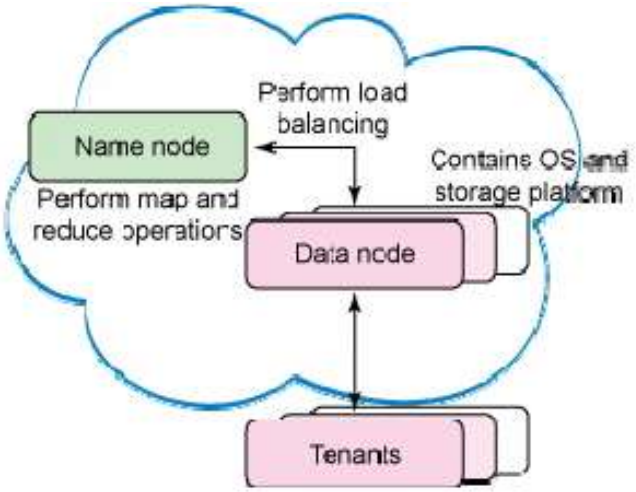

Figure 2: Map and Reduce functions do searching and indexing

If the request is for a particular software, authentication steps are done based on the customer tenant ID, payment dues, eligibility to use a particular software, and the lease period for the software. The server then serves this request and allows the user to consume a selected software combination. If the request is for a particular software, authentication steps are done based on the customer tenant ID, payment dues, eligibility to use a particular software, and the lease period for the software. The server then serves this request and allows the user to consume a selected software combination. The multi tenancy feature of SaaS is provided here, in which a single instance of the software serves a number of tenants. For every tenant specific request there will be a thin line of isolation generated based on the tenant id. These requests are served by a single instance. When a tenant specific client request wants to search files or consume any other multi-tenant software the offerings use Hadoop on the provisioned operating system instance (PaaS). Also, in order to store his data -- perhaps a database or files-- in the cloud, the client will have to take some memory space from the data center (IaaS). All these moves are transparent to the end user.

2.1.2Randomized Hydrodynamic Load balancing

Load balancing is used to make sure that none of your existing resources are idle while others are being utilized. To balance load distribution, it can migrate the load from the *source nodes* (which have surplus workload) to the comparatively lightly loaded *destination nodes* [5]. When it applies load balancing during runtime, it is called *dynamic load balancing*— this can be realized both in a direct or iterative manner according to the execution node selection:

- In the iterative methods, the final destination node is determined through several iteration steps.
- In the direct methods, the final destination node is selected in one step.

For this article, the Randomized Hydrodynamic Load Balancing method is used, a hybrid method that takes advantage of both direct and iterative methods.

2.1.3Map Reduce

MapReduce programs are designed to compute large volumes of data in a parallel fashion. This requires dividing the workload across a large number of machines. Hadoop provides a systematic way to implement this programming paradigm.

The computation takes a set of input key/value pairs and produces a set of output key/value pairs. The computation involves two basic operations: Map and Reduce. The Map operation, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate Key #1 and passes them to the Reduce function. The Reduce function, also written by the user, accepts an intermediate Key #1 and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just an output value of 0 or 1 is produced per Reduce invocation. The intermediate values are supplied to the user's Reduce function via an iterate (an object that allows a programmer to traverse through all the elements of a collection regardless of its specific implementation). This allows you to handle lists of values that are too large to fit in memory. Take the example of WordCount problem. This will count the number of occurrences of each word in a large collection of documents. The Mapper and Reducer function will look like Listing 1. The Map function emits each word plus an associated count of occurrences. The Reduce function sums together all counts emitted for a particular word. This basic functionality, when built over clusters, can easily turn into a high-speed parallel processing system. Performing computation on large volumes of data has been done before, usually in a distributed setting. What makes Hadoop unique is its simplified programming model — which allows the user to quickly write and test distributed systems — and it's efficient, automatic distribution of data and work across machines and in turn utilizing the underlying parallelism of the CPU cores.

In a Hadoop cluster you have the following nodes:

```
mapper (filename, file-contents):
  for each word in file-contents:
    emit (word, 1)

reducer (word, values):
  sum = 0
  for each value in values:
    sum = sum + value
  emit (word, sum)
```

Listing1. Map and Reduce in a WordCount problem

- The NameNode (the cloud master).
- The DataNodes (or the slaves).

Nodes in the cluster have preloaded local input files. When the MapReduce process is started, the NameNode uses the JobTracker process to assign tasks which have to be carried out by DataNodes, through the TaskTracker processes. Several Map processes will run in each DataNode and the intermediate results will be given to the combiner process which generates, for instance, the count of words in file of one machine as(in case of a WordCount problem). Values are shuffled and sent to reduce processes which generate the final output for the problem of interest.

2.1.4 How Load Balancing is used

Load balancing is helpful in spreading the load equally across the free nodes when a node is loaded above its threshold level. Though load balancing is not so significant in execution of a MapReduce algorithm, it becomes essential when handling large files for processing and when hardware resources use is critical. As a highlight, it enhances hardware utilization in resource-critical situations with a slight improvement in performance. A module was implemented to balance the disk space usage on a Hadoop Distributed File System cluster when some data nodes became full or when new empty nodes joined the cluster. The balancer (Class Balancer tool) was started with a threshold value; this parameter is a fraction between 0 and 100 percent with a default value of 10 percent. This sets the target for whether the cluster is balanced; the smaller the threshold value, the more balanced a cluster will be. Also, the longer it takes to run the balancer. (Note: A threshold value can be so small that you cannot balance the state of the cluster because applications may be writing and deleting files concurrently.) A cluster is considered balanced if for each data node, the ratio of used space at the node to the total capacity of node (known as the *utilization of the node*) differs from the the ratio of used space at the cluster to the total capacity of the cluster (*utilization of the cluster*) by no more than the threshold value. The module moves blocks from the data nodes that are being utilized a lot to the poorly used ones in an iterative fashion; in each iteration a node moves or receives no more than the threshold fraction of its capacity and each iteration runs no more than 20 minutes. In this implementation, nodes are classified as *highly- utilized*, *average-utilized*, and *under-utilized*. Depending upon the utilization rating of each node, load was transferred between nodes and the cluster was balanced. The module worked like this:

- First, it acquires neighborhood details:

1. When the load increases in a DataNode to the threshold level, it sends a request to the NameNode.

2. The NameNode had information about the load levels of the specific DataNode's nearest neighbors.

3. Loads are compared by the NameNode and then the details about the freest neighbor nodes are sent to the specific

Data Node.

- Next, the DataNodes go to work:

1. Each DataNode compares its own load amount with the sum of the load amount of its nearest neighbors.

2. If a DataNode's load level is greater than the sum of its neighbors, then load-destination nodes (direct neighbors AND other nodes) will be chosen at random.

3. Load requests are then sent to the destination nodes.

- Last, the request is received:

1. Buffers are maintained at every node to received load requests.

2. A message passing interface (MPI) manages this buffer.

3. A main thread will listen to the buffered queue will service the requests it receives.

4. The nodes enter the load-balancing-execution phase.

2.1.5Evaluating the Performance

Different sets of input files were provided, each of different size, and executed the MapReduce tasks in both single- and

two-node clusters. The corresponding times of execution were measured and we came to the conclusion that running MapReduce in clusters is by far the more efficient for a large volume of input file. The graphs in Figure 3 illustrate our performance results from running on various nodes. Our analysis with Hadoop MapReduce and load balancing lead to two inescapable conclusions:

- In a cloud environment, the MapReduce structure increases the efficiency of throughput for large data sets. In contrast, you wouldn't necessarily see such an increase in throughput in a non-cloud system.
- When the data set is small, MapReduce and load balancing do not affect an appreciable increase in throughput in a cloud system.

Therefore, consider a combination of MapReduce style parallel processing and load balancing when planning to process a large amount of data on your cloud system.

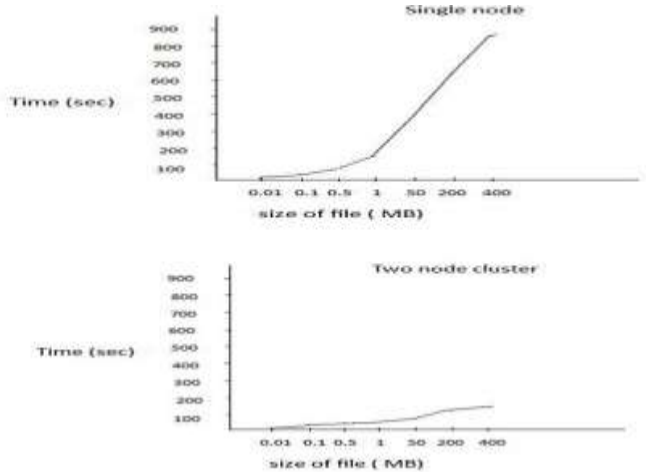

Figure 3: MapReduce load balancing works more efficiently in clusters

3. Vitalizing Technology

Virtualization is a kind of technologies which can make computing element running on virtual machines rather than on physical ones. There are a lot of virtualization technologies, but we focus on the two technologies which are free and open source software and have been widely used.

1) Xen - Xen is a virtual-machine monitor providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently. .It is originally developed by University of Cambridge Computer Laboratory [8]. Xen is free software and licensed under the GNU General Public License.

2) KVM - Kernel-based Virtual Machine (KVM) is a virtualization infrastructure for the Linux kernel. KVM supports native virtualization on processors with hardware virtualization extensions [9] and also KVM is free and open source software.

Virtualization is the process of converting from a purely physical implementation to one using a hypervisor (examples include VMware's ESXi and the Xen hypervisor) which abstract the underlying physical hardware and provide an idealized, or virtual, implementation upon which some higher-level services and/or implementations can be designed and built. Once a physical cluster is virtualized, then higher level services, such as cloning a data node or providing high-availability to a specific node, or providing user controlled provisioning, can be built [2]. A Private Cloud is a collection of virtualized physical hardware that has added services such as catalogs of software or defined platforms that a customer can control [6]. A private cloud differs from a public cloud in that it is generally owned and or managed by the same company or group as the customer.

4. Hint Based Workflow Execution

Apache Oozie [11] is a Java Web application used to schedule Apache Hadoop jobs. It combines multiple jobs s quentially into one logical unit of work. It is integrated with the Hadoop stack and supports Hadoop jobs for Apache MapReduce, Apache Pig, Apache Hive, and Apache Sqoop. It can also be used to schedule jobs specific to a system, like Java programs or shell scripts. There are two basic types of Oozie jobs:

- **Oozie Workflow** jobs are Directed Acyclical Graphs (DAGs), specifying a sequence of actions to execute. The Workflow job has to wait.
- **Oozie Coordinator** jobs are recurrent Oozie Workflow jobs that are triggered by time and data availability.
- **Oozie Bundle** provides a way to package multiple coordinator and workflow jobs and to manage the lifecycle of those jobs.

Apache Oozie [11] allows Hadoop administrators to build complex data transformations out of multiple component tasks. This allows for greater control over complex jobs and also makes it easier to repeat those jobs at predetermined intervals. It helps administrators derive more value from their Hadoop investment. Oozie Workflow is a collection of actions arranged in a Directed Acyclic Graph (DAG). Control nodes define job chronology, setting rules for beginning and ending a workflow, which controls the workflow execution path with decision, fork and join nodes. Action nodes trigger the execution of tasks.Oozie triggers workflow actions, but Hadoop MapReduce executes them. This allows Oozie to leverage other capabilities within the Hadoop stack to balance loads and handle failures. Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique call back HTTP URL to the task, thereby notifying that URL when it's complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion. Often it is necessary to run Oozie workflows on regular time intervals, but in coordination with unpredictable levels of data availability or events. In these circumstances, Oozie Coordinator allows you to model workflow execution triggers in the form of the data, time or event predicates. The workflow job is started after those predicates are satisfied. Oozie Coordinator can also manage multiple workflows that are dependent on the outcome of subsequent workflows. The outputs of subsequent workflows become the input to the next workflow. This chain is called a "data application pipeline". In Hadoop implementation one node which will act as a gateway as shown in fig 4. Client machines can access the cluster through the gateway via the REST API. HttpFS will be used to allow REST access to HDFS, and Oozie will allow REST access for submitting and monitoring jobs. Hints are expressed as sets of conditions

or constraints pointing out a deployment favoring specific task-flows within the virtual infrastructure.

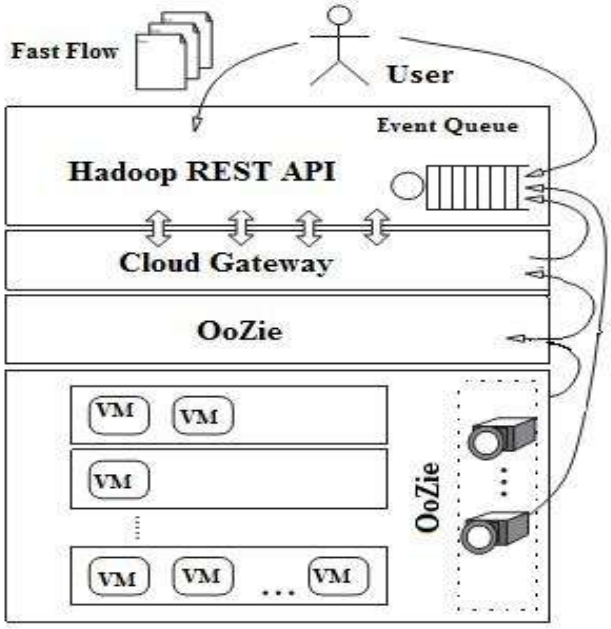

Figure4. Oozie structured layout and interaction model

Each constraint can also be coupled with a weight value indicating its importance relative to the other hints provided. Deployment Profile can be termed as VM-to-host mapping With V being all the VMs to be deployed and H the set of physical nodes, a profile M is a function from V to H it can be represented as $(M : V \rightarrow H)$. As illustrated in fig. 5.

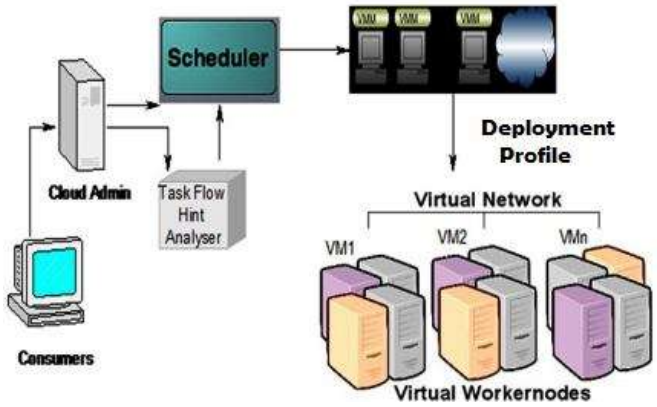

Figure5. Operational Model

Through the Gateway as shown in fig 4, cloud consumers provide deployment hints on the possible mapping of VMs to physical nodes. Such hints include the collocation and anticollocation of VMs, the existence of potential performance bottlenecks, the presence of underlying hardware features (e.g., high availability), the proximity of certain VMs to data repositories, or any other information that would contribute in a more effective placement of VMs to physical hosting nodes. Consumers designate only properties of their virtual infrastructure and remain at all times agnostic to the cloud internal physical characteristics [3]. The set of consumer-provided hints is augmented with high level placement policies specified by the cloud administration. It adds a layer between the user and the infrastructure providing IaaS-cloud services, shown in Fig .this Gateway interfaces with the lower level cloud services that handle the VM lifecycle and perform fundamental administrative tasks. This interface, denoted as a cloud API, allows us to query for specific aspects of the hardware resources as well as manage the VM deployment and migration. During operation, it has to obtain the following information:

Physical node properties: These properties include free memory, total memory, CPU utilization, the name/ID of each hosting node, the amount of free disk space, and redundant hardware enhancing the node's availability.

Physical infrastructure properties: It takes into account the network topology of the physical substrate, the cloud's gateways toward the Internet and any data repositories available through the network.

The current status of each VM: In our approach, each VM may find itself in either STAGING or RUNNING state. A VM is considered to be STAGING when management operations such as disk. image copying during a VM migration do not permit the VM to run.

VM properties: These are similar to the properties acquired for physical hosting nodes. VM properties include the memory usage and the disk space reserved for each VM. It also acquires the IP address of each VM through the cloud API and forwards it to the user. Some commonly used constraints are listed in Table1.

5. Scalability

Hadoop includes a Java implementation of the MapReduce framework, its underlying components and the necessary large scale data storage solutions. Although application programming is mostly done in Java, it provides APIs in different languages such as Ruby and Python, allowing developers to integrate Hadoop to diverse existing applications. It was first inspired by Google's implementation of MapReduce and the GFS distributed file system, absorbing new features as the community proposed new specific sub projects and improvements. Currently, Yahoo is one of the main contributors to this project, making public the modifications carried out by their internal developers. The basis of Hadoop and its several sub projects is the Core, which provides components and interfaces for distributed I/O and file systems. The Avro data serialization system is also an important building block, providing cross-language RPC and persistent data storage. On top of the Core, there's the actual implementation of MapReduce and its APIs, including the Hadoop Streaming, which allows flexible development of Map and Reduce functions in any desired language. A MapReduce cluster is composed by a master node and a cloud of several worker nodes. The nodes in this cluster may be any Java enabled platform, but large Hadoop installations are mostly run on Linux due to its flexibility, reliability and lower TCO. The master node manages the worker nodes, receiving jobs and distributing the workload across the nodes. In Hadoop terminology, the master node runs the JobTracker, responsible for handling incoming jobs and allocating nodes for performing separate tasks. Worker nodes run TaskTrackers, which offer virtual task slots that are allocated to specific map or reduce tasks depending on their access to the necessary input data and overall availability. Hadoop offers a web management

Cloud Consumer Constraints	
FavorVM	Try to reserve a single hosting node for a specific VM.
MinTraf	Deploy on the same host a set of VMs so as to minimize traffic over the physical network
ParVMs	Try to Deploy a set of VMs on separate physical nodes so as not to compete over the same resources.
PinVM	Try not to migrate a specific VM
HighAvail	Try to deploy a specific VM on a host with high availability features.
Cloud-Administration Constraints	
PowerSave	Reduce the number of hosting node use for VM Deployment.
EmptyNode	Offload a specific Hosting node
EvenLoad	Distribute VMs evenly among Hosting Nodes
StopPingPong	Cease the same VM from Migrating back anf forth among hosting nodes.

Table1. Commonly used Constraints

which allows administrators to obtain information on the status of jobs and individual nodes in the cloud. It also allows fast and easy scalability through the addition of cheap worker nodes without disrupting regular operations [12]. another approach is by using CloudStack that is an open source software platform that pools computing resources to build public, private, and hybrid Infrastructure as a Service (IaaS) clouds. CloudStack manages the network, storage, and compute nodes that make up a cloud infrastructure. CloudStack can be used to deploy, manage, and configure cloud computing environments. [10] With CloudStack, you can do things below:

 Set up an on demand elastic cloud computing service. service provider can sell self-service virtual machine instances, storage volumes, and networking configurations over the internet.

 Set up an on premise private cloud for use by employees, rather than managing virtual machine in the same way as physical machine. With CloudStack an enterprise can offer self-service virtual machines to users without involving IT departments. but a problem comes: CloudStack use template to create virtual machines, which makes that all the virtual machines has the same hostname, it will bring conflict to Hadoop. To solve, we introduce Auto Change Hostname Service (ACHS) [10]. When a virtual starts, it firstly run a program, we name it Auto Change Hostname Client (ACHC), ACHC ask ACHS whether this machine is registered, if not, register and request a hostname, then change hostname and write in into OS configuration and run Hadoop services. If ACHC find that this machine has been registered, run Hadoop services immediately.

Conclusion

In this paper we proposed, a hint-based VM scheduler in Hadoop multi node environment that serves as a gateway to IaaS-clouds. Users are aware of the flow of tasks executed in their virtual infrastructures and the role each VM plays. This information is passed to the cloud provider, as hints, and helps drive the placement of VMs to hosts. Hints are also employed by the cloud administration to express its own deployment preferences. Gateway combines consumer and administrative hints to handle peak performance, address performance bottlenecks, and effectively implement high-level cloud policies such as load balancing and energy savings as well as also provides the mechanism to monitor and schedule job in a scalable cloud environment using CloudStack, Our future work is Securing an Apache Hadoop Cluster through a Gateway.

References

[1] Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis, "Hint-Based Execution of workload in Clouds with Nefeli" Proc. IEEE Transactions ON Parallel And Distributed System, VOL, 24, NO.7, July 2013.

[2] M. Rosenblum and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends,"Computer, vol. 38, no. 5, pp. 39-47, May 2005.

[3] H.N. Van, F.D. Tran, and J.-M. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms,"Proc. ICSE Workshop Software Eng. Challenges of Cloud Computing, pp. 1-8, 2009.

[4] X. Wang, D. Lan, G. Wang, X. Fang, M. Ye, Y. Chen, and Q.Q.B. Wang, "Appliance-Based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center,"Proc. FourthInt'l Conf. Autonomic Computing, p. 29, 2007.

[5] M. Zaharia, et al., "Improving MapReduce performance inheterogeneous environments," Proc. Proceedings of the 8th USENIX conference on Operating systems design and implementation, USENIX Association, 2008, pp. 29-42.S. Ibrahim, et al., "Evaluating MapReduce on Virtual Machines: The Hadoop Case," Book Evaluating MapReduce on Virtual Machines: The Hadoop Case, Series Evaluating MapReduce on Virtual Machines: The Hadoop Case 5931,ed., Editor ed. Springer Berlin / Heidelberg, 2009, pp. 519-528

[6] http://www.ibm.com/developerworks/cloud/library/cl-mapreduce/

[7] Xen, http://en.wikipedia.org/wiki/Xen

[8] KVM, http://en.wikipedia.org/wiki/Kernel basedVirtual_Machine.

[9] CloudStack 3.0 InstallGuide.

[10] http://hortonworks.com/hadoop/Oozie

[11] https://www.cloudera.com

Author Profile

RASHI SAXENA received the B.Tech. Degree in Computer Science & Engineering from the Uttar Pradesh Technical University, Lucknow, UttarPradesh, in 2011, and pursuing the M.Tech. Degree in Computer Science & Engineering from the Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan, respectively. Currently, She is an associate Professor of Computer Science & Engineering at Bhaavya Technical Institute, Agra. Her teaching and research areas include cloud computing, load balancing, virtualization, Hadoop framework etc.

Ms. Rashi Saxena may be reached at
rashi.saxena@yahoo.co.in

TARUN GUPTA received the B.Tech. Degree in Computer Science & Engineering from the Uttar Pradesh Technical University, Lucknow, UttarPradesh, in 2011. Currently, he is an associate level 2 at Sapient Corporation Pvt Ltd. His area of interest are cloud computing, J2EE, Spring, Hibernate, Adobe CQ, virtualization etc.

Mr. Tarun Gupta may be reached at
dtarun2009@gmail.com