Data Location Aware Scheduling for Virtual Hadoop Cluster Deployment on Private Cloud Computing Environment

Asmath Fahad Thaha, Anang Hudaya Muhamad Amin, Subarmaniam Kannan, Nazrul Muhaimin Ahmad
Thundercloud Research Lab
Faculty of Info. Science & Technology (FIST)
Multimedia University
Jalan Ayer Keroh Lama, Melaka 75450
Malaysia
Email: {asmath.fahad,anang.amin,subar.kannan,nazrul.muhaimin}@ mmu.edu.my

***Abstract*—With the advancements of Internet-of-Things (IoT) and Machine-to-Machine Communications (M2M), the ability to generate massive amount of streaming data from sensory devices in distributed environment is inevitable. A common practice nowadays is to process these data in a high-performance computing infrastructure, such as cloud. Cloud platform has the ability to deploy Hadoop ecosystem on virtual clusters. In cloud configuration with different geographical regions, virtual machines (VMs) that are part of virtual cluster are placed randomly. Prior to processing, data have to be transferred to the regional sites with VMs for data locality purposes. In this paper, a provisioning strategy with data-location aware deployment for virtual cluster will be proposed, as to localize and provision the cluster near to the storage. The proposed mechanism reduces the network distance between virtual cluster and storage, resulting in reduced job completion times.**

I. Introduction

In todays computing environment, virtualization technology is replacing traditional physical data processing platforms due to its high scalability and availability. Cloud computing provides an avenue for such technology to rapidly evolve by virtualizing processing and storage platforms either through services such as Platform-as-a-Service (PaaS) or Infrastructure-as-a-Service (IaaS). OpenStack is one of the examples of cloud computing platforms for managing large pools of compute and storage nodes, networking resources, and software stacks that spread throughout multiple data centers. It allows applications to be deployed more dynamically, and provides a basis for IaaS, PaaS and Software-as-a-Service (SaaS) implementations.

Hadoop is a distributed and scalable data processing framework that can be deployed across multiple computing clusters [12]. Hadoop enables data partitioning and computation on numerous hosts, and performing parallel computations close to the data [9]. Conventional Hadoop systems run on physical commodity hardware with local computing and storage capabilities. The distributed property of Hadoop however, makes it suitable for virtual Hadoop cluster provisioning in cloud computing platform such as OpenStack [2].

Although Hadoop virtual clusters are able to perform automation and higher utilization of shared infrastructure, the performance of Virtual Hadoop is still an open problem. In cloud implementations, the VMs in which Hadoop cluster is being deployed are attached to virtual storage volumes over the network. Although cloud provides numerous advantages, virtual Hadoop clusters introduce their own set of challenges [3], [8].

One of the major challenges is the lack of data locality in virtual Hadoop deployments on cloud [5]. A key disadvantage of running Hadoop on the cloud is such that it is not aware of the underlying cloud infrastructure. Hence, Hadoop reads and writes data to nodes not knowing the latency between them, resulting in increased job completion time. In order to improve the transfer time, a location aware virtual placement strategy for virtual Hadoop cluster deployment should be put into consideration.

The distributed nature of cloud deployments inhibits the possibility for common storage platform implementations. Distributed storage models, such as Network Attached Storage (NAS) or Storage Attached Networks (SAN) are not suitable for Hadoop deployment due to the additional network overhead imposed, violating the Hadoop data locality principal [13]. Nevertheless, running Hadoop on the cloud [1], [11] has some benefits that outweighs its drawbacks. Nguyen and Shi [4] implemented a distributed cache system in which the data is cached persistently in the compute cluster. The respective data is not removed during the VM termination to reduce data movement. Dynamic VM reconfiguration as proposed by Park et al. [6] performs adjustments the computing capability of VMs, in order to maximize resource utilization. Palanisamy et al. [5] analyzes the network flows between devices that store the input/intermediate data and those that process the data.

Due to lack of data locality in cloud implementation, network latency has been the main focus area of performance improvements in virtual Hadoop deployments [7]. It is vital to achieve high data locality in such environments due to the transfer of data over the WAN network connecting multiple geographically separated regions.

This paper presents an investigative work on the effects of

978-1-5090-0676-2/16/$31.00 ©2016 IEEE

data locality in virtual Hadoop setup on cloud. The performance analysis conducted identifies network latency as a key factor impacting the performance of virtual Hadoop clusters on the cloud. Network latency is analyzed with regards to lack of data locality and a mechanism to address such issue will be proposed.

This paper is structured as follows: Section II describes the data locality issue in cloud, specifically with regards to Hadoop cluster deployment. A discussion on location-aware scheduling to address data locality issue in virtual Hadoop deployment is presented in Section III. Section IV presents the proposed framework design for location-aware scheduling. Results of analysis is presented in Section V. Finally Section VI concludes the paper.

II. Data Locality Consideration

Data locality is a concept commonly being considered in large-scale distributed systems such as cloud, that aims to process data close to the storage location in order to reduce data movement between compute and storage facilities [10]. The overall concept of data locality is depicted in Fig. 1.

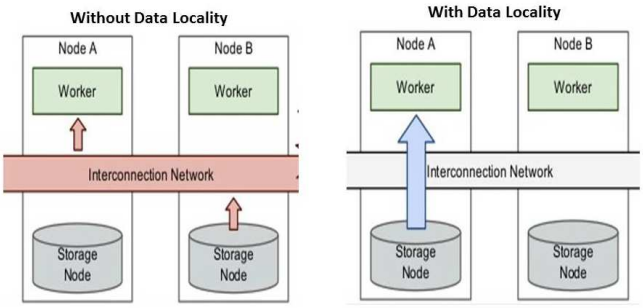

Fig. 1. Worker node with data locality and without data locality.

In order to consider data locality, files are broken into small chunks called blocks in Hadoop Distributed Filesystem (HDFS). Each block is mapped to a task for data processing. During MapReduce session, Hadoop utilizes the data block location information to run the job directly on the node that host the data. Hadoop's data replication policy means it has a few hosts to select from.

VMs in virtual Hadoop cluster are randomly instantiated in compute nodes and are attached to persistent block storage devices over the network. With this configuration, data have to be moved across the network interconnects for processing. If the cloud infrastructure stretches across different regions, data has to be transferred over the internet to the region where the compute node resides. For large cluster, excessive network bandwidth congestion will be experienced, resulting in system performance deterioration.

In the following subsection, we present a case study on a distributed private cloud architecture.

A. Case Study: Distributed Private Cloud Architecture

The proposed solution is designed for virtual Hadoop implementations in distributed *OpenStack* private cloud architectures. In the context of this work, distribution refers to locations involving several geographical sites. For example: A business corporation with multiple branches where each has its own compute and storage facilities. A controller node in the main office controls these large pools of resources (compute, storage, network) across all the branches. The controller recognizes each branch as a separate entity in the cloud. Each region will execute all standard *nova* services, with exception of the *nova-api*. The top level region in which the controller resides (headquarters) will run the *nova-api* service.

Fig. 2 illustrates the assumed structure for the private cloud architecture. The corporate office has 3 regions namely *R1*, *R2* and *R3* and headquarters (*HQ*). Each region has its own compute and storage facilities. The controller in *HQ* controls all requests in and out of the cloud platform.

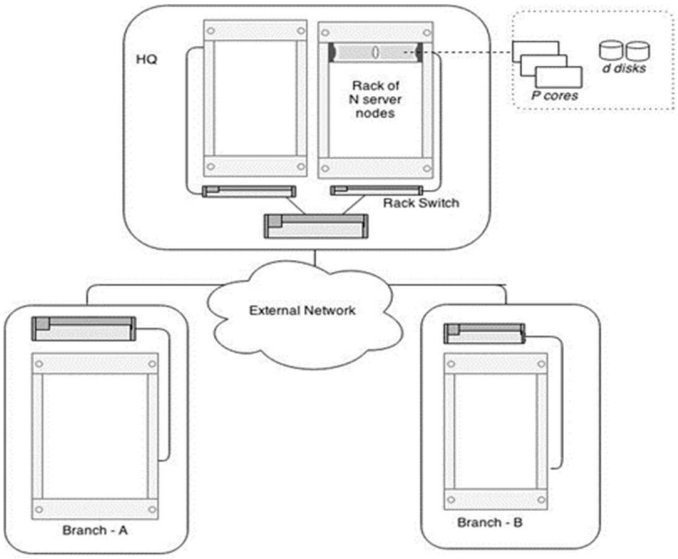

Fig. 2. Assumed structure for private distributed cloud architecture with three regions (including HQ).

Given that the server nodes are organized into R racks with N nodes each. Each node is then consists of p processors and d disk units, and all nodes are connected to a Rack Switch (*RS*). The uplinks of these switches are connected to a single cluster switch or router. This simple arrangement constitutes a total of $R \times N \times d$ disks with a minimal resource requirements in networking components. Each branch has its own dedicated storage. The following example shows a common use case scenario for data locality consideration.

If *Branch-A* requires analysis on their data, a request will be sent to *HQ* to instantiate a cluster. The *HQ* will create a cluster based on the resources available in the cloud. The VM placements of the cluster are based on availability. If the controller has placed the VMs in *Branch-B*, the data from *Branch-A* will be moved to *Branch-B*. Once the jobs are

978-1-5090-0676-2/16/$31.00 ©2016 IEEE

completed the results will be returned back to *Branch-A*. The amount of I/O bound operations will impose a significant load on both storage and network. An increasing amount of load is being put on the network bandwidth due to the lack of data locality consideration.

B. Impact of Data Locality towards Hadoop Deployment

Consider a physical Hadoop cluster with 15 nodes is launched to process over 100GB of input data. All the nodes in the cluster are configured as HDFS data nodes. HDFS is configured to use 128MB block size (800 maps created) with replication factor of 3. Once the mapreduce job is launched Hadoop attempts to schedule as many as data-local maps as possible to reduce latency.

Another virtual Hadoop cluster identical to the above mentioned configurations was launched in the cloud. All 15 virtual nodes in the cluster are configured as HDFS data nodes. As expected Hadoop will schedule as many data local maps as possible, but Hadoop is not aware of the fact that all disk volumes of the HDFS data nodes are located in different physical storage nodes and are attached to the virtual nodes via the network.

Since the Hadoop job scheduler does not have knowledge of physical nodes in the cloud, it will only guarantee that the map task is assigned to the virtual node, which is attached to the storage volumes, which has the data for the assigned map task. The Hadoop scheduler does not takes the latency between the virtual node and the attached storage volume into account when assigning map tasks.

On the other hand, *Openstack* has the physical architecture of all compute and storage nodes, yet when virtual nodes are scheduled for virtual Hadoop clusters, the latency between storage and compute nodes is not considered. The virtual nodes are randomly placed on compute nodes based on availability of resources. In a distributed cloud environment, virtual nodes will also be placed in different regions increasing the network latency since the data has to travel through the WAN network.

Sahara, a tool for cluster deployment on *OpenStack* platform can be configured to use empirical storage (volumes attached to nodes), persistent storage (cinder blocks) or object storage (swift). Empirical storage provides better data locality, because it runs HDFS on physical disk volumes attached to the compute nodes. The major drawback of this model being that data lives as long as the cluster lives, if the cluster is deleted the data also gets deleted. Another drawback is such that data resides on storage volumes attached to compute nodes, hence the storage space is limited to the capacity of the compute node. The above drawbacks means empirical storage violates the policies of cloud computing making it a bad storage model to be implemented for HDFS.

Persistent block storage provided by *Cinder* attaches disk volumes residing in storage nodes to VMs in compute nodes via the network. In this model the data is independent of the cluster hence the data does not get deleted with the life time of the cluster. Since storage nodes are separated from compute nodes, the storage capacity can be increased on demand. Even though cinders storage model suits the cloud framework and overcomes the drawbacks of empirical storage, it comes at the cost of data locality. The network latency between the compute nodes and the storage nodes means slower job completion times. The more the latency the slower the job completion time impacting performance of the Hadoop cluster.

Swift object storage also separates the compute resources from storage resources. This is beneficial if long term storage needs a required and data processing is only done periodically. Table I depicts the benefits of using different storage models.

TABLE I
DATA LOCALITY WITH DIFFERENT STORAGE MODELS.

	Ephemeral Storage	Block Storage	Object Storage
Used to	Run OS and scratch space	Add additional persistent storage to VM	Store data, including VM images
Accessed through	A file system	A block device that can be partitioned, formatted, and mounted (such as, */dev/vdc*)	The REST API
Access latency	Disk I/O	Network Link (iSCSI, FC)	Network Link
Accessible from	Within a VM	Within a VM	Anywhere
Managed by	nova	cinder	swift
Persists until	VM is terminated	Deleted by user	Deleted by user
Sizing determined by	Size settings (flavours)	User specifications	Amount available
Typical usage	10GB first disk, 30GB second disk	1TB disk	10s of TBs of dataset storage

III. LOCATION-AWARE PROCESS SCHEDULING

In this work, our primary objective is to minimize communication latency in cloud-based Hadoop deployment by placing VMs in virtual Hadoop cluster close to the storage node. The reduction in network distance will result in improved network transfer time. MapReduce jobs are highly dependable on the file transfer time. Smaller transfer time will result in faster MapReduce job completion.

Although reduction in network distance will improve data locality, most of the existing works overlook the ad-hoc property of a cloud environment. Virtual deployment of Hadoop has the adaptability of provisioning clusters on demand in dynamic locations within few minutes. In comparison to a physical Hadoop cluster with static location, the dynamic location property of a virtual cluster allows us to move computation to the data on demand. Fig. 3 depicts moving computation close to the data to avoid performance degradation due to the data transfer over the network. However, there are two conditions to be met by *OpenStack* in order to provision the cluster close to where the data resides: (1) *OpenStack* need prior knowledge of where the data is located. (2) A location-aware scheduling

978-1-5090-0676-2/16/$31.00 ©2016 IEEE

strategy to identify the physical hosts based on the input data location [10].

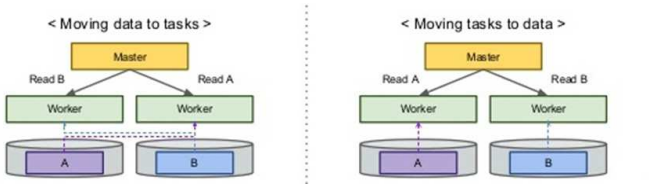

Fig. 3. A conceptual representation of moving computation to data.

The proposed solution is broken into two parts. Firstly, to identify data location and secondly, to create a customized filter for the scheduler for mapping the data location information to the compute node location, for VMs placements. In physical Hadoop environment, prior knowledge of data location is not important since clusters are provision first. In this work, we propose to provision virtual clusters close to the storage nodes.

The significance of placing the VMs in the right region is illustrated in Fig. 4, here the different geographical regions are named as *R1*, *R2*, *R3* and *R4*. A user from *R3* requests the controller in *R1* to provision a cluster of six nodes from *R3*. The controller uses the default nova scheduler to filter the available compute nodes and places the requested virtual machines on the highest weighted compute node. The process is repeated until all six virtual machines are provisioned. Looking at the results the scheduler has placed VMs 1 to 3 in *R3*, VM 4 in *R2*, VMs 5 and 6 in *R4*.

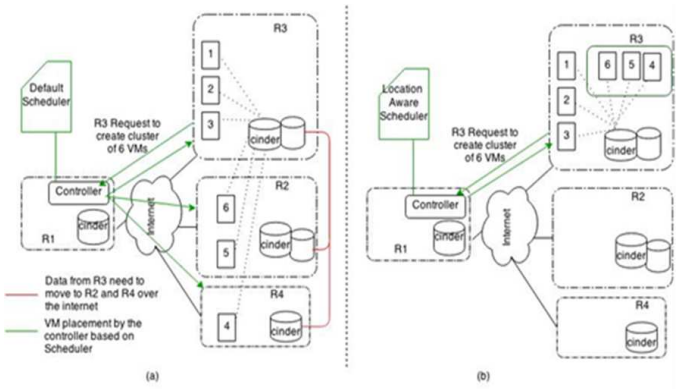

Fig. 4. Data location-aware VM placement: (a) Default nova scheduler (b) Location aware nova scheduler. (Adopted from [10]

OpenStack-Cinder will provision HDFS block storage volumes to all six virtual machines from the allocated block storage volumes of the region they belongs to. For example: VMs 1 to 3 will be assigned to storage volumes from *R3* and VM 4 and 5 from their respective regions. Since data to be processed resides in *R3* it has to be transferred to the volume in *R2* and *R4* or otherwise, VMs 4 and 5 should be pointed to the volumes in *R2* and *R4*. Assuming we consider the latter case as our example, once a MapReduce job is executed, data nodes 1 to 3 will be accessing the data through local network since their volumes reside in *R3*. This is not the case for nodes 4-6 as they need to access the data across the network, resulting in slow job completion time impacting system performance. Our proposed scheduler with data locality consideration will localized all the VMs in R3. By doing this, the network distance will significantly decreased, hence all the VMs can access the data via local networks.

IV. Proposed Framework Design

An instance scheduling process can be categorized as one of complex operation carried out by the nova scheduler. Given the complexity of the problem, nova scheduler process scheduling requests one at a time even in multiple instance requests as to reduce the complexity. The scheduler only deals with one VM request at time; this behavior reduces the delay introduced by the scheduling process. The computational load on the controller is minimized, compared to a multiple instance request process. This philosophy fits well into the remote procedure call (RPC) based message queue used by all *OpenStack* components. In order for the proposed solution to be compatible with the current Filter Scheduling framework, we decided its best not to change the native *OpenStack* code. The possible options available to implement a custom filter scheduler for *OpenStack* are as follows:

- Write a substitute of the current Filter Scheduler in Python.
- Write an entirely new custom filter in Python.
- Find a combination of the available filters that fits well.
- Implement a custom algorithm for the cost function of the filter scheduler.

We have implemented our proposed solution as weight and cost functions that can be plugged into to the existing filter scheduler. *OpenStack* uses a driver based architecture for customizing *OpenStack* features. A custom function or and algorithm can be written as a driver and be plugged into the *OpenStack* configurations. This kind of architecture gives the flexibility of modifying *OpenStack* without disturbing the core project. The scheduler is one such component which can be customized using the driver architecture, we have fully utilized this flexibility to implement our proposed solution into the *OpenStack* filter scheduler. As mentioned above these function will be applied to scheduling one instance at a time.

Fig. 5 depicts the proposed solution. A user will create the necessary MapReduce job binaries for the analysis and save them in *OpenStack Sahara*. When user is ready to execute an analysis job, the input and output directory URL need to be specified first. Next the user will create the cluster configurations specifying the number of master nodes and number of data nodes required for the cluster. Once the user request to launch the cluster, the request will be converted to REST calls and passed to the *nova-api*. The *nova-database* will pick up the request and place it on the queue.

The *nova-scheduler* will pick the request from the queue and call the nova scheduler. Nova scheduler will compare the resources available on each physical node with the resources requested for the first VM. All physical hosts with sufficient resources will be filtered by filter scheduler. Then, all filtered

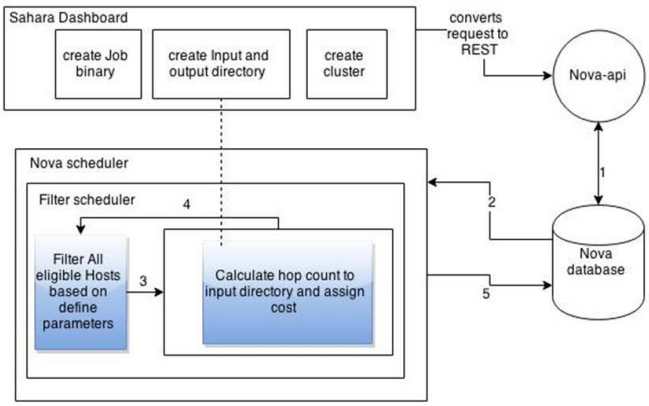

Fig. 5. The proposed location-aware scheduling framework for Hadoop deployment on cloud.

hosts will be send to the weighting stage to determine the host to place the virtual machine. The weighting function consist of multiple cost functions, *Location aware cost function will execute* $traceroute$ to determine the hop count from the input data location to filtered host. The RAM cost function will determine the available RAM in each filtered host and assign a cost value.

The weight function will then be called to calculate the sum of all the cost functions and multiply by the weight constant defined in $nova.conf$ file. The physical host with the lowest weight will be returned back to the queue as the selected host to place the virtual machine. The process will be repeated until the number of selected host is equal to the number of total nodes specified in the cluster.

Location-aware cost function takes into account the distance to the input data location from each host. Distance refers to the number of hops from the input data location to the filtered compute host. The hop count will be calculated for each filtered physical host. The host with the lowest hop count will be chosen to the place the virtual machine.

Determining the total cost of each node is the most essential part in the Location Aware Scheduling algorithm as depicted in Fig. 6. We will be using a Location-Aware Cost function f_{LAC} which takes the hop count and available RAM to calculate the total cost of each physical cost.

In general location aware cost function can be described by the following algorithm:

$$f_{LAC}(r,d) = \alpha R(r) + \beta D(d) \tag{1}$$

Where:

- r : free resources available on the host.
- d : Network distance between the input data source and the filtered hosts on the same OpenStack deployment.
- R, D : cost functions taking into account the probably non-linear behavior of the variables.
- α, β : real number weights.

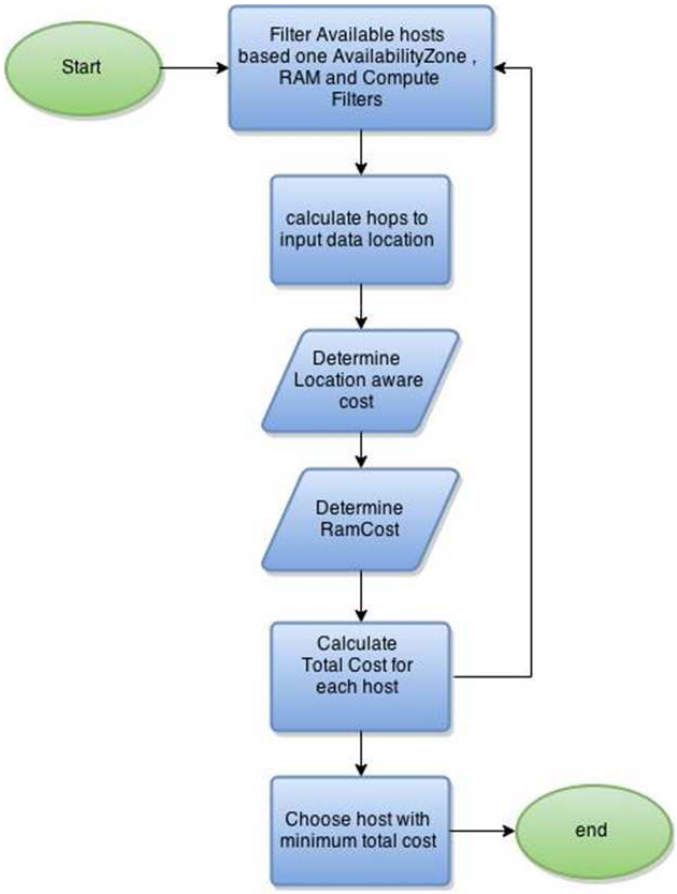

Fig. 6. Proposed location aware scheduling algorithm.

In context of the filter scheduler the Location-Aware Cost function can be described as follows:

$$f_{LAC}(r,d) = \sum_{p} [\alpha_p R_p(r_p)] + \beta D(d) \tag{2}$$

Where:

- R : Cost functions for all the resources already included in the OpenStack. R provides a function to determine free RAM and another function to track the number of failed scheduling attempts.
- D : Hop count from the filtered host to the input data location.
- α, β : real number weights listed in the nova.conf file.

Given the nature of the Location-Aware Cost function elaborated above, our goal will be to find the values maximizing the function. It is best to note that the filter scheduler takes the least cost value, hence the results has to be multiplied by -1 in order to find the maximum value.

V. Results and Analysis

In this section we discuss the performance of cloud based Hadoop cluster with the proposed Location aware virtual machine scheduler in place. In order to test our proposed solution, we executed the TeraSort benchmark suit with the

proposed Location scheduler configuration. TeraSort combines high CPU utilization, high storage disk I/O throughput and moderate networking bandwidth. In order to understand the impact of location aware scheduling, we have made a comparison between results obtained from TeraSort benchmark with the results of similar benchmark conducted using OpenStack default filter scheduler. This comparison of results will provide better indications on the benefits of location aware scheduling in cloud based Hadoop environments. Throughout this section we will refer to the default filter scheduler as "FS" and location aware filter scheduler as "LS".

A Hadoop virtual cluster have been configured for this experiment. The cluster had all 3 nodes places in three homogenous physical nodes (having similar configurations). The Hadoop cluster node configurations are listed in Table II.

TABLE II
EXPERIMENTAL HADOOP CONFIGURATIONS.

Master Node	Processor: 2 Virtual Cores Memory: 4GB Storage: 40 GB
Data Node	Processor: 2 Virtual Cores Memory: 4GB Storage: 40 GB

OpenStack Sahara was used to launch the Hadoop cluster.

A comparison of TeraGen job completion time between FS and LS is illustrated in Fig. 7. The overall trend of the graph suggests writing different sizes of data to a cluster with location aware placement is significantly faster as compared to writing data to a cluster with random placement of virtual machines. The writing time reduces by 34% when the data size is 1GB. A reduction 11-15% achieved when the data sizes increase to 3GB and 5GB respectively. Since the nodes are close to each other in location aware placement the write latency is reduced.

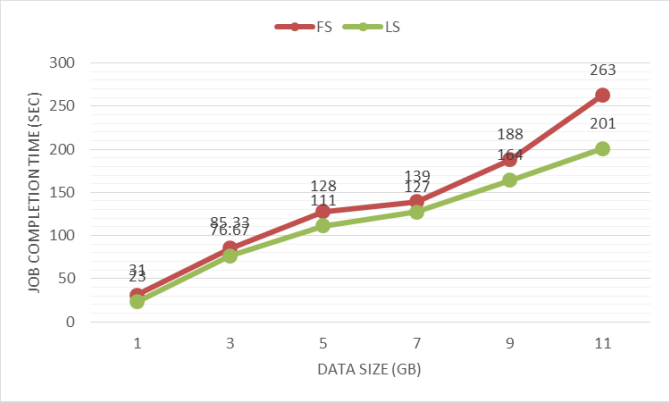

Fig. 7. TeraGen Job completion time comparison.

Fig. 8 depicts a comparison of TeraSort job completion time between FS and LS based virtual placement. As expected the job completion increases with the increase in sorted data file size for both FS and LS virtual machine placement. However, comparing it to FS placement, the overall TeraSort job completion is significantly reduced with location aware virtual machine placement. Similar behavior was noticed earlier during TeraGen job execution with LS based virtual machine placement.

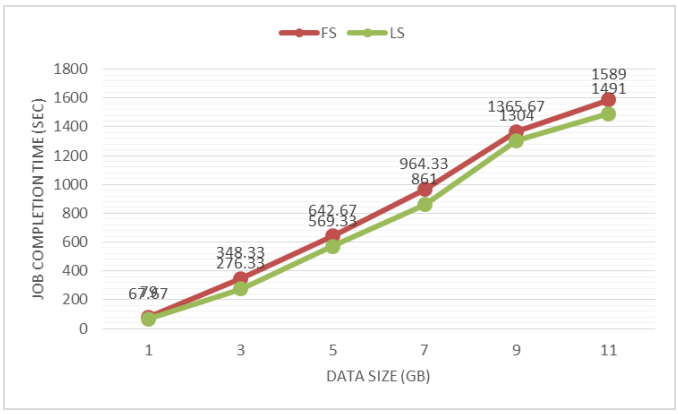

Fig. 8. TeraSort Job completion time comparison.

VI. CONCLUSION

In this paper, we present the current virtual Hadoop architecture and identify data locality as the key characteristic in deteriorating system performance. To achieve better data locality, we proposed a framework to identify the location of the data before clusters are provision, this method allows to provision clusters with least amount of network distance to data. We implemented this solution by improving the current filter scheduler in OpenStack to accept the network proximity to the storage node as additional parameter during the weighting stage. Our proposed solution was verified through TeraSort benchmarking suite and comparing the results with the TeraSort benchmarking results of standard OpenStack filter scheduler. As hypothesized our proposed data locality technique improves the performance MapReduce job completion time of cloud based Hadoop deployment.

REFERENCES

[1] M. A Vouk, "Cloud computing–issues, research and implementations," *CIT. Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[2] A. Corradi, M. Fanelli, and L. Foschini, "Vm consolidation: A real case based on openstack cloud," *Future Generation Computer Systems*, vol. 32, pp. 118–127, 2014.

[3] K. Kambatla, A. Pathak, and H. Pucha, "Towards optimizing hadoop provisioning in the cloud," in *Proc. of the First Workshop on Hot Topics in Cloud Computing*, 2009, p. 118.

[4] T. Nguyen and W. Shi, "Mapreduce in the cloud: Data-location-aware vm scheduling," *ZTE Communications*, vol. 4, p. 005, 2013.

[5] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: locality-aware resource allocation for mapreduce in a cloud," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 58.

[6] J. Park, D. Lee, B. Kim, J. Huh, and S. Maeng, "Locality-aware dynamic vm reconfiguration on mapreduce clouds," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*. ACM, 2012, pp. 27–36.

[7] G. Porter, "Decoupling storage and computation in hadoop with superdatanodes," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 41–46, 2010.

[8] B. T. Rao, N. Sridevi, V. K. Reddy, and L. Reddy, "Performance issues of heterogeneous hadoop clusters in cloud computing," *arXiv preprint arXiv:1207.0894*, 2012.
[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.
[10] A. F. Thaha, M. Singh, A. H. M. Amin, N. M. Ahmad, and S. Kannan, "Hadoop in openstack: Data-location-aware cluster provisioning," in *Information and Communication Technologies (WICT), 2014 Fourth World Congress on*, Dec 2014, pp. 296–301.
[11] A. VMware, "Benchmarking case study of virtualized hadoop performance on vmware vsphere 5," *Benchmarking Case Study Virtualized Hadoop Perform. VMware VSphere*, vol. 5.
[12] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
[13] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.