# Provisioning Hadoop Virtual Cluster in Opportunistic Cluster

Arindam Choudhury, Elisa Heymann, Miquel Angel Senar[1]

*Abstract*— **Traditional opportunistic cluster is designed for running compute-intensive jobs. Data-intensive jobs need distributed and parallel processing. It also needs data-locality to overcome the overhead of copying data between the nodes. MapReduce is a popular framework for running data-intensive jobs.**

**Availability of a specific runtime environment can not be guaranteed in a opportunistic environment. The application needs to be deployed dynamically. Virtual machines can be used to deploy a dynamic virtual cluster to run data-intensive jobs. Provisioning of virtual machine cluster needs to take account the opportunistic properties of the cluster. The proposed provisioning system must be able to decide the allowable systems based on user request, input data and number of tasks. It also should be able to add new node to the virtual cluster. The proposed system should be fault-tolerant to the virtual machine loss and provide persistent and reliable data storage.**

*Key words*— **Virtualization, Data Intensive Jobs, Opportunistic Cluster, MapReduce, Hadoop**

## I. Introduction

IN organizations, most of their computers are under-utilized. This results in resource wastage. These systems can be used to create a opportunistic cluster[1]. The opportunistic batch scheduler harnesses the idle resources, which improves resource utilization.The systems participating in opportunistic cluster are not tightly coupled, any system can leave or join the cluster anytime.

In recent decade, the popularity of cloud computing has emerged the virtualization technology as a key mechanism to scale IT infrastructure. Servers can be sliced into smaller systems to increase the utilization with virtualization. As the workstations and desktop computers are getting powerful, virtualization can also be useful to better utilize these systems.

Opportunistic cluster can not guarantee that the job will find its required execution environment on the execution system. To support user specific jobs, the opportunistic scheduler should be able to dynamically deploy user needed runtime environment on the execution node.The virtualization technology can solve the problem. Using virtual machine provisioning techniques user required execution environment can be provided dynamically on user request.

Most scientific jobs required to process big data. Processing of big data needs distribution of data among the cluster nodes in small chunks and processing those chunks in parallel. The jobs are sent near the data to overcome the overhead of copying data between nodes. The scheduler must be data-aware to provide this data-locality. Most of the traditional opportunistic scheduler is designed for compute-intensive jobs and they do not support data-awareness. Map-reduce[2] is popular data-aware distributed and parallel framework for processing big data.

The proposed system provisions virtual machines on execution systems to create a virtual machine cluster of data-intensive application. The provisioning system have to take account the opportunistic behavior of the cluster. In opportunistic cluster availability of certain amount of resources is not guaranteed. So, the scheduler have to decide the number of allowable virtual machine not solely on user request but also on amount of input data and number of tasks. The scheduler should be able to return a minimum number of virtual machines to the user so user can start executing his job. When more system become available, more virtual machines can be added to the virtual machine cluster to aid the execution. The provisioning system must be able to provide fault tolerance when some virtual machines become unavailable. The virtual machine cluster should be able provide a reliable and persistent data storage.

The rest of the paper is organized as follows. Section II introduces and discusses Opportunistic cluster, virtualization and data-intensive Jobs. In data-intensive jobs a brief introduction to MapReduce and Hadoop has given. Section III describes some of the related work done on running Hadoop on virtual machine and on opportunistic cluster. Section IV discusses the proposed provisioning technique used to create Hadoop virtual cluster in opportunistic cluster. Section V concludes this paper.

## II. Background

### A. Opportunistic Cluster

Organizations like Universities and IT farms are equipped with hundreds of workstations and desktops equipped with multi-core CPU and GBs of RAM connected together with gigabit ethernet. But these powerful systems are usually under utilized as the student or the worker can only use a fraction of the computing power provided by the systems. These computers can be used to create an opportunistic cluster. Opportunistic cluster provides a batch scheduler. Users submit their jobs to the scheduler and scheduler runs these jobs on idle resources of the cluster. In opportunistic cluster, two

[1]Computer Architecture & Operating Systems Department, Autonomous University of Barcelona, e-mail: {`Arindam.Choudhury,elisa.heymann,miquelangel.senar`}`@uab.es`.

kind of users can be observed for a system. One is the local users and another is the remote users who want to harness the idle resources and run their jobs. The opportunistic cluster balances the load between these users. One way of doing the balancing is to just simply stop the remote job execution when the scheduler encounter local user activity. Another way is to partition the system and allocate certain amount of resources for the scheduler. This partitioning can be done using control groups[3].

Condor[4] is a specialized job and resource management system (RMS) for compute intensive jobs. Condor uses opportunistic scheduling. Condor chooses idle system to submit the jobs. This opportunistic scheduling support dynamicity of the cluster where any node can join the cluster at any time. Condor provides job management mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their jobs to Condor, and Condor subsequently chooses when and where to run them based upon a policy, monitors their progress, and ultimately informs the user upon completion.

A Condor pool consists of one central manager and multiple submit and execution nodes. Central manager is responsible for collecting information, matching available resources with resource requests. The matching of job to resource is done using ClassAds matchmaking. Every machine in Condor pool advertises their attributes to the central manager. Jobs also have their own ClassAds which declare their requirement. Condor plays the role of matchmaker by continuously reading all job ClassAds and all machine ClassAds, matching and ranking job ads with machine ads. Condor ensures that the requirements in both ClassAds are satisfied.

### *B. Virtualization*

Virtualization[5][6] is changing the way organizations manage and deploy their infrastructure. Servers are virtualized to be sliced into smaller individual virtual machines to deliver flexibility, scalability and efficient utilization of resources. Today's desktops exhibit the CPU and memory capability which were once reserved for mainframe and high end UNIX systems. The workstations or desktops used in organizations are mostly IA-32 architecture systems. Using the virtualization technique, the host system can be splitted into several guest systems with the help of a hypervisor or Virtual Machine Monitor(VMM)[7]. These guest systems run independent of the host system. The guest system can have different processor architecture and different guest operating system. In traditional system where the operating system run at most privileged level, but in virtualized system the hypervisor needs to be run in high privileged mode. In a opportunistic cluster, the virtual machines can also be used to provide custom execution environment on the remote execution systems. Systems can be virtualized using three different techniques:

#### B.1 Full Virtualization

In full virtualization model, the guest operating system is unaware of the virtualization environment. The underlying hardware is fully emulated. The guest contains no knowledge about the host operating system. The hypervisor handles all the guest issued privileged instructions. This virtualization technique suffer from high virtualization overhead.

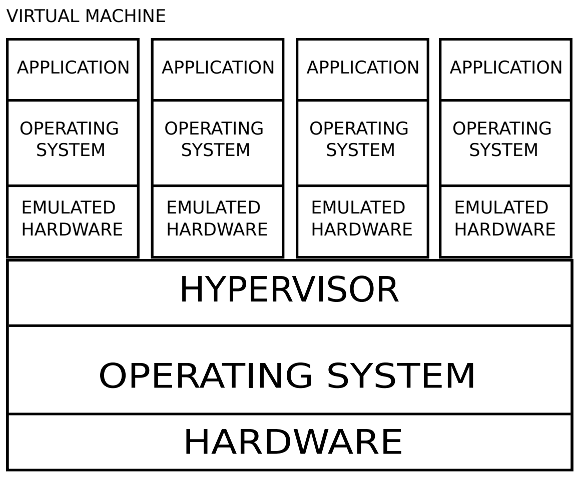

Fig. 1. Full Virtualization.

#### B.2 Paravirtualization

The guest operating systems are virtual environment aware. The privileged instructions are redirected to the hypervisor. This cooperation between hypervisor and the guest operating system results in improved performance. This approach needs input from the operating system vendors and starting with kernel 2.6.23, it is included in Linux mainline kernel. The hypervisor takes care of CPU and memory virtualization, power management and virtual machine scheduling. The host operating system is also managed by the hypervisor. The host operating system is modified to make it virtualization ready. This modifications are incorporated in kernel 3. The host operating system has direct access to devices. It provides device drivers and I/O management for virtual machines. Hypervisor handles all the CPU and memory access, but the I/O request are redirected to the host operating system.

Xen is a popular para-virtualization hypervisor. Xen has a modified host operating system, named Domain-0, which contains all device drivers. The guest operating systems are also modified and they are called Domain-U. Domain-U contains virtual driver interface. All I/O requests of Domain-Us are redirected to Domain-0 by the hypervisor.

#### B.3 Hardware Assisted Virtualization

The Intel VT-X and AMD-V extensions to the IA-32 architecture simplified the CPU virtualization. The CPU operates in host mode or guest mode. While in guest mode, the CPU traps the privileged instructions and returns control to the hypervisor. Virtualized memory management unit

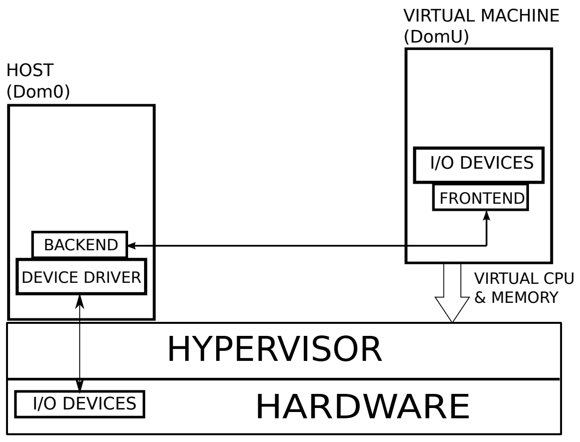

Fig. 2. Paravirtualization with Xen.

(MMU) is provided in hardware with the inclusion of Rapid Virtualization Indexing(RVI) for AMD and Extended Page Table(EPT) for Intel in the processors. New features are continuously added by both Intel and AMD to improve the performance of the virtualization. These features will improve the virtual machine performance without significant modification of the host or guest operating system.

Xen also supports Hardware Assisted Virtualization.

Kernel Virtual Machine (KVM) relies on hardware virtualization technologies. It is fully integrated with Linux kernel. The virtual machines run as a regular Linux process. I/O requests are provided by emulation of real device. KVM can provide paravirtualization of I/O devices using `virtio`. The main advantage of KVM is that it can take advantage of capability of Linux kernel. Where for each new improvement, Xen have to be separately updated.

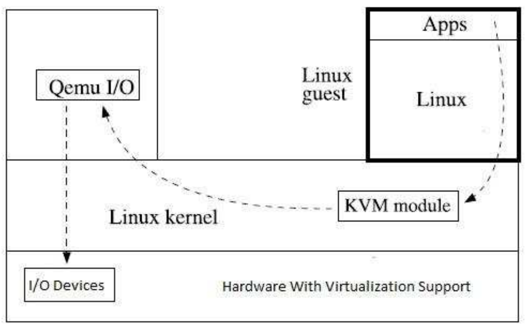

Fig. 3. Hardware Assisted Virtualization with KVM.

As the systems used in opportunistic cluster are heterogeneous, the provisioning system have to deal with different kind of hypervisor.

## C. Data-intensive Jobs

Opportunistic schedulers are designed for handling compute intensive jobs. As the input data for compute intensive jobs are small, these jobs can be scheduled on any idle system and then the input data can be fetched for processing. But when input data is big, as for data intensive jobs, data locality becomes important. The data should be available on the computing node to overcome the overhead of data transferring. The job scheduler have to be data-aware to achieve this criteria. The efficient way to analyze big data is to distribute it in small chunks and process those chunk in parallel[8]. So, a data-intensive job scheduler does not only schedule the jobs but also manage, access and distribute the input data.

MapReduce[2] is a popular framework for data-intensive computing. This model contains two functions Map and Reduce. These functions are written by the user. MapReduce works with key-value pairs.

- The Map function gets a pair of key and value for processing and generates modified intermediate key-value pairs.

$$\text{Map (key1, value1)} \Rightarrow \text{list (key2, value2)}$$

- The reduce function merges all intermediate values associated with the same intermediate key.

$$\text{Reduce (key2, list (value2))} \Rightarrow \text{list (value2)}$$

The MapReduce model allows programmers to easily design parallel and distributed applications, simply by writing Map and Reduce components, while the MapReduce run time is responsible for parallelization, concurrency control and fault tolerance.

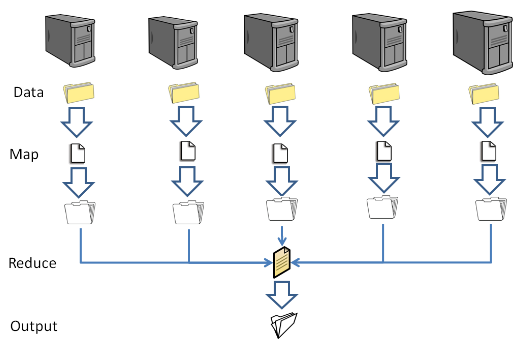

Fig. 4. MapReduce Job and Data Flow.

Hadoop[9] is an open source implementation of MapReduce based on Java. Hadoop also includes Hadoop Distributed File System (HDFS) to provide high throughput access to application data. HDFS is based on master-worker architecture. The master is called namenode. It splits the data and distributes it to the slave node, which are called datanode. Namenode stores the metadata about stored files. Datanodes actually store the data. Splits are replicated to assure reliability.

Hadoop MapReduce runs on top of HDFS and also based upon master-worker architecture. The master is called jobtracker and the slaves are called tasktrackers. The jobtracker queries the namenode for the split locations, considering the information retrieved from the namenode, jobtracker schedules the tasks on slaves. It also monitors the success and failures of the tasks.

Users submit jobs consisting of a map function and a reduce function. Hadoop breaks each job into multiple tasks. First, map tasks process each split of input and produce intermediate results, which are key-value pairs. Next, reduce tasks fetch the list of intermediate results associated with each key and run it through the user's reduce function, which produces the final output.

## III. Related Work

Amazon also introduces the Amazon Elastic MapReduce [10]. The Amazon Simple Storage Service (S3) is used to store the application script, the input data, log files and the output results. It supports HDFS. The Amazon Compute Cloud (EC2) is used to create the Hadoop cluster. Elastic MapReduce takes care of provisioning a Hadoop cluster, running the job flow, terminating the job flow, moving the data between Amazon EC2 and Amazon S3, and optimizing Hadoop. The details of configuring Hadoop, executing the job flow, managing the networking are hidden from the user.

The user launches AMI instances to create the Hadoop cluster. Amazon Elastic MapReduce starts instances in two security groups: one for the master node and another for the core node and task nodes.

Master node assigns Hadoop tasks to nodes and monitors their status. Core node is an EC2 instance that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node. Core nodes run both the datanodes and tasktracker Hadoop daemons. Task node is an EC2 instance that runs Hadoop map and reduce tasks, but does not store data. Task nodes are managed by the master node.

Amazon elastic MapReduce shows the possibility of using virtual machines to run MapReduce jobs. The virtual machine MapReduce cluster will be easier to manage as the cluster resource management daemons and MapReduce daemons are separated by the virtualization. The data locality will be provided by the cluster resource management system. The MapReduce daemons will be setup and run on the virtual machines.

In [11], the performance of running Hadoop jobs on virtual machines are evaluated using Xen hypervisor. It is pointed that virtual machines have overheads when running I/O intensive jobs as in paravirtualization the I/O requests have to be transferred to the host system for processing. In [12][13], interference-aware scheduling is proposed to use to schedule CPU between the virtual machines and the host operating system in Xen to obtain better performance in I/O intensive jobs.

Storing the big data in a opportunistic cluster is a problem, as the environment is volatile. When in a dedicated Hadoop cluster a chunk is replicated three times to get a higher availability, a chunk have to be replicated eleven times to get high availability in a opportunistic cluster with 0.4 unavailability rate[14].

In [14], proposed MapReduce On Opportunistic eNvironments (MOON) system contains a small number of dedicated resources. These dedicated resources help to obtain the availability of input data on the cluster with much smaller replication factor. The dedicated resources works as a backup storage and chunks are replicated just once on these nodes. The input data are replicated on non-dedicated system using a higher replication factor to gain data-locality. MOON also proposes two-phase task replication depending on the total job progress to ensure sufficient progress with high node volatility. The dedicated resources can also be used to run the task if available.

## IV. Virtual Machine Provisioning

The opportunistic cluster imposes several challenges in provisioning virtual machine cluster. As the cluster will be used to run data-intensive application, the cluster must be able to provide a consistent and reliable storage for input data as well as a reliable parallel platform to analyze the data.

### *A. Provisioning Virtual Machine Cluster*

When the user requested application is distributed parallel application, the virtual machine provision gets complex as the virtual machines need to work together to create a virtual cluster to be able to execute the user job properly. So, the virtual machines needs proper network configuration.

The networking can be configured in two ways:

- Virtual LAN:
  The virtual machines can be connected using Virtual LAN. The virtual machines and the clusters will reside on different network. This can be done using Open vSwitch[15].
- Bridged networking:
  The virtual machine can obtain IP address belong to the cluster sub network using bridged networking. Virtual Machines configured this way can send and receive data from any other system of the cluster.

While the Virtual LAN provides isolation, the bridged networking is easy to manage and configure.

In opportunistic cluster, when a user requests virtual machines, available systems in the cluster might be less than the user request. In this kind of situation, the provisioning system have to decide to provide the user the available systems or wait for more systems to become available. The decision depends on the type of user application. As for Hadoop, the user can start with a small cluster and when systems become available get more virtual machine to scale the cluster.

On the submit system, when user submits a Hadoop job, the submission process needs to decide the number of allowable virtual machines. Submission process needs to inquire central manager to obtain number of available nodes and decides the allowable nodes depending on user request, input data size and number of map and reduce tasks. The submis-

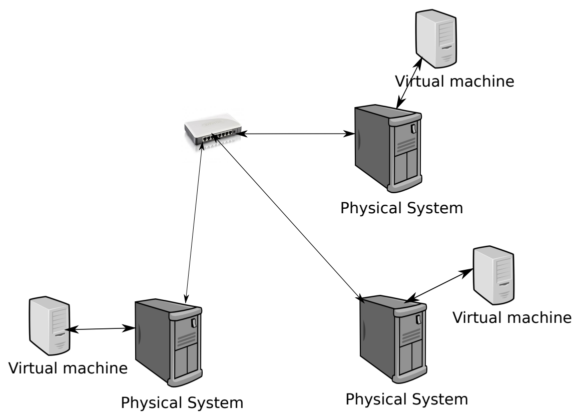

Fig. 5. Hadoop Virtual Cluster

sion process will then create the network and Hadoop configuration. The process will generate separate job for each alloted system. This job will run on execution system to configure and power on the the virtual machine. When powered on it will configure Hadoop and start the Hadoop daemons.

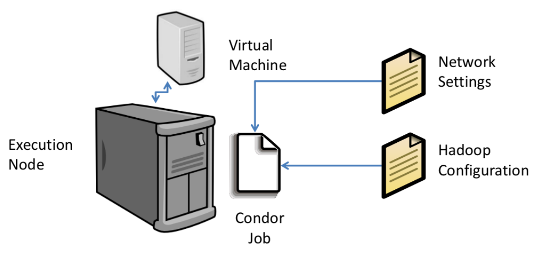

Fig. 6. Hadoop Virtual Cluster

The provisioning system should be able to interact with the data-intensive application to provide job management and monitoring. The proposed system should be able to transfer the input data to the virtual machine cluster and submit the jobs to the data-intensive application. In case of Hadoop, input data need to be transferred to the virtual machine Hadoop cluster and uploaded into the HDFS. This step can take significant time if the input data is large. Once the data is transferred, the Hadoop job can be launched.

### B. *Fault-tolerance*

As in our environment any system can leave the cluster at anytime, virtual machines can disappear with the input data. If input data is lost, the running job will fail. As stated in [14], the input data have to be replicated with a higher factor. Also, the task running on the lost virtual machine have to be rescheduled. if the task is a reduce task then all the intermediate data created by the map tasks are also lost. So, the map tasks have to be rescheduled to generate the intermediate data before scheduling the reduce task.

### C. *Scaling the Virtual Machine Cluster*

In opportunistic cluster, new system may become available and can be used to scale the virtual machine cluster for efficiency. It is easy to add a new node in Hadoop cluster. But HDFS but not move data chunks to new nodes automatically. The cluster need to be re-balanced to transfer data chunks to the newly added block.

The provisioning system keeps track of the execution of data-intensive job and if needed add new system to the virtual cluster. The decision of adding new virtual machine depends upon if the remaining task needs more nodes.

## V. Conclusion

The virtualization technology is used to provide Hadoop virtual cluster to run data-intensive job in opportunistic cluster. Virtualization suffers penalty from I/O intensive applications. But as in opportunistic cluster, the harnessed resources are free to the organization, this small penalty can be ignored in comparisons to the benefits of having dynamic virtual environment to run specialized jobs. The virtualization will aid in harnessing the idle resources as well. Also, as the virtualization technology is getting input from both the software and hardware development, the virtualization will give more performance in the future.

The provisioning of virtual machine in opportunistic cluster needs more decision making as it is very volatile. When granting virtual machine, if the available system is less than the user need, the allowable should be decided by evaluating user need against the available system, size of input data and number of map and reduce jobs. While granting more system to a exiting virtual cluster, the remaining jobs need to be evaluated with exiting systems in the virtual cluster to decide wheather adding new system is beneficial. To obtain fault-tolerance from system loss, the replication factor should be higher so that system loss does not result in input data loss.

## VI. Acknowledgement

This research has been supported by the TIN2011-28689-C02-01.

## Referencias

[1] D. Nichols, “Using idle workstations in a shared computing environment,” *SIGOPS Oper. Syst. Rev.*, vol. 21, no. 5, pp. 5–12, Nov. 1987.
[2] Jeffrey Dean and Sanjay Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[3] Take advantage of cgroups-based capabilities on new Linux platforms, “https://condor-wiki.cs.wisc.edu/index.cgi/tktview?tn=1831,” .
[4] Douglas Thain, Todd Tannenbaum, and Miron Livny, “Distributed computing in practice: the condor experience.,” *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.
[5] Robert P. Goldberg, “Survey of Virtual Machine Research,” *Computer*, pp. 34–45, 1974.
[6] James E. Smith and Ravi Nair, “The architecture of virtual machines,” *Computer*, vol. 38, no. 5, pp. 32–38, May 2005.

[7] Mendel Rosenblum and Tal Garfinkel, “Virtual machine monitors: Current technology and future trends,” *Computer*, vol. 38, no. 5, pp. 39–47, May 2005.

[8] Jiubin Ju, Gaochao Xu, and Jie Tao, “Parallel computing using idle workstations,” *SIGOPS Oper. Syst. Rev.*, vol. 27, no. 3, pp. 87–96, July 1993.

[9] Apache Hadoop, “http://hadoop.apache.org/,” .

[10] Amazon Web Services, “http://aws.amazon.com/,” .

[11] Shadi Ibrahim, Hai Jin, Lu Lu, Li Qi, Song Wu, and Xuanhua Shi, “Evaluating mapreduce on virtual machines: The hadoop case,” in *Cloud Computing*, Martin Jaatun, Gansen Zhao, and Chunming Rong, Eds., vol. 5931 of *Lecture Notes in Computer Science*, pp. 519–528. Springer Berlin / Heidelberg, 2009.

[12] Ron C. Chiang and H. Howie Huang, “Tracon: interference-aware scheduling for data-intensive applications in virtualized environments,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2011, SC ’11, pp. 47:1–47:12, ACM.

[13] Jun Fang, Shoubao Yang, Wenyu Zhou, and Hu Song, “Evaluating i/o scheduler in virtual machines for mapreduce application,” in *Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing*, Washington, DC, USA, 2010, GCC ’10, pp. 64–69, IEEE Computer Society.

[14] Heshan Lin, Xiaosong Ma, Jeremy Archuleta, Wu-chun Feng, Mark Gardner, and Zhe Zhang, “Moon: Mapreduce on opportunistic environments,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, New York, NY, USA, 2010, HPDC ’10, pp. 95–106, ACM.

[15] Open vSwitch, “http://openvswitch.org/,” .