Hadoop Acceleration USIN Open Flow Technique

Santhosh Pradhan. K[1] Parthipan. V[2]
[1,2]Department of Computer Science And Engineering
[1,2]Saveetha School of Engineering, Saveetha University Chennai, India.

Abstract— This paper is about of how Hadoop can control its network resources to improve performance using OpenFlow. . OpenFlow is a very popular example of software-defined network technology. Our study will explore the use of OpenFlow to provide better link bandwidth for shuffle traffic and decrease the time which Reducers have to wait to gather data from the Mappers. Hadoop's distributed compute framework which is called as MapReduce, exploits the distributed storage architecture for Hadoop's distributed file system which will deliver a reliable parallel processing services for arbitrary algorithms. The shuffle phase of Hadoop's MapReduce are the computation involves movement of intermediate data from the Mappers to Reducers. Reducers often are delayed due to inadequate bandwidth between them and thereby lower the performance of the cluster. Our experiments shows the decrease of execution time for a Hadoop job and when the shuffle traffic can be used for more of the available bandwidth on a link. Our approach illustrates how high performance of the computing applications can improve performance by controlling their underlying network resources. The work presented in the paper is for a starting point for some experiments being done as part of SC12 SCinet Research Sandbox which will be quantified the performance advantages of different versions of Hadoop that uses OpenFlow to dynamically adjust the network of local and wide area Hadoop clusters.

Key words: Software Defined Networks, Hadoop, Mappers, BigData, OpenFlow

I. INTRODUCTION

Hadoop has been emerged as a prominent platform in data intensive computing. It provides a reliable storage and analysis system that would be scalable and built from the standard inexpensive of hardware. It is an evident that HDFS's can performance various characteristics that are very much dependent on the performance of disk. Its distributed storage system which is called HDFS is a single and are consolidated storage platform with a new type of repository where are structured data and complex data can be combined easily. The analysis system is called as MapReduce framework which exploits the distributed storage architecture of HDFS to deliver scalable and reliable parallel processing services for arbitrary algorithms [1].

The MapReduce Framework's performance depends on the compute power of the cluster and however the data movement in the Hadoop clusters is the major factor in the overall performance of the Hadoop cluster. Even the jobs are scheduled and closer to where its data lie in the cluster and network traffic is still an issue because of inadequate bandwidth in nodes when they need it. One approach foe which to address the network issues in Hadoop is to provide an application developer with ability to control their network just as they can control memory to assign to a task or what files an application can write to. Ability to the control bandwidth as allocation as to be needed by an application are expected to be provided improvement in the Hadoop performance just like the specialized interconnects, such as InfiniBand can be. Software Defined Network (SDN) is the revolutionizing of the networking industry and offering a multiple opportunities for control on the network to the entities that did not or otherwise say in how their network was configured for their use.

These entities could be of anyone like network operators, service providers or application developers, and even end users also. The leading SDN technology is based upon the OpenFlow protocol which is standard that has been designed for SDN and already being deployed in the variety of networks both hardware- and software-based.

In the OpenFlow network architecture and the control data planes are to be decoupled and state are logically centralized, and the underlying the network infrastructure is to be abstracted from the applications. As a result, enterprises and carriers can gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs.

II. OVERVIEW OF HADOOP

MapReduce is a programming model where a MapReduce program can consist of two functions—map and reduce, each of which defines a mapping from and increasing cluster size decreases the execution time one set of key-value pairs to another. MapReduce programs can be written in various languages; Java, Ruby, Python, and C++. Files in HDFS are split into smaller blocks these functions work the same for the cluster size. Increasing data size causes increased execution. MapReduce programs are inherently parallel, with Map tasks running concurrently, followed by their corresponding Reduce tasks, which also run concurrently.

HDFS (Hadoop Distributed File System) is a distributed file system that any data size, but the execution time depends on the data size and stores and manages the data in a Hadoop cluster. As illustrated in Figure 1, there is central node called NameNode to store the meta-data of the file system and other nodes called Data Nodes that store the data., typically 64MB, and each block is stored separately at a Data Node and replicated as per the specified replication factor to provide data durability. A HDFS client contacts the Name Node to get the location of each block,

And then interacts with Data Nodes responsible for the data.

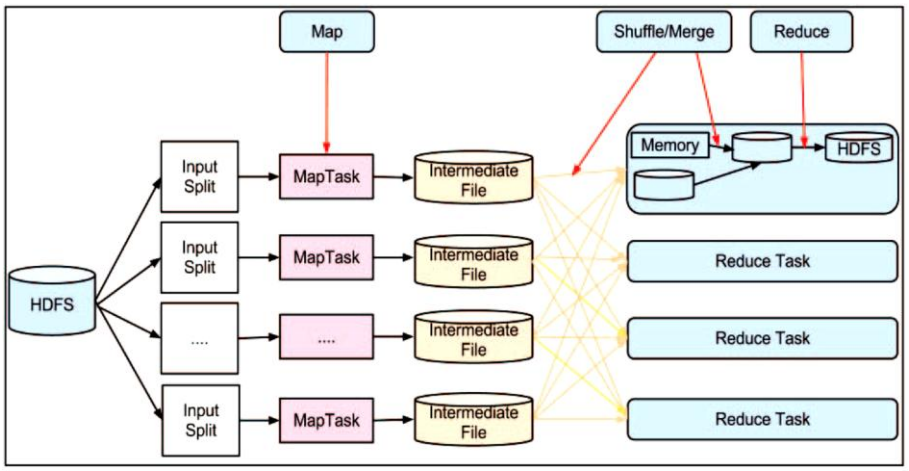

Fig. 1: Data Processing in Hadoop

Hadoop Map Reduce Framework consists of two types of components that control the job execution process: a central component called the Job Tracker and a number of distributed components called Task Trackers. Based on the location of a job's input data, the Job Tracker schedules tasks to run on some Task Trackers and coordinates their execution of the job. Task Trackers run tasks assigned to them and send progress reports to the job tracker.

MapReduce's execution model has two phases, map and reduce, both requiring moving data across nodes. Map tasks can be run in parallel and are distributed across nodes available. The input data, that is stored in HDFS prior to beginning of the job, is divided into datasets or splits, and for each split a map task is assigned to Task Tracker on a node that is close to that split. The map computation begins when its input split is fetched by the Task asker, which processes the input data according to the map function provided by the programmer, generates intermediate data in the form of <key, value> tuples, and partitions them for the number of reduce tasks. In the reduce phase, the Task Trackers assigned to do the reduce task fetch the intermediate data from the map Task Trackers, merge the different partitions, perform reduce computation, and store the results back into HDFS.

Hadoop network and compute architecture considerations have been presented in [11], which explores the effect of network on Hadoop performance. The work identifies the burst nature of Hadoop traffic, the overall network speed and Hadoop's design for reliability as some of the factors affecting Hadoop performance. Many industry and research efforts are addressing these issues. [12] Explores the use of Infiniband interconnect in Hadoop clusters. [13] Presents a new pipeline design that overlaps the shuffle, merge and reduce phases, along with an alternative protocol to enable data movement via RDMA (Remote Direct Memory Access). Instead, we focused on utilizing the network control capabilities provided by Open Flow to provide resources to straggling Reducers, who can be speeded up by prioritizing their use of the network.

Network traffic in Hadoop can be separated into several categories:

- HDFS read and write by client
- HDFS replication
- Interaction between Task Trackers, which includes Hadoop shuffle traffic
- Hadoop split traffic between HDFS and Task Trackers
- Hadoop results stored in HDFS
- Interactions between Name Node and Data Node
- Interactions between Job Tracker and Task Tracker

In addition to these traffic categories, the cluster may be multi-use cluster and may run other applications concurrently. This background traffic can also affect completion times of Hadoop-jobs [11]. If the network connecting the Hadoop cluster is able to provide timely and orderly delivery of data, and the application (Hadoop) is able to control the network based on its needs for different categories for traffic, then the application can perform more operations concurrently, and thus improve its performance. In a Hadoop cluster, the characteristics of the network are known a-priori and can be used to the advantage of setting up a more performant network, similar to the earlier

Circuit-switching ATM networks. The technology that makes it possible to setup flow-paths similar to circuit switching is ONF's Open flow.

III. OVERVIEW OF OPEN FLOW

In a classical router or switch, the data path and the control path occur Open Flow Switch presents disk I/O performance on the same device. An Open Flow Switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller. When an Open Flow Switch receives a packet it has never seen before, for which it has no matching flow entries, The Open Flow Switch and Controller communicate via the Open Flow protocol, which defines messages, such as packet-received, modify-forwarding-table, and get-stats, that are exchanged between the Open Flow switch and the controller. The data path of an is the key consideration to put a clean flow table abstraction; each flow table entry contains a set of packet fields to match, and an action (such as send-out-port, modify field, or drop). It sends this packet to the controller. The controller then makes a decision on how to handle this packet. It can drop the packet, or it can add a flow entry directing the switch on how to forward similar packets in the future. [4].

A. I/O-Intensive Job:

The average performance of VM Hadoop clusters becomes worse as the number of simultaneous executed jobs

increases. The performance degradation between 1 job and 4 jobs is almost 31.7%. The gap between 1 job and 2 jobs is almost Hadoop jobs on the same physical servers and this result also comes from the negligible because the disk striping allows 2 parallel I/O writes to 2 disks at one time.

Another feature of Open Flow is simple Quality-of-Service (Qu's). Thus, the difference of the reduce phase, the disk I/O, as we shown in previous sections. For I/O-intensive jobs and of course the less multiplicity gives better execution performance.

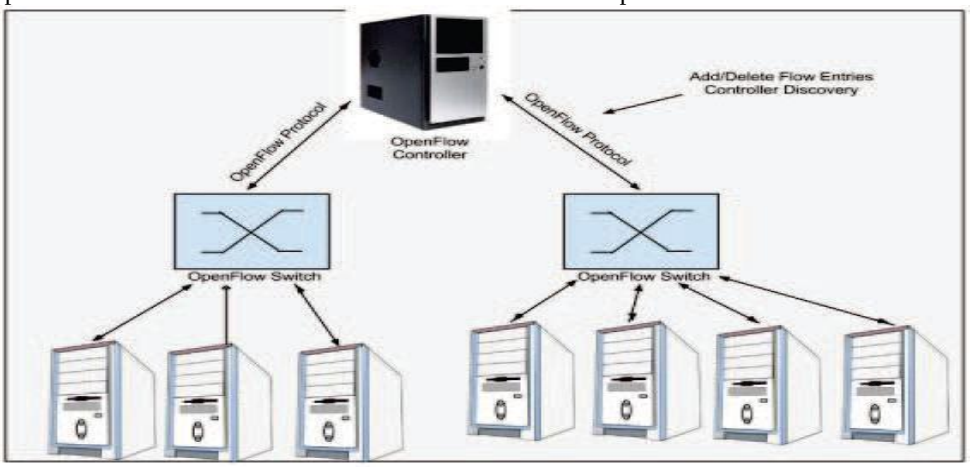

Fig. 2: Open Flow Paradigm

B. CPU-Intensive Job:

The average times of job total in Fig.13 illustrate CPU intensive job is not affected by the number of simultaneous

Executed jobs in multiple clusters on the same physical servers. Since the number of cores on a physical server is 4,

4 simultaneous executed jobs with 4 VCPUs do not affect the performance each other. Thus, VM Hadoop cluster works for CPU-intensive job does not interrupt other users' jobs if the context switch among VCPUs is not happened. Thus, Open Flow enables one to easily deploy innovative routing and switching protocols in a network. An Open Flow switch provides limited support for Qu's through a simple queuing mechanism. One or more queues can attach to a

Port and be used to map flows on it. Flows mapped performance degradation does not occur to use up to 4 VCPUs. To a specific queue will be treated according to that queue's Qu's configuration (e.g. minimum rate).

IV. OPENFLOW AND HADOOP

Using the two features of Open Flow discussed above, we can provide different network characteristics to various

Categories of traffic in a Hadoop cluster. Some simple examples are: a) provide Hadoop traffic with higher priority

Over other traffic in the cluster and b) provide Hadoop shuffle traffic with higher priority. To illustrate this we setup a simple

Hadoop cluster with Open Flow switches (Fig. 4). The 10 node Hadoop cluster ran the Cloud era distribution of Hadoop on 3 physical systems (Server1, Server2 and Server3) connected to a physical switch (Figure 4). The Hadoop nodes were run on virtual machines (VMs) in an EServer virtualized environment [6]. The Cloud era Manager (CM) ran on one of the VMs in the cluster and managed the Hadoop cluster. The Open Switch (OVS) [7] is the default network stack for the EServer. Open Switch is a multi-layer software switch that supports Open Flow standard 1.0. We used Big Switch's open source Apache-license, Java-based Open Flow Controller called Floodlight [8] to setup flow entries in the Open Flow switches. We also ran our experiments with another Open Flow switch, LINC, which is an open-source software switch developed in Erlangen [11] that provides a full implementation of Open Flow 1.2 standard.

V. EXPERIMENTS

For the experiments, we used the Hadoop Sort program to run as the job under test. Hadoop comes with a Map Reduce Program that does a partial sort of its input. It is very useful for benchmarking the whole Map Reduce system, as the full input dataset is transferred through the intermediate shuffle phase. The three steps of the Hadoop Sort program are: generate random data, perform sort, and validate the results [3]. For the first set of experiments we used the Qu's capability of the Open Flow standard so that we could explore the use of Open Flow in a small cluster with just three systems. First, we setup the maximum rate (Qu's) on all the three Open Flow switches to be 300 Mbps. Next, we created three queues, the first (q0) with 50 Mbps, second (q1) with 200 Mbps and the third (q2) with 5 Mbps. Even though our network supported higher bandwidth we used these rates to artificially create congestion and investigate the benefit of Open Flow switches under such situations. We generated additional traffic in the cluster using a utility called Pier [9]. There are several mechanisms to inject cross traffic or congestion in to the network. Some popular mechanisms to achieve this are Pier, Scaly, Nemesis and Tcprelay. We chose Pier as it is simple to use and sufficient for our experimental purposes to create cross traffic in the underlying network. We ran a Pier server on one VM and a client on another VM on a different physical system.

In all cases, we ran Hadoop Sort jobs for the different input payloads (size: 0.4 MB, 4 MB, 40 MB, 400 MB and 4GB).

In the first (baseline) experiment (Experiment 1), we did not insert any flows in the Open Flow switches. Thus, all the Traffic traveled through the queue q0. No other significant traffic was present in the cluster. We recorded the time taken for each sort job. In the second experiment

(Experiment 2), we added one flow entry to each of the Open Flow switches, which put the Pier request traffic which uses port 5001 on queue q2 (low maximum rate). Again, we ran the Hadoop Sort job on the input payloads and recorded the time taken for each sort job. In the third experiment (Experiment 3), we used Open Flow to provide different network characteristics to some traffic categories of Hadoop traffic. In Hadoop, the Task Trackers communicate with each other using HTTP on port 50060. The major part of this interaction between Task Trackers is due to the shuffle phase where intermediate data is moved from Mappers to Reducers. We added new flow entries in all the Open Flow switches to direct all traffic on port 50060 to queue q1, which had higher maximum rate. Iperf traffic continued on queue q2 and the rest of the traffic used queue q0. We ran the same Hadoop Sort job for the same input payloads and recorded the time taken for each sort job.

In the next experiment (Experiment 4), we added one flow entry to each of the Open Flow switches, which put all the Traffic including Hadoop and Iperf traffic on queue q3 (highest maximum rate). Again, we ran the Hadoop Sort job on the input payloads and recorded the time taken for each sort job.

VI. CONCLUSION

It shows that the Hadoop job performs better when IPerf traffic is assigned to a queue with lower maximum rate and the rest of the traffic is assigned to. Experiments 3 and 4 showed that Hadoop job performs queue with higher maximum rate. Experiment 3 shows the improvement in performance seen for the case where Hadoop shuffle traffic is assigned to the queue with a higher rate better when shuffle traffic is given higher priority. In addition, we also performed the same experiments on a cluster with LINC Open Flow switches and saw similar results. We also observed that in general Hadoop jobs are long running and the reconfiguration time (in MS) to setup the switches to the overall job completion times. Our next set of experiments is in progress, where we are working to setup different topologies for different categories of traffic and to do this in a dynamic manner. We believe that this work shows promise for achieving one of the goals of software-define networking, which with different flow entries is very small compared is to provide control over the network to applications that use the network.

REFERENCES

[1] Stuart Bailey, Sandhya Narayan, Anand Daga, Robert Grossman, Matthew Greenway, OpenFlow Enabled Hadoop Over Local and Wide Area Clusters. Demo at the “SCinet Research SandBox”, SuperComputing 2012.
[2] HADOOP: Scalable, Flexible Data Storage and Analysis.http://www.cloudera.com/wpcontent/uploads/2010/05/Olson_IQT_Quarterly_Spring_2010.pdf
[3] ONF, Software-Defined Networking: The New Norm for Networks. https://www.opennetworking.org/images/stories/downloads/whit e-papers/wp-sdn-newnorm.pdf
[4] Tom White, Hadoop: The Definitive Guide, 2nd edn., O’Reilly, Sebastopol, CA, 2011.
[5] Introduction to OpenFlow, https://www.opennetworking.org/standards/intro-to-openflow
[6] XenServer, http://blogs.citrix.com/2011/09/30/xenserver-6-0-ishere/
[7] Open Vswitch, http://openvswitch.org/
[8] FloodLight, http://floodlight.openflowhub.org/
[9] LINC Switch, http://flowforwarding.org/
[10] Iperf, http://code.google.com/p/iperf/
[11] Jacob Rapp. Hadoop Network & Compute Architecture Considerations. Hadoop World, 2011.
[12] S. Sur, H. Wang, J. Huang, X. Ouyang, and D. K. Panda. Can High-Performance Interconnects Benefit Hadoop Distributed File System? In Workshop on Micro Architectural Support for Virtualization, Data Center Computing, and Clouds (MASVDC). Dec 2010.
[13] Yandong Wang, Xinyu Que, Weikuan Yu, Dror Goldenberg, and Dhiraj Sehgal. Hadoop acceleration through network levitated merge. In Proceedings of the 2010 International Conference on Superc
[14] S. Ibrahim, H. Jin, L. Lu, B. He, and S. Wu, "Adaptive disk I/O scheduling for MapReduce in virtualized environment," Proc. of ICPP 2011, pp.335-344, 2011.
[15] H. Liu and D. Orban, "Cloud MapReduce: a MapReduce implementation on top of a cloud operating system," Proc. of 11th IEEE/ACM CCGrid, pp.464-474, 2011.
[16] A. Mandal, Y. Xin, I. Baldine, J. Chase, and V. Orlikowski, "Provisioning and evaluating multi-domain networked clouds for Hadoop-based applications," Proc. of CloudCom, pp.690-697, 2011.
[17] Y. Geng, S. Chen, Y. Wu, R. Wu, G. Yang, and W. Zheng, "Locationaware MapReduce in virtual cloud," Proc. of ICPP 2011, pp.275-284, 2011.
[18] F. Chang, J. Ren, and R. Viswanathan, “Optimal resource allocation in clouds,” Proc. of CLOUD 2010, pp.418-425, 2010.
[19] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis, "Flexible use of cloud resources through profit maximization and price discrimination," Proc. of ICDE, pp.75-86, 2011.
[20] S. Kadirvel and J. A. B. Fortes, "Grey-box approach for performance prediction in Map-Reduce based platforms," Proc. of 21st ICCCN, pp.1-9, 2012
[21] H. Mao, Z. Zhang, B. Zhao, L. Xiao, and L. Ruan, “Towards deploying elastic Hadoop in the cloud,” Proc. of CyberC 2011, pp. 476–482, 2011.
[22] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," IEEE Trans. on parallel and distributed systems, pp.985-997, Vol.22(6), 2011.

[23] A. Qin, D. Tu, C. Shu, and C. Gao, "XConveryer: Guarantee Hadoop throughput via lightweight OS-level virtualization," Proc. of 8th GCC, pp.299-304, 2009.
[24] J. Chen, Y. T. Larosa, and P. Yang, "Optimal QoS load balancing mechanism for virtual machines scheduling in Eucalyptus cloud computing platform," Proc. of BCFIC, pp.214-221, 2012.
[25] B. Li, E. Mazur, Y. Diao, A. McGregor and P. Shenoy, "A platform for scalable one-pass analytics using MapReduce," Proc. of SIGMOD '11, pp.985-996, 2011
[26] E. Krevat, T. Shiran, E. Anderson, J. Tucek, J. J. Wylie and G. R. Ganger, "Applying simple performance models to understand inefficiencies in data-intensive computing," CMU-PDL-11-103, 2011

