Network-Aware Scheduling of MapReduce Framework on Distributed Clusters over High Speed Networks

Praveenkumar Kondikoppa , Chui-Hui Chiu, Cheng Cui, Lin Xue and Seung-Jong Park
Department of Computer Science, Center for Computation & Technology,
Louisiana State University, LA, USA
pkondi1,cchiu1,ccui,lxue2,sjpark@lsu.edu

ABSTRACT

Google's MapReduce has gained significant popularity as a platform for large scale distributed data processing. Hadoop [1] is an open source implementation of MapReduce [11] framework, originally it was developed to operate over single cluster environment and could not be leveraged for distributed data processing across federated clusters. At multiple federated clusters connected with high speed networks, computing resources are provisioned from any of the clusters from the federation. Placement of map tasks close to its data split is critical for performance of Hadoop. In this work, we add network awareness in Hadoop while scheduling the map tasks over federated clusters. We observe 12 % to 15 % reduction of execution time in FIFO and FAIR schedulers of Hadoop for varying workloads.

Keywords

Federated Clouds, Hadoop Scheduling

1. INTRODUCTION

Large scale scientific and engineering applications generate vast amount of data. For example, the Large Hadron Collider project generates peta bytes of data which need to be processed, analyzed and stored. Cloud computing coupled with MapReduce programming paradigm has made huge strides in big data analysis. Hadoop which is an open source implementation of MapReduce is extensively used on compute clouds at Amazon, Yahoo and Rackspace. Many of these commercial clouds provide unlimited computing power. In addition to those commercial clouds, many universities and research laboratories have deployed their own private clouds for research and education.

To maximize the utilization of heterogenous public and private clouds, researchers want to launch their jobs over those clouds at the same time. For example at Louisiana State University, researchers can make use of different cloud resources: LONI [5], FutureGrid [4] and Xsede [8]. This paper focuses on how to aggregate cloud resources from private and public clusters and run MapReduce jobs more efficiently.

FederatedClouds'12, September 21, 2012, San Jose, California, USA.

Conventional MapReduce(Hadoop) schedulers are optimized to operate over single cluster which connects computing nodes within local networks. However with considerable administrative efforts MapReduce can be deployed over distributed clusters. There are several issues on performance of MapReduce over distributed clusters. For example, network latency and bandwidth between a master node and slave nodes reduce the performance of MapReduce because MapReduce configuration requires a direct communication between a master node and slave nodes directly. Additionally, scheduling tasks among slave nodes which are scattered over remote clusters becomes an important factor on performance because of data locality.

There are several ways of deploying MapReduce Hadoop over distributed clusters. First approach is to set up independent Hadoop clusters at each site, a centralized job scheduler to manage job submission. Second approach is to aggregate underlying physical clusters as a single virtual cluster and run MapReduce tasks on top of this virtual cluster. And the third approach would be to make MapReduce framework work directly with multiple clusters without additional virtual clusters. In this paper, we deploy global MapReduce over federated clusters to utilize the computing effectively. However, deploying MapReduce framework directly on multiple cluster degrades the performance because of failing to exploit data locality. Our research provides a method to add network awareness to global MapReduce so that a scheduler has the information about data locations to launch map tasks at nodes with data.

2. BACKGROUND

2.1 MapReduce Paradigm

MapReduce framework abstracts programmers from the complexities of input data distribution, parallelization, fault tolerance and synchronization of various constructs. The open source implementation of MapReduce framework consists of computing environment called Hadoop and distributed filesystem called Hadoop distributed filesystem (HDFS). Hadoop acts like runtime environment for the Map and Reduce functions provided by programmers. HDFS is designed to provide replication, integrity of the data and fault tolerance. The architecture of Hadoop has master services namely Jobtracker and Namenode and client services namely Tasktrackers and Datanodes. The Jobtracker receives the job submitted by the user and splits it into map and reduce tasks. It

assigns tasks to Tasktrackers, monitors about task completion status and when all the tasks for a job are complete it reports back to the user.

2.2 Related Works

Cardos et al. [10] suggest different architectures to set up MapReduce framework when source data and computation are distributed. They recommend three architectures, Local MapReduce where in data is moved to centralized cluster to perform computation. Global MapReduce where cluster is established with nodes from all the clusters and Distributed MapReduce where in small independent MapReduce clusters similar to architectural overlay. However, their suggestions are different configurations and fail to address data locality concerns.

Yuano Luo et al. [12] introduce hierarchical MapReduce by organizing the resources into two layers: a top layer consisting of job scheduler and workload manager and a bottom layer consisting of distributed clusters running local MapReduce. When a job is submitted, it is split into sub jobs, which are assigned to distributed clusters. A Jobtracker at each of the clusters reports the job progress and sends result back to a global controller where reduce tasks take place. The disadvantage of this approach is that it is suitable for map-intensive or map-mostly type of applications. If intermediate data generated from the map phase are huge and need to be transferred to a reducer then data transfer becomes bottleneck for the performance of Hadoop. Majority of the data intensive applications involve significant reduce phase computations. Also users need to add additional programming construct to perform global reduce at the global controller.

2.3 Scheduling in Hadoop

2.3.1 FIFO Scheduler

Default scheduling in Hadoop is through FIFO scheduler. The communication between the master node and slave nodes happen through heartbeat packets. When a slave node with empty map slot sends a heartbeat packet to the Jobtracker, scheduler checks the head of the queue job for the tasks. If the job has tasks whose input split is located on the slave node then task is assigned to the Tasktracker. If scheduler does not find local map task then it assigns only one non local map task to this Tasktracker.

2.3.2 FAIR Scheduler

FAIR scheduling ensures that every job gets an equal share of resources over time. If there is single job running then entire cluster is allocated for the job. If new jobs are submitted then task slots which free up are allocated to the newly submitted jobs, so that each job gets roughly equal share of cpu time. Small jobs gets finished within reasonable time and long jobs are not starved. FAIR scheduler organizes jobs into pools and inside pool FIFO or FAIR queuing of the jobs can be configured.

2.4 Global MapReduce over Federated Clusters

MapReduce Hadoop cannot be easily deployed over distributed clusters of different administrative domains to form single MapReduce cluster, since master node services require direct communication between slave node services. If clusters belonging to different administrative domains need to be federated, then considerable administrative effort is needed to make internal nodes of clusters to communicate with each other. We address this issue of internal nodes accessible from outside by introducing a virtualization on top of all the physical clusters. Cloud computing software like Eucalyptus [3] or Openstack [7] is installed on each of the underlying physical clusters. Computing units are provided as virtual machines from any of the distributed clusters. A virtual layer unifies the resources provided by all the distributed clusters into single infrastructure layer of virtual machines. MapReduce cluster is set up on top of these virtual machines where in a single Jobtracker takes the responsibility of scheduling the jobs and managing the Tasktrackers as shown in figure 2. The advantage of this approach is that, it is easy to manage distributed clusters through cloud computing software. However, the critical aspect in this approach is to make a scheduler be topology-aware in order to accomplish data local computation of map tasks. The proposed work focuses on improving data locality while scheduling map tasks for global MapReduce so that all types of data intensive applications such as, map-only, map-mostly and map-reduce application types are supported. In the below sections, we describe data locality issue and propose network aware scheduling to make global MapReduce scheduler data aware.

2.5 Case for Data Locality over Federated Clusters

Unlike HPC systems where computation and file system are decoupled, in MapReduce Hadoop filesystem (Datanode) and computation (Tasktracker) are co-located on a node. When data is transferred to HDFS it is replicated to datanodes so that Tasktracker running on that node will have data blocks prior to execution of a task. In the current implementation of MapReduce Hadooop, data locality is achieved by replicating the data at three levels of cache. At the first level, a data split is stored on a node. At the second level, a data split is saved on a node in the same rack and at the third level, a data split is stored off-rack. To achieve data locality, map tasks are scheduled on nodes which already have the data blocks (this constitutes node local). If it cannot find such a node then tasks are scheduled on nodes which can fetch data blocks from any node in the same rack (this constitutes rack local). If both situations are not met then the map task is scheduled on any tasktracker requesting for a task. Thus MapReduce scheduling algorithms exploiting data locality are based on the assumption of multiple nodes in a rack, which are connected by an aggregate switch.

Figure 1 shows the typical architecture for federated clusters which may be connected by regular internet or by dedicated link. Furthermore, if virtualization is added on top of these federated clusters to provide virtual machines as computing units, then Hadoop's assumption of racks connected through aggregate switch cannot be extended to achieve data locality. To get maximum performance of Hadoop, it is critical to configure Hadoop so that it knows the topology of the entire network while making scheduling decisions.

3. NETWORK AWARE SCHEDULING

The goal of task scheduling in Hadoop is to move computation towards data. If it can't meet this objective then a task is scheduled on a node which is requesting for a task, this causes data to be transferred to compute node for pro-

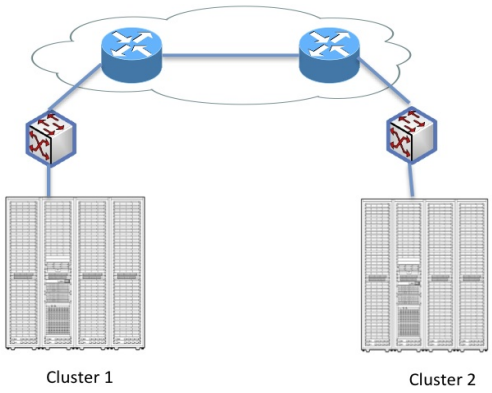


Figure 1: Federated Clusters

cessing. When resources are provisioned by distributed clusters, moving the data across the network causes the degradation in Hadoop's performance.

In case of distributed clusters, the Tasktrackers from any of underlying cluster might request for a map task. The technique to make Hadoop scheduler aware of network topology is to extend the rack aware feature of the existing Hadoop scheduler to provide one more level of caching. An administrator controlled script will hold the information about which cluster the Tasktracker is associated with. In our implementation of global MapReduce architecture, network topology script has information about virtual machines and physical location of the cluster from which they are provisioned. We use Neuca [3] enabled Eucalyptus cloud computing software to provide virtual resources from the distributed clusters. The locations of the virtual machines are organized as $/\text{cluster}_N/\text{rack}_N/\text{vm}_N$. Where cluster_N denotes the physical which cluster, rack_N denotes the rackid and the vm_N indicates the hostname for the virtual machine.

We enable delay scheduling [13] to take maximum use of data locality while scheduling a task. When head of the queue task doesn't find a compute node with data then scheduling of the task is delayed for a specified duration of time. If any of the compute nodes become free with a data split corresponding the job being processed then scheduler assigns a map task to requesting Tasktracker. Duration for which a head of the queue map task is to be delayed is based on the average length of the map tasks for a job hence requires careful tuning.

4. EXPERIMENTAL SETUP

CRON [9] is an Emulab based testbed; a cyberinfrastructure of reconfigurable optical networking environment that provides multiple emulation testbeds operating up to 10Gbps bandwidth. It consists of two main components: (i) hardware (H/W) components, including a switch, optical fibers, network emulators, and the workstations required to physically compose optical paths or function at the ends of these paths; and (ii) software (S/W) components, creating an automatic configuration server that will integrate all the H/W components to create virtual network environments based on the users requirements. All components are connected with 100/1000 Mbps Ethernet links for control. Each workstation is connected with 10 Gbps optical fibers for data movement.

Table 1: Jobs with Varying Task Lengths

Task	Average Execution Time
Map	5 sec
Map	8 sec
Map	14 sec

We use Eucalyptus as cloud computing software, we setup two NEuca-patched [6] Eucalyptus Clouds to construct the distributed cloud scenario. The NEuca patch attaches additional exclusive virtual NICs to a VM for application data transmission. We have two computing nodes in each Cloud and each computing node accommodates three VMs. Then, all VMs connect to each other through the additional NICs to form a virtual cluster. We deploy network aware Hadoop over this virtual cluster as shown in figure 2

We avoid multiplexing VMs on a physical machine for the better and stabler execution. Each VM is allocated with 2 physical CPU cores, 2 GB of physical memory, and 10 GB of local hard disk space. Regarding the NIC, the VirtIO interface is adopted to serve the virtual NIC so that the transmission rate is as high as the Hypervisor can provide.

4.1 Experiments

4.1.1 *Experiments with Native Hadoop*

In this set of experiments we run Hadoop over distributed cluster without network awareness. One of the node will be a master node running Jobtracker and Namenode services. Remaining nodes from both clusters run slave services namely Tasktrackers and Datanodes. We use the wordcount application for evaluation. Input file size for word count application is varied so that average execution time for map task differs. Multiple jobs consisting of tasks with different execution times as shown in the table 1 are submitted.

4.1.2 *Experiments with Network Aware Hadoop*

In this set of experiments, Hadoop with network awareness is configured over the federated cluster. One Jobtracker will manage the scheduling over all the clusters. Tasktrackers from different clusters request for map tasks as and when they get map slots freed. Jobtracker uses the cluster awareness to schedule tasks on these Tasktrackers there by im-

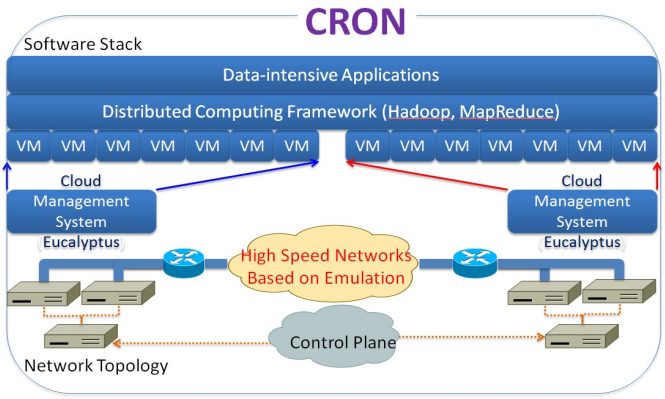


Figure 2: Experimental Setup

Table 2: Evaluation Environment

Nodes	Quantity	Hardware and Hadoop
Master Node	1	2 CPU core, 2 GB RAM, 10Gbps NIC Jobtracker and Namenode
Slave Nodes	12	2 CPU core, 2 GB RAM 10Gbps NIC Tasktracker and Datanode Hadoop-0.20.203.0 2 Map and 2 Reduce Tasks

proving the data locality. We perform experiments with varying inter-cloud bandwidth and delay and measure the performance of Hadoop with network awareness.

5. RESULTS

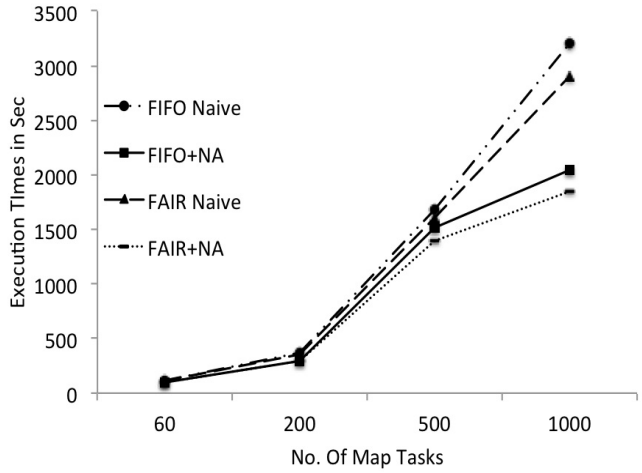


Figure 3: Execution times for Different schedulers for 1Gbps ,30ms inter cloud bandwidth and delay

We compare the execution times of Hadoop native scheduling with network-aware Hadoop for varying delay values. Figure 3 shows the execution times for varying number map tasks, FIFO scheduler with network awareness identified as FIFO+NA, shows the reduction in execution time of about 12 % on average and maximum reduction of 15 % when number of map tasks is higher compared to Hadoop Naive scheduler identified as FIFO Naive. FAIR scheduler with network awareness identified as FAIR+NA also shows the similar results, FAIR scheduler in general takes less execution time compared to FIFO, The reason being effective utilization of the cluster. Network aware Hadoop shows maximum performance improvement for map tasks between 500 to 1000. As the map task number increases, the data splits associated with the map tasks will be spread over the distributed clusters. When Jobtracker has to schedule a map task over a tasktracker, it check if the tasktracker is located in the same cluster as the datanode having the data to process. If this condition is met then only task will be assigned to the tasktracker otherwise it will be delayed for certain delay in seconds specified for the delay scheduling.

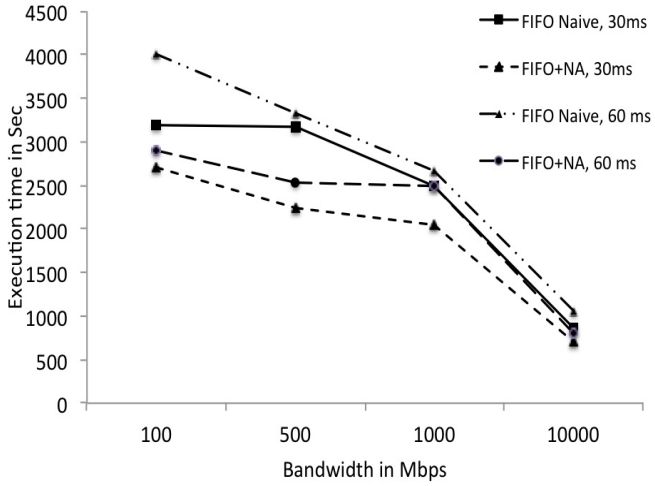


Figure 4: Execution times for varying inter-cloud bandwidth

Bandwidth is a significant factor for federated clusters. The performance of Hadoop greatly depends on the bandwidth between the clusters . Figure 4 is for varying inter-cloud bandwidth with fixed delay. We emulate the varying bandwidth cases on our experimental set up using dummynet [2]. At 100Mbps bandwidth between the clouds, there is significant difference in execution times for processing 10GB data. If the tasks are not executed either with node local or rack local then large amounts of data will be moved from one cluster to another cluster for map task execution. With 100Mbps of bandwidth there will be less available bandwidth between nodes. As the bandwidth provisioned between the clusters is increased overall execution time decreases. However, for processing tera bytes of data there should be sufficient bandwidth provisioned between the clusters. We performed experiments with different network conditions, in all the cases significant reduction in execution time is observed between native Hadoop scheduler and network-aware scheduler as the number of map tasks increased. It is evident that for higher inter cloud bandwidth the performance of Hadoop with network aware is greatly increased compared to native Hadoop scheduling.

Figure 5 shows percentage increase in the local map tasks. As explained in the earlier sections, Hadoop scheduling follows the philosophy of moving the computation to data. All the tasks which have the data on the same node as the one requesting for the tasks are scheduled first, this is referred as node local. If scheduler cannot find such tasks then all the tasks whose data is located in the same rack as the node requesting for the task are scheduled, this is referred as rack local. Network awareness is applied to non local map tasks which require to fetch data from some other data nodes. Network aware Hadoop minimizes the data movement from one cluster to another while executing map task by adding cluster level locality. We enable delay scheduling to optimize the data locality. Delay scheduling ensures that before the task is scheduled on a tasktracker which does not have the data to process will be skipped for configured amount of time. If any of the tasktracker becomes free in that dura-

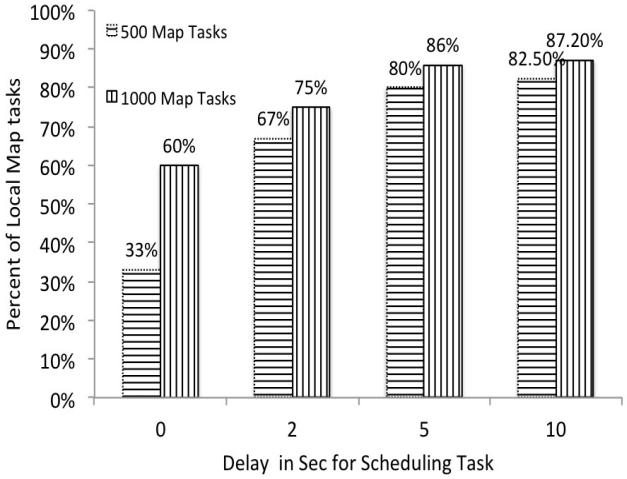


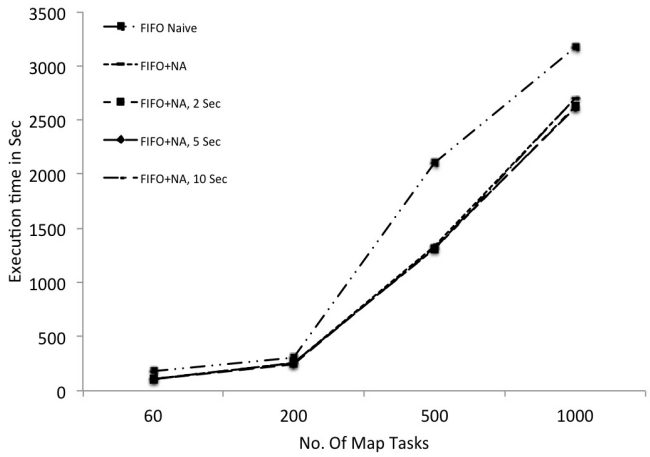
Figure 5: Increase in local map tasks with Network-aware

tion which as the data to process then, task is scheduled on the second tasktracker. For data intensive applications, data split movement takes more time than processing of the data split. Delay parameter is a sensitive parameter and should be carefully configured. In the Figure 5, delay of 0 sec refers scheduling without enabling the delay scheduler. With 2 Sec delay, non local map tasks will be delayed for 2 sec to check if any of the other tasktrackers request for a task. Since the length of each task is small, it is observed that tasktrackers get freed very frequently and request for a task. It is for this reason the delay parameter depends of the length of the task execution. These results show that network awareness coupled with delay scheduling could be used to minimize the transfer of the data between the clouds.

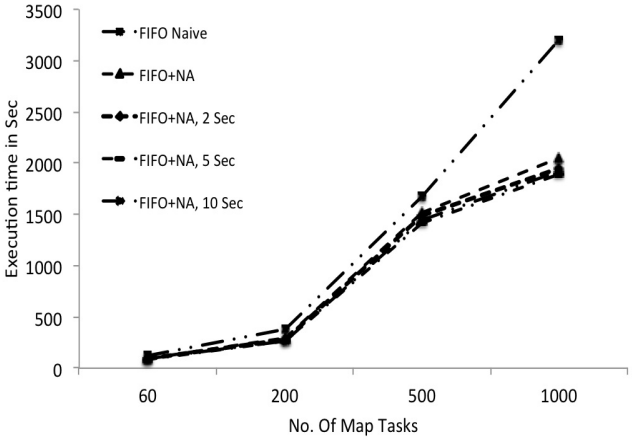
Figure 6 shows the results for 30ms inter cloud delay for varying bandwidths of 100Mbps and 1 Gbps. For the same number of map tasks, the execution times for a Job is less for 1Gbps bandwidth. In case of 100Mbps bandwidth the gap between curves for different scheduling schemes is less significant. Reduced available bandwidth between the clusters causes bottleneck for data split movement for tasks scheduled on distributed clusters. In figure 6(a), for tasks ranging from 500 to 1000 the execution times for native Hadoop and different variants for Network aware Hadoop, significant difference is observed though the inter cluster bandwidth is less.

Figure 6(b) shows experimental results for 1Gbps inter cloud bandwidth and 30 ms delay. Since higher bandwidth is provisioned between the clusters the overall execution times for the jobs compared to figure 6(a) is less. Further for jobs with higher number of tasks i.e tasks ranging from 500 to 1000 map tasks, the performance of network aware Hadoop is significantly improved. This shows that the performance Hadoop could be increased with provisioning higher bandwidth, less delay coupled with locality awareness. Similar results were observed for experiments conducted with fair scheduler.

Figure 7 shows 60ms inter cloud delay and bandwidths of 100 Mbps and 1 Gbps. There is significant reduction in the overall job execution times between 100 Mbps and 1 Gbps scenarios. However, figure 7(a) shows results for 100 Mbps



(a) 100Mbps bandwidth and 30ms delay between clusters



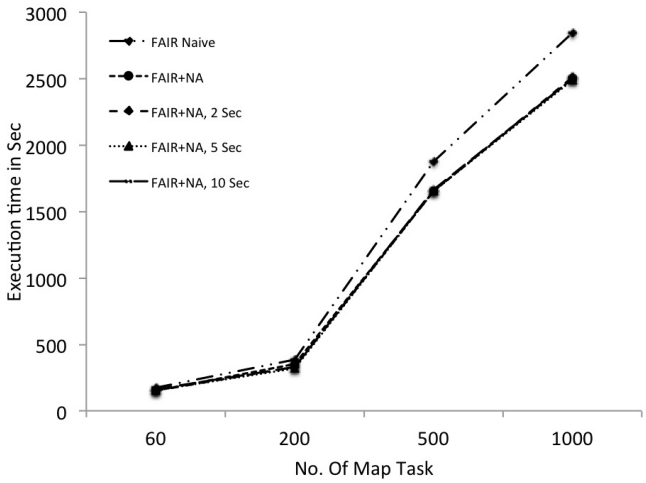
(b) 1Gbps bandwidth and 30ms delay between clusters

Figure 6: Execution times for 30ms inter-cloud delay

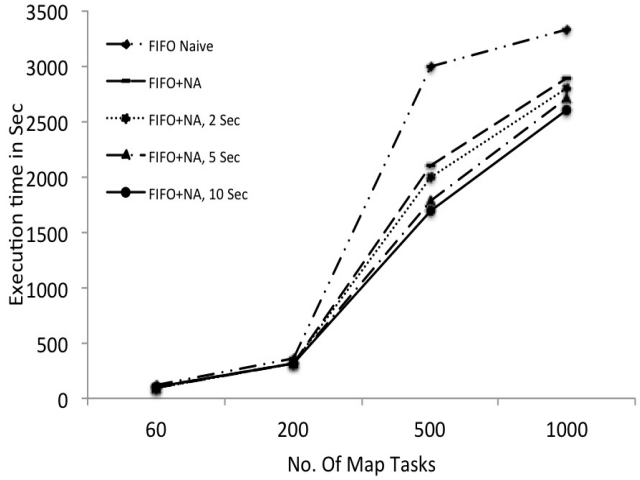
and 60 ms inter cloud network parameters. Because of the higher delay between the clusters, there is not significant improvement between the native Hadoop and the different variants of network aware Hadoop. Figure 7(b) is for 1 Gbps inter cloud bandwidth and 60 ms delay. In this case also because of higher delay between the federated clusters, less performance improvement is observed. However, it is better compared with the 100Mbps scenario.

5.1 Further Discussion

Figure 8 shows results for 10Gbps bandwidth and 30 ms delay between clusters. We observe that for higher bandwidth and low latency, overall execution times are greatly reduced for both native Hadoop and network aware Hadoop. However it is expected that network aware Hadoop to perform significantly better for higher bandwidth and low latency but significant difference is not seen from the results shown in the graph. The cause of this behavior is that the virtual interfaces of VM's cannot provide throughput more than 1.2 Gbps. Even though available bandwidth is 10Gbps,



(a) 100Mbps bandwidth and 60ms delay between clusters



(b) 1Gbps bandwidth and 60ms delay between clusters

Figure 7: Execution times for 60ms inter-cloud delay

the utilization from each of VM does not exceed 1.2 Gbps.

6. CONCLUSIONS AND FUTURE WORK

Placement of the map task on a node which has the data to process is critical for the performance of Hadoop over federated clusters. Single-cluster MapReduce architecture may not be suitable for situations when data and compute resources are widely distributed. In this work, we provide network awareness to the FIFO and FAIR schedulers in Hadoop. We evaluate our implementation on resources provided by CRON testbed. Performance improvement of 12 % to 15 % is observed in both FIFO and Fair schedulers. We plan to extend the network awareness while placing the reduce task since Reduce phase in MapReduce adds up significantly to the overall execution time.

Acknowledgement: This work has been supported in part by the NSF MRI Grant #0821741 (CRON project), GENI grant (BBN,NSF) and DEPSCoR project N0014-08-1-0856.

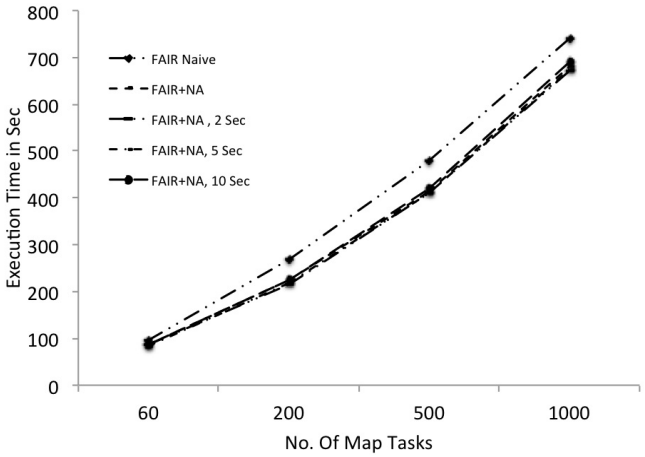


Figure 8: Execution times for 10Gbps and 30ms between clusters

7. REFERENCES

[1] Apache Hadoop. http://hadoop.apache.org/.

[2] Dummynet software emulator. http://info.iet.unipi.it/luigi/dummynet/.

[3] Eucalyptus Cloud Computing Software. http://www.eucalyptus.com/.

[4] Guture Grid Project. http://www.futuregrid.org/.

[5] Louisiana Optical Network Initiative. http://www.loni.org/.

[6] Neuca Eucalyptus Cloud Computing Software. https://geni-orca.renci.org/trac/wiki/Eucalyptus-2.0-Setup/.

[7] Openstack Cloud Computing Software. http://www.openstack.org/.

[8] XSEDE: Extreme Science and Engineering Deicovery Environment. http://www.futuregrid.org/.

[9] CRON Project: Cyberinfrastructure for Reconfigurable Optical Networking Environment, 2011. http://www.cron.loni.org/.

[10] M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman. Exploring mapreduce efficiency with highly-distributed data. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 27–34, New York, NY, USA, 2011. ACM.

[11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.

[12] Y. Luo, Z. Guo, Y. Sun, B. Plale, J. Qiu, and W. W. Li. A hierarchical framework for cross-domain mapreduce execution. In *Proceedings of the second international workshop on Emerging computational methods for the life sciences*, ECMLS '11, pages 15–22, New York, NY, USA, 2011. ACM.

[13] M. Zaharia, K. Elmeleegy, D. Borthakur, S. Shenker, J. S. Sarma, and I. Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *In Proc. EuroSys*, 2010.