2012 Third International Conference on Services in Emerging Markets

Enhancement of Hadoop Clusters with Virtualization Using the Capacity Scheduler

Aparna Raj *aparnaraj.k@iiitb.org*
Kamaldeep Kaur *kamaldeep.kaur@iiitb.org*
Uddipan Dutta *Uddipan.Dutta@iiitb.org*
V Venkat Sandeep *Sandeep.VV@iiitb.org*
Shrisha Rao *shrao@ieee.org*

***Abstract*—We present a virtualized setup of a Hadoop cluster that provides greater computing capacity with lesser resources, since a virtualized cluster requires fewer physical machines. The master node of the cluster is set up on a physical machine, and slave nodes are set up on virtual machines (VMs) that may be on a common physical machine. Hadoop configured VM images are created by cloning of VMs, which facilitates fast addition and deletion of nodes in the cluster without much overhead. Also, we have configured the Hadoop virtualized cluster to use capacity scheduler instead of the default FIFO scheduler. The capacity scheduler schedules tasks based on the availability of RAM and virtual memory (VMEM) in slave nodes before allocating any job. So instead of queuing up the jobs, they are efficiently allocated on the VMs based on the memory available. Various configuration parameters of Hadoop are analyzed and the virtualized cluster is fine-tuned to ensure best performance and maximum scalability.**

***Keywords*-Hadoop, virtualization, capacity scheduler, MapReduce, VM cloning, virtualized cluster**

I. Introduction

In this work, we propose a way to enhance the Hadoop [1] cluster with virtualization, which reduces the time and money spent on the infrastructure as well as the manpower needed to maintain it, considerably. This approach increases the hardware utilization and improves performance. Hadoop clusters are widely deployed to process large amounts of data. Though large clusters are preferred due to the improved computing power, setting them up is both costly and time-consuming. After installation of Hadoop, each node in the cluster has to be configured individually for running MapReduce tasks. Also, if the cluster is to be extended later by adding a new physical machine, the whole procedure of installation and configuration of Hadoop on the new machine is to be repeated. Thus, allocation of new resources is time-consuming for a normal cluster. Though Hadoop MapReduce works well on physical systems, quick provisioning of resources is a problem. Setting up and configuring Hadoop on a new physical machine and adding it as a node to an existing Hadoop cluster is time consuming. But with virtualization, a new node configured with Hadoop can be easily created with cloning of an already configured VM.

1) As the computations involved are large, a lot of resources should be quickly coordinated and properly allocated. If there are a large number of requests at a particular time, and if the Hadoop environment requires more resources to process these requests, adding and configuring a new physical machine takes a considerable amount of time.
2) For a large job, adding more TaskTrackers to the cluster helps in faster computations, but there is no flexibility in adding or removing nodes from a Hadoop cluster set up entirely on physical machines.

This work aims at enhancing the Hadoop environment with more virtualized infrastructure so that a Hadoop cluster can be easily extended. If more nodes are required to finish jobs in a cluster, a VM can be cloned to produce a new node. Such a clone can be created from a predefined machine image with the required Hadoop configuration and software. As it is already configured according to the requirements, it can be added to the Hadoop cluster easily, to solve the problem of quick resource provisioning on Hadoop. With multiple VMs running, the overall utilization of the cluster is improved. Integrating a Hadoop cluster with virtualization can solve such management problems for Hadoop. In this work we present a virtualized setup of Hadoop with some nodes set up on VMs. This enhances an existing cluster with quicker resource allocation capabilities and improved performance. VM cloning is done for quicker creation of new nodes in the cluster, and the performance of the cluster is improved by configuring the cluster to use the capacity scheduler. This reduces the hardware footprint, which in turn reduces the cost. This approach also works for Hadoop clusters where some of the nodes are physical machines, i.e., it does not require virtualization of the complete cluster. Along with reducing the cost and time required for cluster setup, the performance of the virtualized cluster is improved by configuring the cluster to use the capacity scheduler instead of the default FIFO scheduler. The capacity scheduler takes the memory capacities of nodes into consideration while scheduling jobs. Memory-intensive jobs are well supported by this scheduler as such jobs can

978-0-7695-4937-8/12 $26.00 © 2012 IEEE
DOI 10.1109/ICSEM.2012.15

specify higher memory requirements if necessary. With the capacity scheduler, jobs get allocated considering how much RAM is available on each node. This particularly suits a virtualized cluster, because though VMs have less RAM when compared to physical machines, jobs are allocated according to the RAM available. This results in faster execution of jobs and hence performance improvement.

To analyze the performance, a virtualized cluster was compared with a cluster consisting of only physical machines. Although both clusters had the same number of physical machines, the number of nodes on the virtualized cluster was more. The execution time on the virtualized cluster was measured against that in the normal cluster. We observed a decrease of about 24 percent in the execution time of virtualized cluster when compared to the normal cluster. The performance of the virtualized cluster with and without capacity scheduler was also analyzed experimentally. With all other parameters remaining the same, the scheduler of the virtualized cluster was changed. The execution times with both FIFO scheduler and capacity scheduler were measured on the virtualized cluster. We observed that job execution took less time with the capacity scheduler and an average improvement of 13 percent was achieved. The improvement was especially significant for larger input sizes.

The rest of this paper is organized as follows. Section II outlines the technologies used in this work, such as Hadoop, capacity scheduler and virtualization. The modified architecture of a Hadoop cluster with virtualization and the configuration details of VM and capacity scheduler are described in Section III. VM cloning and the required network settings for VM are also explained. In Section IV, we present the results showing the performance improvement achieved with the virtual cluster. We conclude our paper in Section V with suggested future work.

II. Background

In this section we give an overview of the working of Hadoop and the capacity scheduler and the advantages of using virtualization.

A. Hadoop

Hadoop has emerged as one of the most popular implementations for MapReduce [1]. Hadoop started with its MapReduce implementation but expanded very fast, and nowadays includes other projects to provide the required infrastructure, such as HDFS (Hadoop Distributed File System) [1] that provides the distributed file system required for a MapReduce implementation, becoming in this way a complete software solution. A Hadoop cluster has one node acting as the master node, and the other

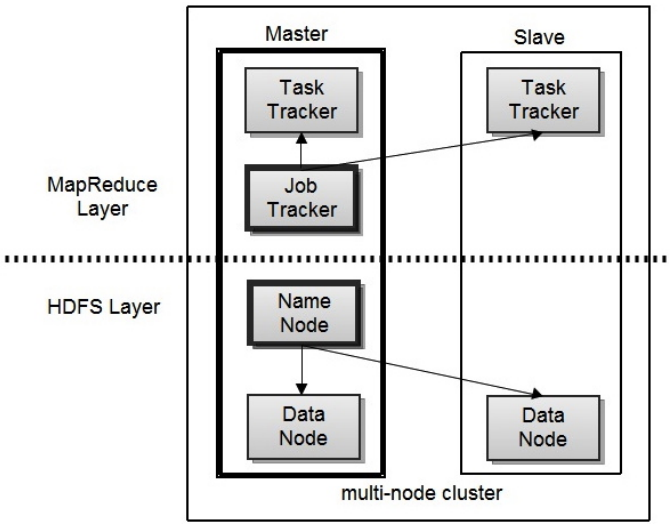

Figure 1. Architecture of Hadoop Cluster [1]

nodes are slave nodes. The Hadoop deamon, which deals with the user-submitted jobs, known as JobTracker, resides on the master node. The JobTracker receives a request from the user, which is known as a job. It then splits the job into different small tasks and schedules these tasks on the TaskTrackers to execute. The TaskTracker is a Hadoop deamon where the actual execution takes place, and each slave node has one TaskTracker. The jobs also include the "map" and "reduce" functions and their configurations.

The Hadoop cluster has a distributed file system within it, known as the HDFS. It also has client server architecture just like MapReduce. The basic architecture of the Hadoop cluster is shown in Figure 1. The central node here is known as the NameNode, and holds all the meta data about the files. It also includes several slave nodes known as DataNodes, which store the data. Thus a master node in the cluster consists of a JobTracker and a NameNode, while each of the slave nodes comprises a TaskTracker and a DataNode. As shown in Figure 1, the JobTracker on the master node controls the TaskTrackers on the slave nodes, and similarly the NameNode on the master node controls the DataNodes on the slave nodes.

The JobTracker creates map and reduce tasks, once the data is available in the DataNodes. The number of map tasks is determined by the size of the input data. The JobTracker keeps track of all running tasks, and when the map tasks are finished, it starts the reduce tasks [2]. Reduce tasks consists of three phases namely, copy, sort and reduce. Normally, the reduce phase waits untill the map phase is finished. But it is also possible to run the first of the reduce task while the partial results of the map phase arrive. As soon as the sort and copy

phases are done, the data is passed to the reduce function and its output is written to the HDFS.

As the JobTracker has to maintain all the other nodes, the overhead on it is considerably higher when compared to the TaskTracker nodes. Similarly a NameNode has more tasks at hand than the DataNodes. So the master node of the cluster is configured on a dedicated physical machine. The master node also has a SecondaryNameNode, for use in case of failure of the NameNode.

B. Capacity Scheduler

Hadoop has a default FIFO scheduler that runs jobs in the order of submission, and supports basic priorities. Alternative and customized schedulers have also been added, particularly for the multi-user and multi-job scenarios. One such scheduler is the capacity scheduler [3], which supports scheduling of tasks on a TaskTracker based on a job's memory requirements and the availability of RAM and Virtual Memory (VMEM) [3] on the TaskTracker node. Jobs are submitted to the multiple queues of the capacity scheduler. The jobs allocated and the fraction of the capacity allotted are balanced by the scheduler to have a uniform allocation. The queues support job priorities as well. Each queue enforces an upper limit on the percentage of resource allocation per user. This prevents a few jobs from dominating the resources. The capacity scheduler takes only the memory capacity of the nodes into consideration while scheduling jobs and hence it is best suited for scheduling memory intensive jobs. By using this scheduler, it is possible to reduce the execution time of jobs, and it also helps achieve improved throughput. The utilization of the cluster can also be increased. Jobs can specify higher memory requirements if necessary, and such jobs are run only on TaskTrackers with more memory.

C. Virtualization

Virtualization is an older software technology that has, within the past decade, become widely used and is well on the way to transforming the practice of IT, changing the manner that computing hardware (particularly for enterprise systems) is used. Virtualization permits the running of multiple virtual machines (VMs) on a single physical machine, sharing the resources of a single hardware unit across multiple virtualized devices. Different virtual machines can run different operating systems and multiple applications on the same physical computer. For instance, a single large blade server can host VMs running a file server, a mail server, a web server, etc. Since a given server is not utilized highly at all times, such virtualization brings about an increase in the effective utilization of the hardware. Organizations can also reduce their "hardware footprint" by running many VMs on a smaller set of physical machines, which in turn brings about savings in costs for power, real estate, and the like.

Likewise, we hold that by virtualizing nodes in a Hadoop cluster, better resource utilization can be ensured. This also enables quicker addition and deletion of nodes without much overhead. Partitioning each node to a number of virtual machines (VMs), gives us a number of benefits [4]:

1) Scheduling: With VMs, hardware utilization can be increased. When more computing capacity is required for scheduling batch jobs, the unused capacity of the hardware can be used by other VMs.
2) Resource Utilization: Different kinds of VMs can be hosted on the same physical machine. Hadoop VMs can be used along with VMs for other tasks. This results in better resource utilization by consolidating different kinds of applications.
3) Datacenter Efficiency: There is a greater variety of the types of tasks that can be run on a virtualized infrastructure, and it is possible to run cross-platform applications as well.
4) Deployment: Deployment times for new nodes in a Hadoop cluster can be greatly reduced by virtualization. Configuring Hadoop on a machine can be done quickly by cloning an already configured VM.

By creating a Hadoop cluster with VMs, better resource utilization can be achieved. The Hadoop environment is modified to use the capacity scheduler. Its capability to schedule tasks based on a job's memory requirements in terms of RAM and virtual memory on a slave node makes it better suited to a virtualized Hadoop cluster.

III. Design And Implementation

To analyze the performance of Hadoop clusters with virtualization, different Hadoop clusters were designed with some of the nodes on virtual machines. Thus the clusters formed contained Hadoop daemons on both physical machines as well as on virtual machines. The architecture, configuration and working of a virtualized Hadoop cluster are explained further in this section.

A. Architecture

A virtualized Hadoop cluster was set up with the master node on a physical machine and slave nodes on VMs. The architecture of the system is depicted in Figure 2.

As shown in Figure 2, the master node of the cluster is set up on a physical machine. The Hadoop daemons, NameNode and JobTracker are run on the master node. Other slave nodes required to form the cluster are set up on VMs. Multiple VMs are set up as slave nodes on

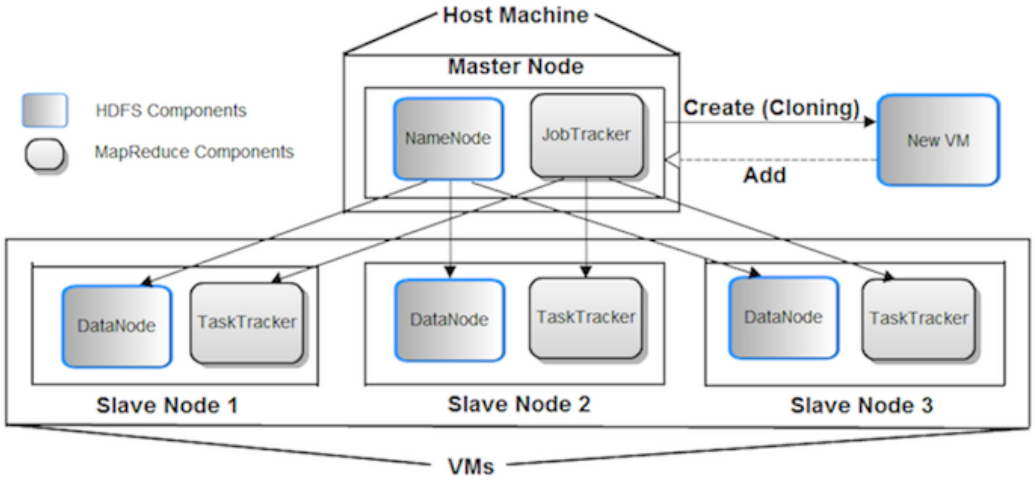

Figure 2. Architecture of Virtualized Hadoop Cluster

which the Hadoop daemons, DataNode and TaskTracker [1] are run. The addition of a new slave node can be easily achieved by cloning of an already configured VM. In order to extend the cluster, a new node can be created by cloning an already configured VM. When the new node is included in the list of slaves in the master node, it also becomes a part of the cluster.

Once the input files are copied to the file system, HDFS distributes it among the DataNodes. Once the data is distributed and made available in the DataNodes, the JobTracker creates map and reduce tasks. Depending on the size of the input, the number of map tasks is determined. On a standard Hadoop cluster, the FIFO scheduler runs jobs in the order of submission. Here in the proposed virtualized cluster, the capacity scheduler is configured, which takes the memory and availability of RAM on the TaskTrackers into consideration while scheduling jobs.

B. VM Installation and Configurations

Oracle VM VirtualBox [5], a software package for creating and maintaining VMs, was used for creating VMs. Oracle VM VirtualBox was installed on an existing host operating system as an application. This host application allows additional guest operating systems, each known as a Guest OS, to be loaded and run, each with its own virtual environment.

VM Installation

First, a virtual hard disk image was created, with base memory 512 MB and a maximum of 8GB storage. For more flexible storage management, a dynamically allocated virtual disk file was created [6], which used space on the physical hard disk as it fills up. Though it does not shrink again automatically when space is freed, it has much better performance than a statically allocated hard disk. Initially the disk was very small and did not occupy any space for unused virtual disk sectors. Thereafter, the disk grows every time a disk sector is written for the first time, until the drive reaches the maximum capacity chosen.

Hadoop on VM

A VM was created with the above specification, and Ubuntu was installed as the guest operating system. The Sun Java was installed on the VM, after adding the necessary repository to the VM [7]. SSH was then installed and configured on the VM. The Hadoop source folder was copied to the VM, by using the `scp` command, i.e., secure copy command. scp is used for secure copying of data. First a single node cluster was run on the VM. After ensuring the correct working of hadoop on VM, it was configured as a slave node in the cluster.

Network Settings for VM

To establish connection between the VM and the host machine, network settings were changed. Bridge utils, vtun and uml utilites [8] were installed on the host machine. A bridge and a tap device were created on the host machine for the guest to use. Two network adapters were enabled in the VM. Along with the default Network Address Translation (NAT) Adapter, a bridged adapter was also configured to use the tap device [8] created on the host. With these settings SSH connection was successfully established between the Host and the VM. The process of creating the bridge and the tap device, on the host machine, was automated. A script was scheduled to run at boot time. Thus at system startup, the bridge and tap device were created automatically. For extending it to multiple machines, multiple tap devices can be created and each VM can be attached to one device.

VM Cloning

A virtual image of the Hadoop configured machine was obtained by cloning the VM. The cloned VM is an exact replica of the machine from which it is cloned. This made the whole process of creating a new Hadoop configured machine very much faster. As the machine is cloned, the new machine also has the same name and ip address as the first one. In order to distinguish between each node in the cluster, the machine names and ip addresses should be distinct. This was fixed by editing the name of the new machine in `/etc/hosts/` and `/etc/hostname`. A new ip address was assigned to the bridged adapter interface by setting the network parameters in `/etc/network/` interfaces. Thus a fully configured slave node was created to add to the Hadoop cluster.

C. Configuration of Capacity Scheduler

The capacity scheduler was built from source by executing the Ant package. The obtained jar file of capacity scheduler was placed in `hadoop/build/contrib/` folder. The Hadoop master node was then configured to use the capacity scheduler instead of the default scheduler.
The configuration settings for the capacity scheduler are as follows.

HADOOP CLASSPATH [7] in `conf/hadoop-env.sh` was modified by specifying the capacity-scheduler jar. Then the file `Mapred-site.xml` [3] was modified to include a new property, `mapred.jobtracker.taskScheduler` with value `org.apache.hadoop.mapred.CapacityTaskScheduler` The configuration is given in Table I

Table I
CONFIGURATION FOR CAPACITY SCHEDULER IN **mapred-site.xml** [3]

```
<property>
<name>mapred.job.tracker.taskScheduler</name>
<value>org.apache.hadoop.Mapred.
capacityTaskSceduler</value>
<description>
The scheduler which is to be used by the jobtracker
</description>
</property>
```

The capacity scheduler does memory based scheduling based on the VMEM or RAM on TaskTracker nodes. It is done as follows [3].

1) After the necessary parameters are set, the capacity scheduler enables scheduling based on VMEM. For this, the free VMEM on a TaskTracker node is computed. On each heartbeat, a TaskTracker node sends the node's total VMEM and the offset. Available VMEM is the total VMEM minus the offset. The free VMEM is then calculated as the difference between the available VMEM and the sum of VMEM task limits on the VMs already allocated to running tasks. Now the scheduler compares this value with the VMEM requirements for the job at hand. If the job's VMEM requirements are less than the available VMEM on the node, the job's task can be scheduled. Otherwise, that task is not scheduled on that TaskTracker. By doing this, the scheduler ensures that jobs with high memory requirements are not left out. It gets executed as and when the TaskTracker has enough VMEM available. In case the job with high memory requirements does not have any task to run, the scheduler goes on to the next job in queue.
2) The capacity scheduler can also consider RAM on a TaskTracker node along with the VMEM. Scheduling based on RAM is done in the same way as that based on VMEM. The total RAM available on a TaskTracker is taken into account along with an offset. Then the available RAM on the node is computed by the scheduler. Users can optionally specify a RAM limit for the jobs submitted just like a VMEM limit can be submitted. The scheduler takes the RAM requirements of a job into account, if specified.
3) Along with taking the specified RAM or VMEM requirements into account, the scheduler also ensures that jobs cannot demand for RAM or VMEM higher than certain limits. If higher RAM or VMEM than the configured limit is specified by any job, then the job is failed when it is submitted.

D. Setting Up Virtual Hadoop Cluster

Virtual clusters of varying size were set up. One of the physical machines was configured to be the master node of the cluster. VMs were configured to be the slave nodes of the cluster. The master node was configured on a physical machine so that the NameNode and JobTracker daemons can be run on a machine with higher capacity. This ensures better performance for the jobs run on the cluster. In order to ensure scalability of the cluster, it was extended to multiple VMs.

After establishing connection between all the nodes of the cluster, MapReduce programs were run. The execution of MapReduce tasks was repeated for different input sizes. Clusters of different size were also configured by repeating the above steps. The time taken for each MapReduce job was analyzed. The performance of the virtual cluster was enhanced by making use of the capacity scheduler, which takes memory storage on each node, into account. The master node was configured to use capacity scheduler and the whole virtual cluster was running MapReduce with capacity scheduling. Running

MapReduce Jobs: Following procedure was followed for running MapReduce [1] tasks.

1) `$HADOOP_PATH` was configured in `/usr/local/hadoop`.
2) Formatting NameNode: The NameNode was formatted prior to starting up the cluster, in case hdfs size changes or new data comes in.
   Command: `$HADOOP_PATH/bin/hadoop` namenode -format
3) Starting Hadoop-Deamons : The Hadoop deamons JobTracker, TaskTracker, NameNode, DataNode and Secondary NameNode was started up.
   Command: `$HADOOP_PATH/bin/start-all.sh`
4) The deamons can also be started individually on each node.
   Command: `$HADOOP_PATH/bin/hadoop-deamon.sh` start <deamon-name>
5) Copying data files : The files on which the MapReduce job is executed is first copied into the HDFS.
   Command: `$HADOOP_PATH/bin/hadoop` dfs-copyFromLocal <local-path> <hdfs-location>
6) Running MapReduce : Once the data files are in place, the MapReduce job is run.
   Command: `$HADOOP_PATH/bin/hadoop` <program-name> <input-path> <output-path>

Automated Running Through Scripts

Scripts were created to run Hadoop MapReduce. Scripts were also created for cloning of VM to obtain hadoop configured VM image. Scripts were also created for the process of starting up VM and setting up network configurations of the VM. The process of hadoop startup in the VM was also automated. A script was created for starting up the Hadoop daemons. The script was scheduled to run at the boot time of the VM. Thus the Hadoop daemons were started up automatically when the VM was started [6].

IV. Performance Metrics

Virtualized clusters of various sizes were set up, and performance was analyzed for varying input sizes by running MapReduce tasks. The variation in execution times was also observed by varying cluster size and configuration parameters. The results of the experiments are presented here.

A. Comparison of Normal and Virtualized Clusters

With the number of physical machines remaining the same, normal and virtualized Hadoop clusters were set up. Normal cluster consisted of only the physical machines as the nodes. Virtualized cluster was setup with VMs in each system along with the physical machine nodes.

Table II
Comparison between runtimes of Normal and Virtualized Cluster

Input Size(MB)	T1(s)	T2(s)	Improvement(%)
10	14	34	58.82
20	30	36	29.73
30	41	45	16.67
40	46	50	8.89
50	48	52	7.69
60	33	50	34.00
70	50	60	16.67

Thus the virtualized cluster had more nodes than the normal cluster even though both had the same number of physical machines. The virtual nodes on each machine were decided according to the system capacity. The performances of both clusters were analyzed by running MapReduce jobs with inputs of varying sizes. The experimental data are listed in Table II as input file size in MB, time taken for execution of the job in normal cluster and time taken for the execution in the virtualized cluster.

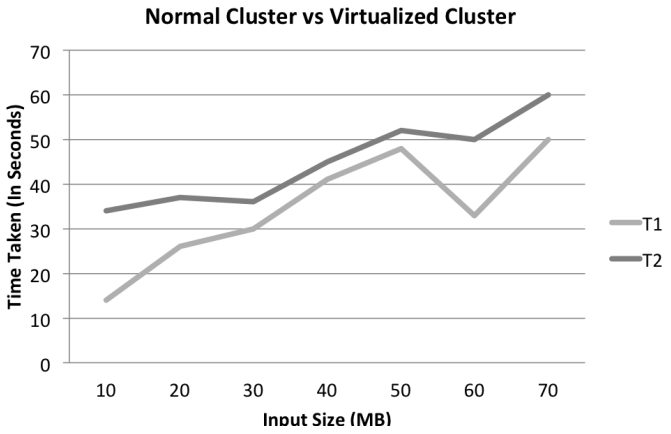

Figure 3. Graph for Normal vs Virtualized Cluster Runtime

The experiment was repeated for the different input sizes, as listed in Table II. In Table II, T1 denotes the time taken for execution by a virtualized Hadoop cluster and T2 denotes the time taken for execution by a normal Hadoop cluster.

Figure 3 shows a comparison between the performance of a normal cluster and a virtualized cluster on a single physical machine. It shows that better performance was achieved with virtual Hadoop cluster, meaning the better resource utilization is achieved with virtualization.

B. Comparison of Virtualized Clusters With and Without Capacity Scheduler

On the virtualized cluster, execution times were measured with the default FIFO scheduler and with the capacity scheduler. While both gave almost similar results for smaller file sizes, as the file size increased, the cluster with the capacity scheduler was found to be significantly faster. The experimental data are listed in Table III as input file size in MB, time taken for the execution in the virtualized cluster configured to use the capacity scheduler and time taken for execution of the job in virtual cluster with FIFO scheduler.

Table III
VIRTUALIZED CLUSTER WITH FIFO SCHEDULER VS VIRTUALIZED CLUSTER WITH CAPACITY SCHEDULER

Input Size(MB)	T1(s)	T2(s)	Improvement(%)
10	35	40	12.5
20	57	60	5
30	67	75	10.67
40	90	94	4.26
50	86	97	11.34
60	78	90	13.33
70	62	94	34.04

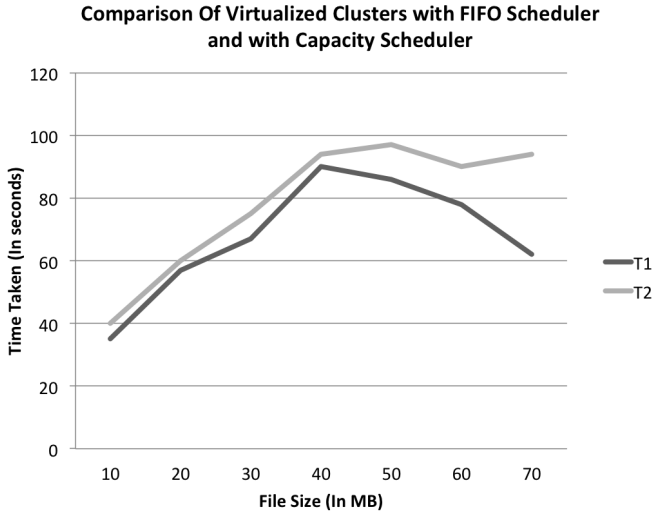

Figure 4. Graph for Comparison Between Virtualized Cluster with FIFO Scheduler and with Capacity Scheduler

In Table III, T1 denotes the time taken for execution on a Hadoop cluster with capacity scheduler and T2 denotes the time taken for execution on the cluster with the default FIFO scheduler.

Figure 4 shows a comparison between the performances of two virtualized clusters. One of the clusters was configured to use the capacity scheduler while the other was configured to use the default FIFO scheduler. The execution times of both clusters were measured. The experiment was repeated for different input sizes listed in Table III. With smaller input sizes, the improvement in execution time is less significant, but as the input file size increases, the capacity scheduler gives significantly better performance than the default FIFO scheduler. The benefits of using virtualization and of using the capacity scheduler are obviously cumulative, but we do not present the experimental comparison of a normal cluster and a virtualized cluster with capacity scheduler.

C. Performance Parameters for Real Time Clusters

In addition to the performance improvement achieved with the use of virtualization, a Hadoop cluster can be fine tuned for even better performance by setting some configuration parameters. There are various Hadoop configuration parameters which directly affect MapReduce job performance under various conditions. These parameters were analyzed in order to improve the performance of the virtaulized cluster. These parameters differ for each cluster depending on the input size, block size etc. Some of the parameters that were analyzed for the virtualized Hadoop cluster are listed below [9].

1) `dfs.block.size`: The input data is split into different blocks before processing. dfs.block.size determines the size of the chunk to which the data is split. Increasing the block size reduces the number of map tasks. It is best to determine DFS block size according to the complexity of the map tasks, i.e., if the computations involved in each map task is such that one data block takes a lot of time, then the number of DFS blocks should be less.
2) Temporary space: If jobs are large, more space is required to store the map output, during execuion. By default, any intermediate data is stored in temporary space. Increasing temporary space is advantageous to large jobs.
3) `mapred.local.dir`: Any temporary Mapreduce data is stored here. More space is advantageous for jobs with large chunks of data.
4) `mapred.map.tasks`: This parameter indicates the number of map tasks (see Section II-A) executed for a job. The number of map tasks for a particular job is determined by the number of DFS blocks in the cluster. For example, if dfs.block.size is 256 MB and input size is 160 GB, minimum number of maps is

$$(160 \times 1024)/256 = 640.$$

Best performance is achieved when the number of map tasks is set to a value approximately equal to the number of map task slots in the cluster.

It can also be a multiple of the number of map slots available. Network traffic is minimized when tasks are sent to a slot on a machine with local copy of the data. Setting the number of map tasks as a multiple of the number of the nodes ensures this and hence results in faster execution. As a rule of thumb, number of map tasks can be set as 10 times the number of slaves (i.e., number of TaskTrackers).

5) `mapred.reduce.tasks`: Number of reduce tasks for the job. After the data is sorted, the reduce output is written to the local file system. The write process requests a set of DataNodes that are used to store the block. If the local host is a DataNode in the file system, the local host is the first DataNode in the returned set. Such a write to a DataNode on localhost is much faster, as they do not require bulk network traffic. Setting the number of reduce tasks as 2 times the number of slaves (i.e., number of TaskTrackers), reduces the network traffic and hence results in better performance.

V. Conclusion and Future work

As Hadoop is widely used for processing huge datasets, improvement of execution time on Hadoop is widely researched upon. Though virtualization has been used with Hadoop, use of the capacity scheduler along with virtualization is a new approach. The advantage of using the capacity scheduler with virtualization is clear from the experimental results obtained. The results show that the approach gives a significant reduction in execution times, which in turn shows that the use of virtualization helps in better utilization of the resources of the physical machines used.

In the context of what Hadoop was designed for, the clusters and data set used in the experiment are both considered small. Though Hadoop is meant to handle much larger data sets running on clusters with many more nodes, the experiments on virtual cluster was conducted on relatively small capacity machines. Given the relatively under power of the machines used in the real cluster the results were fairly relevant. The addition of more machines in the cluster leads to an even greater reduction in runtime. The virtual cluster can be scaled up according to the resources available. The results were especially significant for larger input sizes. This is an indication that when the approach is applied on larger machines and huge clusters, even better results with significant economic benefits would be obtained.

The Hadoop version 0.23.1 released on 27th February, 2012 has improved significantly in the fields of HDFS and MapReduce. It also addresses the issues of having multiple masters in a single cluster. Hence, the scalability issue can be dealt in a better manner in the new version. Setting up of virtual cluster with the latest Hadoop version can bring out much better results. More configuration parameters can be analyzed and the performance of the virtual clusters can be increased by fine tuning the value of relevant parameters.

References

[1] "Hadoop," The Apache Software Foundation, Dec. 2011. [Online]. Available: http://hadoop.apache.org/

[2] D. de Nadal Bou, "Support for managing dynamically Hadoop clusters," in *Master in Information Technology - MTI*, Sep. 2010, Project Director : Yolanda Becerra. [Online]. Available: http://hdl.handle.net/2099.1/9920

[3] "Capacity Scheduler Guide," The Apache Software Foundation, 2008, User Manual.

[4] M. Kontagora and H. GonzalezâĂŞVelez, "Benchmarking a MapReduce Environment on a Full Virtualisation Platform," School of Computing, Robert Gordon University, Aberdeen AB25 1HG, UK, 2010. [Online]. Available: http://www.rgu.ac.uk/computing/

[5] "Oracle VM VirtualBox," User Manual [Accessed : January 19, 2012]. [Online]. Available: http://www.virtualbox.org/manual/

[6] Ravindra, "Building a Hadoop Cluster using VirtualBox," Xebia IT Architects India Private Limited, Oct. 2010. [Online]. Available: http://xebee.xebia.in/2010/10/21/building-a-hadoop-cluster-using-virtualbox/

[7] M. G. Noll, "Running Hadoop on Ubuntu Linux (Multi-Node Cluster)," Aug. 2007, My digital moleskine. [Online]. Available: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

[8] C. Macdonald, "VirtualBox host to guest networking," Oct. 2009, Callum on life. [Online]. Available: http://www.callum-macdonald.com/2009/10/28/virtualbox-host-to-guest-networking/

[9] Impetus, "HADOOP PERFORMANCE TUNING," White Paper, Impetus Technologies Inc., Oct. 2009, Partners in Software R&D and Engineering. [Online]. Available: www.impetus.com

[10] "gliffy," online Diagram Software. [Online]. Available: http://www.gliffy.com/

[11] J. Devine, "Evaluating the Scalability of Hadoop in a Real and Virtual Environment," Dec. 2008, cS380 Final Project. [Online]. Available: jamesdevine.info/wp-content/uploads/2009/03/project.pdf

[12] J. Buell, "A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5," Technical White Paper, VMware, Oct. 2011. [Online]. Available: http://www.vmware.com/files/pdf/techpaper/VMW-Hadoop-Performance-vSphere5.pdf