Middle-East Journal of Scientific Research 23 (9): 2081-2088, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.09.22427

Cloud Opentsack Map Reduce Fastvideo Transcoding as a Service with Qos

[1]D. Kesavaraja and [2]A. Shenbagavalli

[1]Department of Computer Science and Engineering,
Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, India
[2]Department of Electronics and Communication Engineering,
National Engineering College, Kovilpatti, India

Abstract: Cloud computing and big data are changing thetodays modern on demand video service. This paper describes how to increase the speed of video transcoding in aopenstackprivate cloud environment using HadoopMap Reduce. In this paper OpenStack Juno is used to build the private cloud Infrastructure as a service having map code executing on the node where the video transcoding resides significantly reduces this problem. This practice, called "video locality", is one of the key advantages of Hadoop Map/Reduce. This scheme highly describes the relationship of a Hadoop Map Reduce algorithm and video transcoding in the experiment. As a result of map reduce video transcoding experiment in openstack Juno the performance outperforms the performance of physical server when running on the virtual machine in the private cloud based on the metrics Time Complexity and Quality of Service (QoS) Check using PSNR.

Key words: Cloud Computing · Video Transcoding · Openstack · Hadoop · Map Reduce

INTRODUCTION

The virtualization tools have been used as cloud technologies to increasethe accessibility of high performance hardware computing resources in the cloud computing environment [1,2]. In addition, Hadoop Map Reduce software design model emerged as the distributed and parallel processing expertise in order to handle data effectively [3,4,5]. In [6], the author describes, the correlation between the application attributes and the video transcoding through the performance analysis of Mapreduce Hadoop application for thedistributed and parallel process that are conducted in the virtualized cluster environment. For this, it is to generate the virtual machine instance after configuring private cloud using Open Stack [7]. It is to analyze the results after conducting Hadoop application in the virtual machine instances.

Hadoop map code must have the ability to read video "locally". Some popular solutions, such as networked storage (networked-attached storage [NAS] and storage-area networks [SANs]) will always cause network traffic, so in one sense you might not consider this "local", but really, it's a sense of perspective [8, 9, 10]. Depending on the situation, it might define "local" as meaning "within a single datacenter" or "all on one rack". Hadoop must be aware of the topology of the nodes where tasks are executed. Tasktracker nodes are used to execute map tasks and so the Hadoop scheduler needs information about node topology for proper task assignment.HDFS supports data locality out of the box, while other drivers (for example, Openstack Swift) need to be extended in order to provide data topology information to Hadoop [11, 12].

Hadoop uses a three-layer network topology. These layers were originally specified as Data-Center, Rack and Node, though the cross-Data-Center case isn't common and that layer is often used for defining top-level switches [13].

In Figure 1, this topology works well for traditional hadoop cluster deployments, but it is hard to map a virtual environment into these three layers because there is no room for the hypervisor. Under certain circumstances, two virtual machines that are running on the same host can communicate much faster than they could on separate hosts, because there's no network involved.

Corresponding Author: D. Kesavaraja, Department of Computer Science and Engineering,
Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, India.

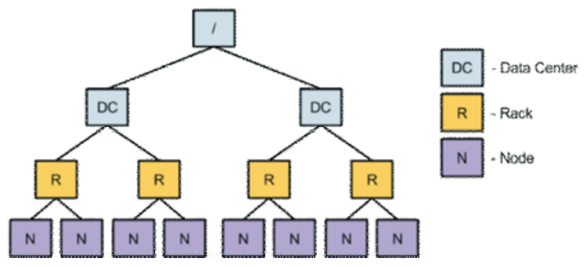

Fig. 1: Hadoop Network Topology

Fig. 2: Openstack Juno

Hadoop is a open source software based middleware on the basis of MapReduce [14]. Hadoop is broadly composed of the two components of HDFS that is the distributed file system and MapReduce that is the parallel programming model. MapReduce application of Hadoop is composed of Job that is the unit of task conducted by clients. Job is composed of input data, MapReduce program and setting information. In addition, Job is executed as being divided into Map task and Reduce task.

Openstack Juno: The successor to the Icehouse release of the OpenStack open source cloud computing platform, OpenStack Juno debuted on October 16, 2014 as the tenth release of OpenStack. Among OpenStack Juno's most prominent additions are numerous new networking capabilities, including a Distributed Virtual Router (DVR), enhanced IPv6 support in the OpenStack Neutron networking project, improved Network Functions Virtualization (NFV) support, better visibility into network information and better support for multiple networks for the OpenStack Nova project [15].

OpenStack Juno, the tenth release of the open source software for building public, private and hybrid clouds has 342 new features to support software development, big data analysis and application infrastructure at scale. The OpenStack community continues to attract the best developers and experts in their disciplines with 1,419 individuals employed by more than 133 organizations contributing to the Juno release [16].

In Figure 2, OpenStack is an open and scalable operating system for building public and private clouds. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services.

Openstack has 7 Major components, they are

- OpenStack Compute (Nova)
- OpenStack Image service (Glance)
- OpenStack Networking (Quantum)
- OpenStack Object Storage (Swift)
- OpenStack Block Storage (Cinder)
- Identity_FKeystone
- Dashboard _FHorizon

In Figure 3 describesin the top Openstack to create a hadoop map reduce environment for performing fast video transcoding technique.

In Figure 4 describes all the major components of openstack Infrastructure as a service. These components are bound together to deliver an environment that allows for the dynamic provisioning of compute and storage resources. From a hardware standpoint, these services are spread out over many virtual and physical servers. As an example, most organizations deploy one physical server to act as a controller node and another to serve as a compute node. Many organizations choose to parse out their storage environment onto a dedicated physical server, as well, which in the case of an OpenStack deployment would mean a separate server for the Swift storage environment [17].

For those organizations looking for an advanced level of flexibility, scalability and autonomy within their big data environment, they can leverage the native abilities of the open source offerings provided by Apache and OpenStack [18,19].

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used [20, 21, 22].

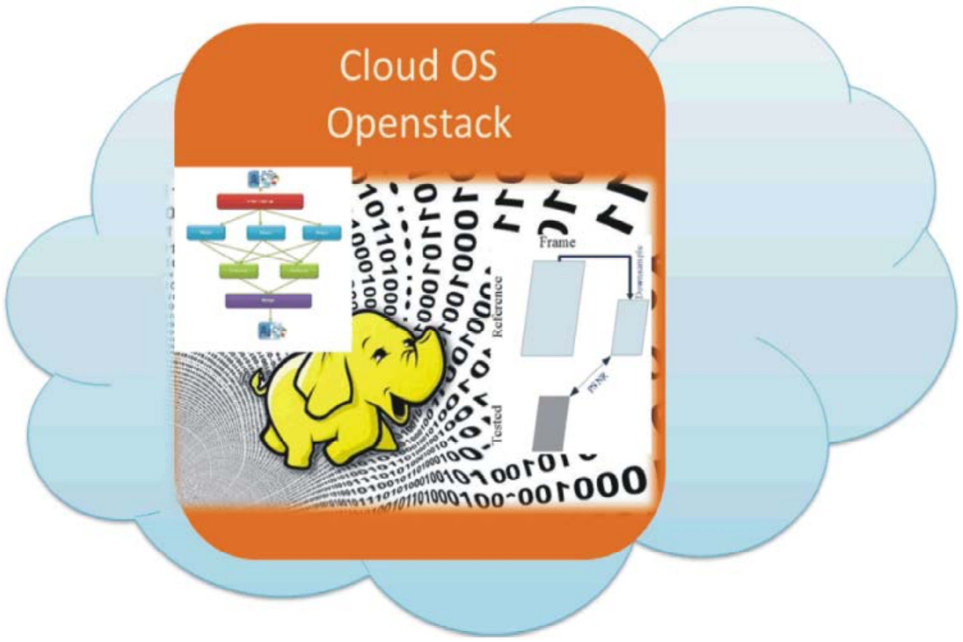

Fig. 3: Openstack Cloud Operating System

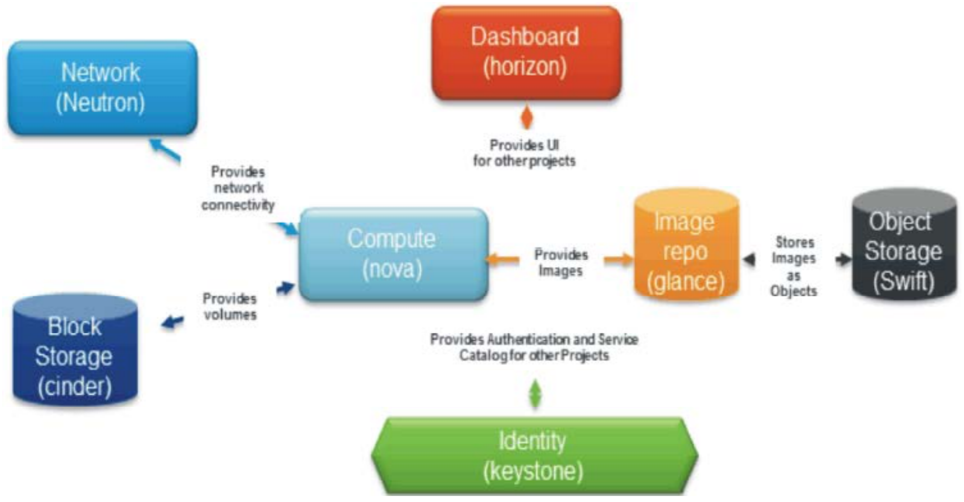

Fig. 4: Openstack Components

Table 1: Openstack Components

COMPONENTS	OPERATIONS
1. OpenStack Compute (Nova)	Compute resource management and Scheduler and VM life cycle management and VNC proxy
2. OpenStack Image service (Glance)	Discovering, registering and retrieving VM images
3. OpenStack Networking (Neutron)	Discovering, registering and retrieving VM images
4. OpenStack Object Storage (Swift)	Object Storage (ex. Amazon S3)
5. OpenStack Block Storage (Cinder)	Provides persistent block storage to VM
6. Identity_FKeystone	User Identity and Components need register to keystone
7. Dashboard _FHorizon	Web dashboard_B(ex. user login, VM create and terminate, volume create, security group and etc.)

HadoopMap Reduce Functionalities: MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems and use more heterogenous hardware) [23, 24]. Processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing it on or near the storage assets in order to reduce the distance over which it must be transmitted [25, 26, 27].

- "Map" step: Each worker node applies the "map()" function to the local data and writes the output to a temporary storage. A master node orchestrates that for redundant copies of input data, only one is processed.
- "Shuffle" step: Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node.
- "Reduce" step: Worker nodes now process each group of output data, per key, in parallel [28].

In Figure 5, MapReduce is as a 4-step parallel and distributed computation:

- Input Video Data – the "MapReduce system" designates Map processors, assigns the input key value *K1* that each processor would work on and provides that processor with all the input video data associated with that key value.
- Map() code – Map() is run exactly once for each *K1* key value, generating output organized by key values *K2*.and the MapReduce system designates Reduce processors, assigns the *K2* key value each processor should work on and provides that processor with all the Map-generated data associated with that key value.
- Reduce() code – Reduce() is run exactly once for each *K2* key value produced by the Map step.
- Output Video Data – the MapReduce system collects all the Reduce output and sorts it by *K2* to produce the final outcome.

These four steps can be logically thought of as running in sequence – each step starts only after the previous step is completed – although in practice they can be interleaved as long as the final result is not affected.

Proposed Video Transcoding Mechanism: In the proposed video transcoding technique uses JSVM and deployed in HadoopMapreduce in the top of openstack platform. The JSVM (Joint Scalable Video Model) software is the reference software for the Scalable Video Coding (SVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG) [25].

The operations are

- Userresquest a video from cloud server
- Cloud Server Recognizes the needed Frame Rate, Resolution and Bitrate based on Device capasity such as Screen height, width, resolution, CPU Power, Battery Energy
- Open Stack Private Cloud Decision Maker makes the Decision of required Frame Rate, Resolution and Bitrate
- Creates the New Profile for Required Video Format
- After Profile Creation Call Prominent Matching Video Transcoding Protocol
- Provide the Service

The proposed video transcoding uses a novel Prominent Matching Video Transcoding Protocol. The Pseudo of Prominent Matching Video Transcoding Protocol is,

```
Void Prominent_ Matching_ Video_ Transcoding_ Protocol()
{
    ReadVideo(profile,videoName)
    VideoSpilit()
    Map()        //Distributed JSVM Transcoding
    Reduce()     //Combine JSVM Transcoding
    VideoMerge()
    StoreVideo(newVideoName)
    }
    function map(Chunck s, Blob video):
    begin
    for each stream s in video:
    encode (s, 1)
    end
    function reduce(Chunck s, Iterator partialCounts):
    begin
    for each s in partialCounts:
    merge(s,video)
    end
```

In this protocol used to perform fastest video transcoding technique.

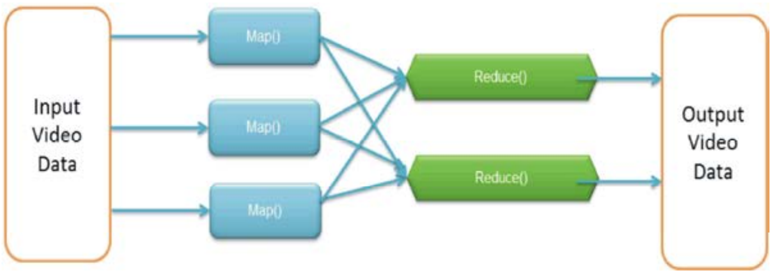

Fig. 5: Hadoop Functions

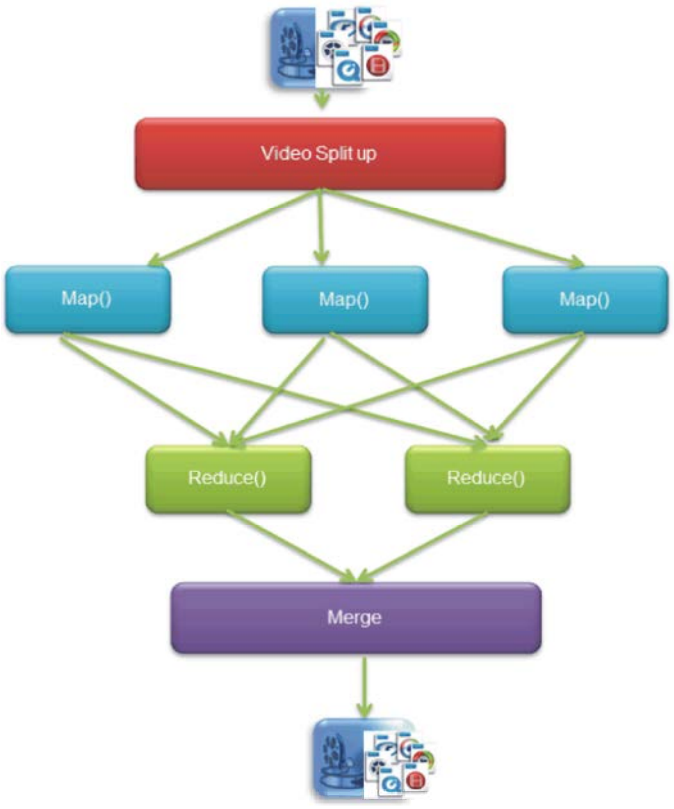

Fig. 6: Proposed Video Transcoding

MapReduce achieves reliability by parceling out a number of operations on the set of data to each node in the network. Each node is expected to report back periodically with completed work and status updates. If a node falls silent for longer than that interval, the master node records the node as dead and sends out the node's assigned work to other nodes. Individual operations use atomic operations for naming file outputs as a check to ensure that there are not parallel conflicting threads running. When files are renamed, it is possible to also copy them to another name in addition to the name of the task.

In Figure 6, describes the step by step functions of fastest map reduce in openstack. In video transcoding to continue to function well as it (or its context) is changed in size or volume in order to meet a user need. So transcoding in Mparecuce is scalable. Time complexity is commonly estimated by counting the number of elementary operations performed by the mpa reduce algorithm, where an elementary operation takes a fixed amount of time to perform.

Testbed And Benchmarks: In Figure 7 the testbed of video transcoding. In thistestbed comprises a cluster of two machines that run Hadoop and Swift with one master that runs the HadoopJobTracker and NameNode and the Swift proxy node and two workers that run the HadoopTaskTrackers and DataNodes and the Swift storage nodes. We turned off replication to ensure that mappers for the same data partition are always launched

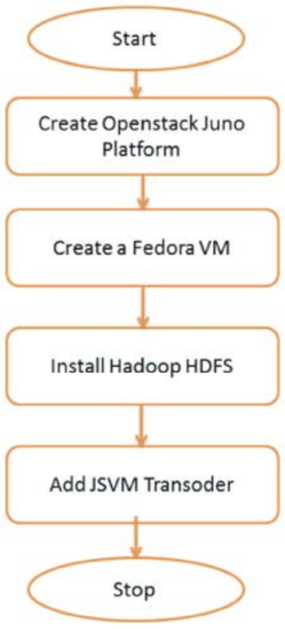

Fig. 7: Testbed of Video Transcoding

on the same machine for HDFS and Swift. The micro benchmarks compare the execution time of file system operations such as ls or open on HDFS and Swift, executed via the Hadoop API. The macro benchmark measures the completion time of MapReduce jobs on both storage systems.

Video is distributed across the workers in the same way for HDFS and Swift. Hadoop accesses data from the underlying storage system by providing a FileSystem interface. To analyze the differences in job completion time in more detail when changing the interface implementation, we instrumented the code to measure the execution times of calls to FileSystem methods. This allows us to identify the specific methods that take longer/are faster.

Experimental Result: The performance of the proposed openstackhadoop merged method is tested by using the Time Complexity and PSNR metric and compared with traditional Physical server methods and individual Hadoop scheme.

Time Complexity: In Figure 8, The new open stack hadoop method is proposed based on transcoding time estimation with transcoding servers' information, movie information and target transcoding bit-rate. In experiments, the proposed method produces the best performance scalability according to the increase of transcoding cloud servers.

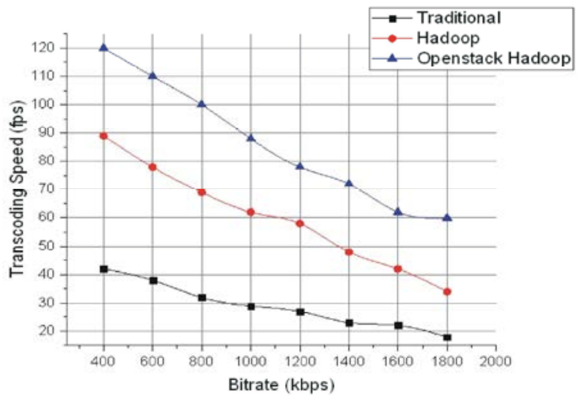

Fig. 8: Time Complexity Measurement

Peak Signal-to-Noise Ratio: The performance measure is the Peak Signal to Noise Ratio (PSNR). It is one of the parameters that can be used to quantify image quality. Compression schemes can be lossy or lossless. Lossless schemes preserve the original data. Lossy schemes do not preserve all the original data; so one cannot recover lost picture information after compression. Lossy schemes attempt to remove picture information the viewer will not notice. PSNR is usually expressed in terms of the logarithmicdecibel scale.The PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). It is most easily defined via the mean squared error (MSE) which for two m×n monochrome images I and K where one of the images is considered a noisy approximation of the other and is defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[I(i,j) - K(i,j)\right]^2 \qquad (1)$$

The PSNR is defined as:

$$PSNR = 10.\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20.\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \qquad (2)$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. For color images with three RGB values per pixel, the definition of *PSNR* is the same except the *MSE*, which is the sum over all squared value differences divided by image size and by three. Typical values for the *PSNR* in lossy image and video compression are between 30 and 50 dB where higher is better. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB. When the two images are identical, the *MSE* will be zero. For this value the *PSNR* is undefined.

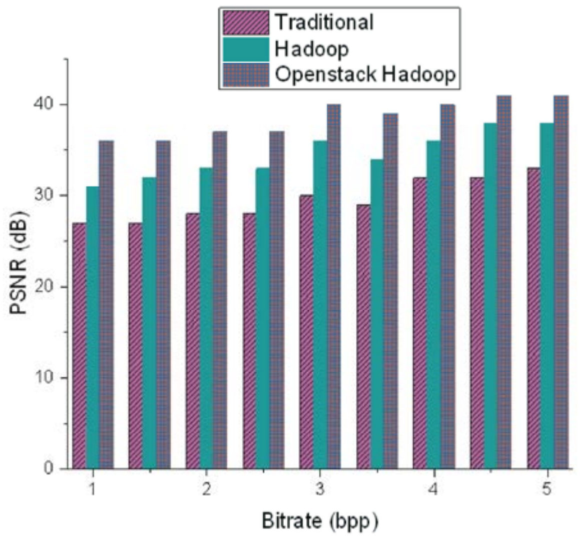

Fig. 9: PSNR Comparision

In Figure 9, transcoding there is a relationship between bit rate or coding rate (compression ratio) i.e. bits per pixel (bpp) and distortion. Distortion measurement is used to measure how much information has been lost when a reconstructed version of a digital image is produced from compressed data and it is usually measured as PSNR. The OpenstackHadoop Quality is little bit out perform the traditional methods.

CONCLUSION

Data processing on Hadoop in virtual environment is, perhaps, the next step in the evolution of Big Data. As clusters grow, it is extremely important to optimize consumed resources. Technologies like data locality can drastically decrease network use and allow you to work with large distributed clusters without losing the advantages of smaller, more local clusters. This gives you the opportunity for nearly infinite scaling on a Hadoop cluster.This paper compared the physical cluster and the performance after conducting the performance evaluation after configuring a variety of virtual machine instances in the configured cloud after configuring the private cloud using OpenStack. Hadoop video transcoding application could obtain the physical server performance of more than traditional methods through this experiment in the case of performing by utilizing the virtual machine. As a result, it was possible through the experiment to find out that the compute intensive application was the element causing a significant change for the performance at both physical environment and virtualized environment; thus, it was the number of cores and the available memory. In the future, we plan to increase the scale of our setup and look at different workloads, the impact of replication and the implications of object size to data locality

REFERENCES

1. Armbrust, M., A. Fox and R. Griggith, 2009. "Above the cloud: A Berkeley View of Cloud Computing," Technical Report No.UCB/EECS-2009-28, EECS Department, University of California at Berkeley, USA.
2. Yunhee, Kang and Geoffrey C. Fox, 2011. Performance Evaluation of MapReduce Applications on Cloud Computing Environment, FutureGrid Grid and Distributed Computing: International Conferences, GDC.
3. Dean, J. and S. Ghemawat, 2010. "MapReduce: A Flexible Data Processing Tool," Communications of the ACM, 53: 72-77.
4. Ekanayake, J., 2008. "MapReduce for Data Intensive Scientific Analyses," the 2008 Fourth IEEE International Conference one Science.
5. Hadoop. http://hadoop.apache.org/
6. Yunhee, Kang and Kyung-Woo Kang, 2013. “AnEmpirical Study of Hadoop Application running on Private Cloud Environment”, Advanced Science and Technology Letters Vol.35(Cloud and Super Computing), pp: 70-73.
7. OpenStack. http://www.openstack.org/
8. Steven C. Markey, 2012. "Deploy an OpenStack private cloud to a HadoopMapReduce environment, IBM Developers Works.
9. (2014). Improving Data Processing Performance with Hadoop Data Locality [Onine] Available: https://www.mirantis.com/blog/improving-data-processing-performance-hadoop-data-locality/.
10. Andrew Lazarev, 2014. "Performance of Hadoop on OpenStack", Mirantis [Onine] Available: https://www.openstack.org/assets/presentation-media/Performance-of-Hadoop-on-OpenStack.pdf.
11. De Candia, G., D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, 2007. “Dynamo: Amazon’s highly available key-value store,” in SOSP.
12. Beaver, D., S. Kumar, H.C. Li, J. Sobel and P. Vajgel, 2010. “Finding a needle in haystack: Facebook’s photo storage.” in OSDI.

13. Dean, J. and S. Ghemawat, 2004. “Mapreduce: Simplified data processing on large clusters,” in OSD
14. Hadoop. http://hadoop.apache.org/
15. http://www.webopedia.com/TERM/O/openstack-juno.html
16. https://www.openstack.org/software/juno/
17. “OpenStack Swift,” http://docs.openstack.org/developer/swift/.
18. “Apache HDFS,” http://hadoop.apache.org/docs/r1.2.1/hdfs design.html.
19. “Apache HadoopMapReduce,” http://hadoop.apache.org/.
20. “HadoopMapReduce - Swift Connector,” http://goo.gl/mJIT7Y.
21. “HiBench,” https://github.com/intel-hadoop/HiBench.
22. https://en.wikipedia.org/wiki/OpenStack
23. Mihailescu, M., G. Soundararajan and C. Amza, 2013. “Mixapart: decoupled analytics for shared storage systems.” in FAST.
24. Sehrish, S., G. Mackey, J. Wang and J. Bent, 2010. “Mrap: a novel map reduce based framework to support hpc analytics applications with access patterns,” in HPDC.
25. Dongmahn Seo, Jongwoo Kim and Inbum Jung, 2010. "Load Distribution Algorithm Based on Transcoding Time Estimation for Distributed Transcoding Servers," Information Science and Applications (ICISA), 2010 International Conference on, 1(8): 21-23.
26. Kesavaraja, D. and A. Shenbagavalli, 2013. "Cloud video as a Service [VaaS] with storage, streaming, security and Quality of service: Approaches and directions," Circuits, Power and Computing Technologies (ICCPCT), 2013 International Conference on, pp: 1093, 1098, 20-21
27. Kesavaraja, D. and A. Shenbagavalli, 2015. “Scalable Cloud Video Broadcasting as a Service with enhanced Quality Using Open Stack SWIFT”, International Journal of Applied Engineering Research (IJAER), 10(20Special Issues): 18916-18921
28. Lukas Rupprecht, Rui Zhang and Dean Hildebrand, “Big Data Analytics on Object Stores: A Performance Study”, The International Conference for High Performance Computing, Networking, Storage and Analysis.