Spanning Tree Method for Minimum Communication Costs In Grouped Virtual MapReduce Cluster

Yang Yang, Xiang Long, Biaobiao Shi
Computer College, Beihang University
Beijing
{yangyang,long,shibiaobiao}@les.buaa.edu.cn

Journal of Digital
Information Management

ABSTRACT: *Today, MapReduce and virtual cluster are sharp swords for this big data and cloud computing era. To combine these two emerging technologies, it brings feasible-scalability, easy-management, fast-deployment and high-efficiency with the system. As every sword has two sides, the I/O bottleneck of virtualization technologies may seriously impacts on the performance of MapReduce cluster which deals with I/O-intensive applications. In this paper, we analyze the combination advantages and disadvantages of virtualization technology of MapReduce cluster. We also analyze the communication model for both of them and build a communication costs model. Then, we propose a novel algorithm of minimum-weight spanning tree to construct a lower communication costs virtual MapReduce cluster. With the help of constructing minimum-weight spanning tree, we find out a method to select local-master and group the cluster. Theoretical simulation and experiment results demonstrate that our method can greatly reduce communication costs. The performance improvement is up to ~40.4% respectively*

Categories and Subject Descriptors
I.3.7 **[Three-Dimensional Graphics and Realism**]: Virtual Reality; H.5 **[INFORMATION INTERFACES AND PRESENTATION]** Hypertext Navigation and Maps

General Terms: Cloud Computing, Virtual Clusters

Keywords: MapReduce, Virtual Machine, Spanning Tree, Cloud Computing

Received: 27 January 2013, Revised 6 March 2013, Accepted 11 March 2013

1. Introduction

The interests in MapReduce program model resurgences in 2006, social network and multimedia develop at fantastic speed at that time. Unfortunately, the database, which is a good weapon for the structure data, cannot handle these huge numbers of unstructured and semi-structured data produced by these new applications well. When Jeffrey Dean [1] proposed MapReduce program model in distributed system, it receives a lot of attentions. Because of the remarkable features of MapReduce in simplicity, fault tolerance, and scalability, it is by far the most successful realization of dada intensive cloud computing platform [2]. Amazon, the famous web services provider, supplies Amazon EMR [3] which automatically spins up a Hadoop implementation of the MapReduce framework on Amazon EC2 instances which base on virtual machines [4].

Virtual machine(VM) plays a key role in cloud computing era. Virtualization technology "*slices*" a single physical machine into multiple VMs and each assigns virtual cores for their execution [5]. This technology is considered to be a foundation technology of cloud computing [6].

When MapReduce runs in a virtual environment, VMs bring the features of easy-deployed, highly-utilized and well-isolated. However, two major problems emerge.

The performance degradation caused by particular I/O framework of virtual machine system is a serious problem in virtual machine system [7]. Unfortunately, the situation becomes worse in virtual MapReduce cluster because of the I/O-intensive applications running on.

The master/slaves architecture of MapReduce would limit the performance with the increasing cluster scale. As Figure 1 shows, the more slaves are, the more packets send and receive. This relationship between the master and the connected slaves' number is nearly linear. It is obvious that the master node would under the stress if huge numbers of packets spring up in a short time. Unfortunately, it is exactly the situation in synchronistical MapReduce. Table 1 shows the results of network communications when the master is overload. It shows the communication speed has little effect but the packet loss has increased sharply.

	Normal Tests		Overwhelmed Tests	
	Avg. latency	Pkg.Loss. Ratio	Avg. latency	Pkg.Loss. Ratio
Inter-com	0.225	0%	0.227	0%
Intra-com	1.768	0%	1.772	74.52%

Table 1. The overwhelmed master effection

A favorable turn happens in the same corner. It was pointed out [1] that locality is an important issue affecting performance in a shared cluster environment, because of the limited network bisection bandwidth. To alleviate the I/O bottleneck in virtual MapReduce cluster, we construct a minimum-weight spanning tree to group the nodes with their communication conditions and data locality.

The rest of the paper is organized as follows. Section 2 gives a preliminaries introduction of MapReduce overview, I/O virtualization of Xen which is the most popular open-source Virtual Machine Monitor(VMM) and related work of spanning tree. Section 3 describes grouped virtual Mapreduce cluster architecture. In Section 4, we build a model and propose our algorithm in Section 5. We evaluate our implementation both in a sampling simulation and real experiments in Section 6. Section 7 discusses our evaluation results and some related issues.

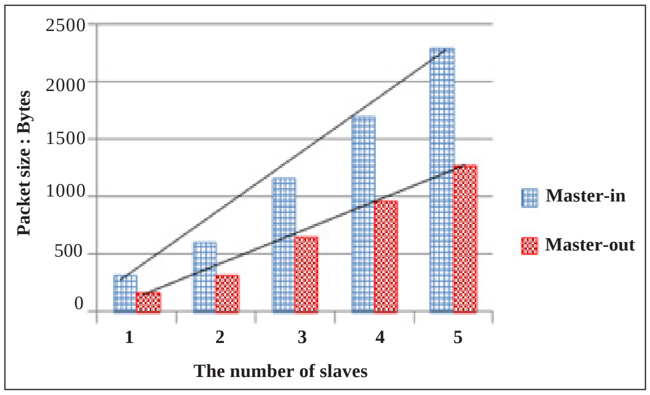


Figure 1.The linear relations between the number of slaves and packet size

2. Preliminaries

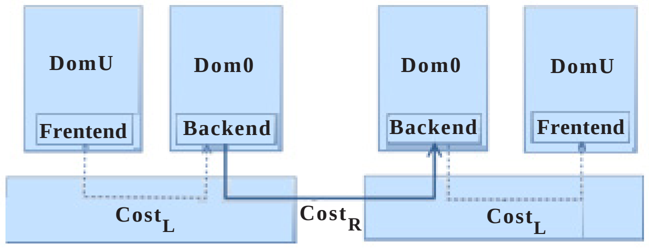


Figure 2. Communication cost schematic for I/O framework

2.1 MapReduce Overview

The data of MapReduce [1] is stored in <key, value>pairs and the computation proceeds are in rounds. Each round is split into map, shuffle and reduce consecutive phase. The architecture of MapReduce is typical master/slaves. The master node runs the services of NameNode and JobTracker. They have capacity for DataNodes and TaskTrackers management. Namenode is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. Datanode is stores data in the Hadoop Filesystems. The JobTracker is the service within Hadoop that farms out MapReduce tasks to specific nodes in the cluster and A TaskTracker is a node in the cluster that accepts tasks.

The communications in MapReduce working flow have two types:

• **Master-to-slaves:** At the beginning of the process, the master node assigns tasks to slaves and pings slaves to know the slaves status periodically during the working process.

• **Slave-to-master:** Slaves send their locations and intermediate file to the master

From this analysis, we know the most serious traffic jams between the master and slaves would happen at the time that intermediate data is produced and transferred after map finished.

Many studies pay attentions data locality to reduce the quality of transfer packets and improve network communications. Studies [2] [8] make Hadoop's reduce task scheduler aware of partitions' network locations and sizes to reduce the communication costs and execution time. [9] adjusts data locality dynamically according to network state and clusters workload.

All of these studies consider the deployment of MapReduce in physical cluster. [4] builds a model that defines metrics to analyze the data allocation problem. It also designs location-aware file block allocation strategy in virtual nodes which achieves better data redundancy and load balance to reduce I/O interference. Ibrahim et al. [10] evaluates MapReduce on virtual machines in Hadoop case. But they just discussed the differences of

MapReduce performance between virtual machines and physical machines. [11] explores the effect of I/O scheduling on performance of MapReduce running on VMs. [12] proposes a MapReduce framework on virtual machine which take full advantage of data locality, virtual machine live migration and checkpoint.

However, none of researches have considered reducing the communication costs produced by the overload master, by off-loads the work the master with local-master and take fully use of data locality of VMs in the same physical machine is predictable idea.

2.2 Xen I/O Framework

At the beginning of Xen, the device driver is provided to the domainUs by VMM [13]. The current version of Xen employs the domain0, which conducts real I/O operations on behalf of domainUs, to enhance the reliability and safety of the system [7]. In this model, a virtual frontend driver in a domainU communicates with a corresponding virtual backend driver residing in the domain0 and forwards I/O requests to physical device driver. The notifying between the drivers is realized by event channel mechanism which is similar with hardware interrupt in physical machine. Figure 2 shows communication costs in Xen. Every communication cost between domainU and domain0 in the same physical machine is (Cost)_L , communication cost between domainU and domain0 in different physical machine is (Cost)_R.

Besides the event channel, I/O ring is a shared memory region used by frontend driver and backend driver. Each time domainU sends/receives packets, it writes/reads to/from the I/O ring. The procedure of I/O working flow shows that every domainUs' access with physical device should be passed through domain0.

The recent study [11] on increasing the performance of virtual MapReduce cluster is to enhance domain0's weight and select physical machine as Master node. These two strategies both reduce the waiting time of I/O.

However, none of research stands on the point that reducing waiting time of I/O bottleneck by reducing data size on virtualized environment.

2.3 Minimum-Weight Spanning Tree

Minimum-weight spanning tree [14] is a algorithm for computing minimum spanning trees. It defines least-weight way of connecting all of vertices together with their edge weight. The basic idea of Minimum-weight Spanning is that the weight $w(u, v)$ is used to specify the costs connected u and v and an acyclic subset tree is found that connects all of the vertices and the total weight is minimized. This algorithm is used to design of computer and communication networks, telephone networks etc. It's useful way to help us to find out the topology of minimum Communication costs.

3. Architecture Of Virtual Mapreduce Cluster

Hadoop [16] is one of the most commonly-applied MapReduce implementation. The basic architecture of MapReduce consists of one master and many workers. Figure 3 is the common virtualized Hadoop architecture [4]. Figure 4 is two types of grouped virtualized Hadoop architecture. The difference between Figure 4a) and Figure 4b) is the location of local-master.

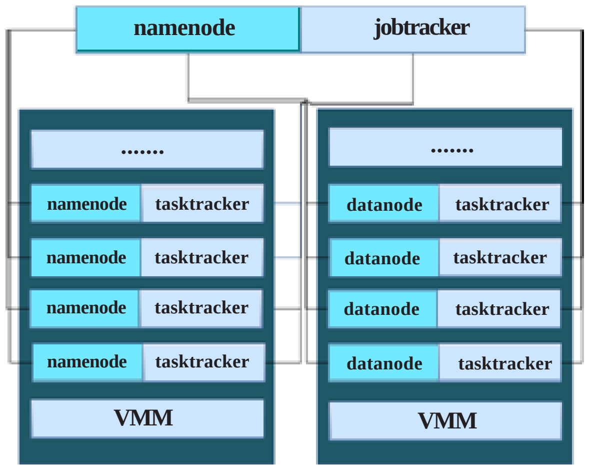


Figure 3. Original virtual MapReduce architecture

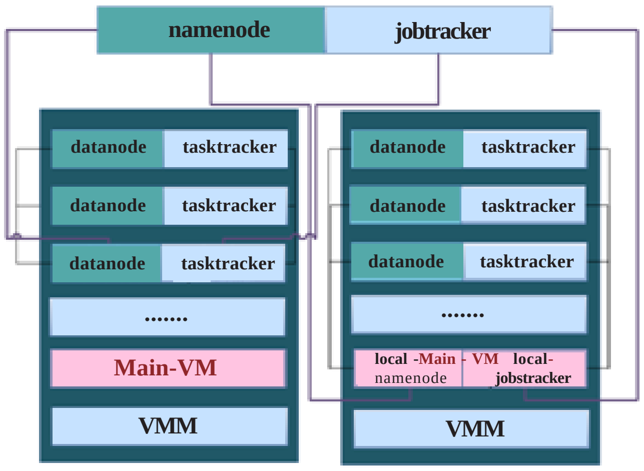


Figure 4. Grouped virtual MapReduce architecture

The master of original virtualized Hadoop architecture has responsibility for the management and file maintenance of slaves which is made of DataNodes and TaskTrackers. Both of Figure 4a) and Figure 4b) has a local-master which is made of local-NameNode and local-JobTracker. The local-master is a group master. The duty of local-masters which have the ability of local-NameNode and local-JobTracker are simplified original master. The local-NameNode is the master of local DataNodes and the local JobTracker is the master of local TaskTrackers. The local-JobTracker only receives the local-TaskTrackers heartbeats, updates and monitors task status. If JobTracker find out that the local-JobTracker is dead or local-JobTracker sends the group status report which shows all the status of the grouped TaskTrackers are dead, JobTracker would put this group into waiting-for-restart queue until the physical machine has been reboot and re-added into cluster. The local-NameNode stores the information of filename and block locations to take advantages of data locality.The realized protocols are Local Client Protocol and Local Data node Protocol communication protocols. The Local Client Protocol defines local namespace operations, including the operations of adding, creating, deleting blocks and getting block locations. The Local Data node Protocol defines the interface between local-NameNode and NameNode.

To minimum the communication costs, we build a minimum spanning tree to group these slaves.

4. Model

We model a Group-Independent Spanning Tree (GIST) in virtual Hadoop cluster as a graph $G = (V, E)$, where V is the set of nodes, and E is a set of edges on V. A tree in G is a connected subgraph $T = (V', E')$ containing no cycles. If $V'=V$, then T is a spanning tree for the graph G. Given a "*weight*" function w: $E \rightarrow Z$, a spanning tree T * is a minimum weight spanning tree if $\sum_{e \in T^*} w_e \leq \sum_{e \in T} w_e, \forall T$: T spanning tree of G. w_T denotes the weight of the tree T.

There are four types of communication costs in this framework. As w_1 represents inter-communication costs between domainUs in the different physical machines, w_2 represents intra-communication costs between domainUs in the same physical machine, w_3 represents communication costs between the domain0 and domainU in the same physical machine and w_4 represents communication costs between the domain0 and domainU in the different physical machine.

As section 2 analyses, $Cost_L$ and $Cost_R$ are used to indicate the edge weight in T.

$$w_1 = 2 * Cost_L + Cost_R \qquad (1)$$

$$w_2 = 2 * Cost_L \qquad (2)$$

$$w_3 = Cost_L \qquad (3)$$

$$w_4 = Cost_L + Cost_R \qquad (4)$$

Figure 5 shows the experiments results of packets transmission in these four weight.

The results show that $Cost_R$ costs much more than $Cost_L$. And the Costs are increased with the packets size raise. These results also verify (5) by experiments.

$$w_3 < w_2 < w_4 < w_1 \qquad (5)$$

Figure 6 shows the three graphs of virtual Hadoop cluster in Xen. Figure 6a) is the original cluster with all of the slaves connecting the master. Figure 6b) is the group-independent virtual Hadoop cluster which the local-master is a domainU and Figure 6c) is the grouped virtual Hadoop cluster which the local-master is the domain0. We construct GIST to find out the minimum communication costs of virtual MapReduce clusters.

5. Algorithm

5.1 Algorithm Describtion

We modified the classical minimum-weight spanning tree Prim algorithm [14] to construct our minimum-weight spanning tree in virtual MapReduce cluster.

```
MSTP-IN-GROUPED-VMC (G, W,M)
1 FOR each $u \in V[G]$
2   DO $key[u] \leftarrow \infty$
3      $\pi[u] \leftarrow NIL$
4 $key[m] \leftarrow 0$
5 $Q \leftarrow V[G]$
6 WHILE $Q \neq \emptyset$
7   DO $u$  $Extract\text{-}Min(Q)$
8       FOR each $v \in Adj[m]$
9           DO IF $v \in Q$ and $w(u, v) < key[v]$
10              THEN set $v$ as local-master
11                  FOR each $u \in Adj[v]$
12                   DO IF $u \in Q$ and $w(u, v) < key[v]$
13                       $\pi[u] \leftarrow u$
14                       $key[v] \leftarrow w(u, v)$
```

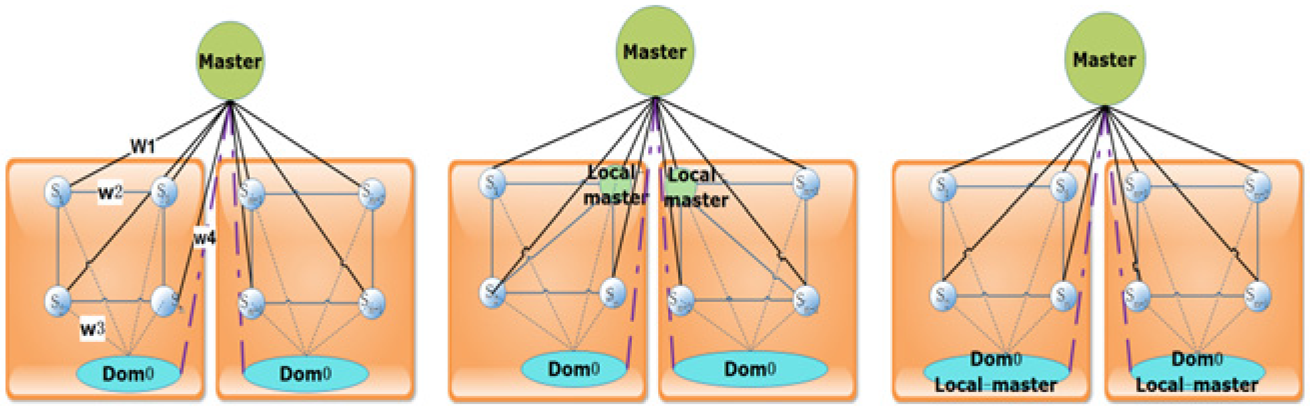
Lines 1-5 set the initial status of spanning tree. Q is the min-priority queue which contains all the vertices. The key of root vertex is set to 0 and key of the other vertices are set to ∞. Every parents of vertex are set to NIL and all of the vertices are contained by Q. Line 6 is entering the loop after judging whether the min-priority is empty. Line 7 identifies a vertex $u \in Q$ incident on a light edge crossing the $cut\,(V-Q, Q)$.

Lines 8-10 choose the local-master. After the local-master set, group should be formed. Lines 11-14 select the minimum edge to form minimum spanning tree and update the key ($key\,[v]$) and the parents fields ($\pi\,[u]$) of every vertex v adjacent to u but not in the tree.

5.2 Case study

Our grouped virtual MapReduce cluster has two different structures as Figure 4 show: Domain0 exclude and Domain0 included.

The tree starts from the master node which is the root of V. According to (1)~(4), the vertex adjacent to the master with w_3 should be added. As Previous study [11] points out that selecting physical machine as master gets better performance, we put the master node in a separated physical machine. That means no adjacent vertex with and the adjacent vertex with w_2 exists.

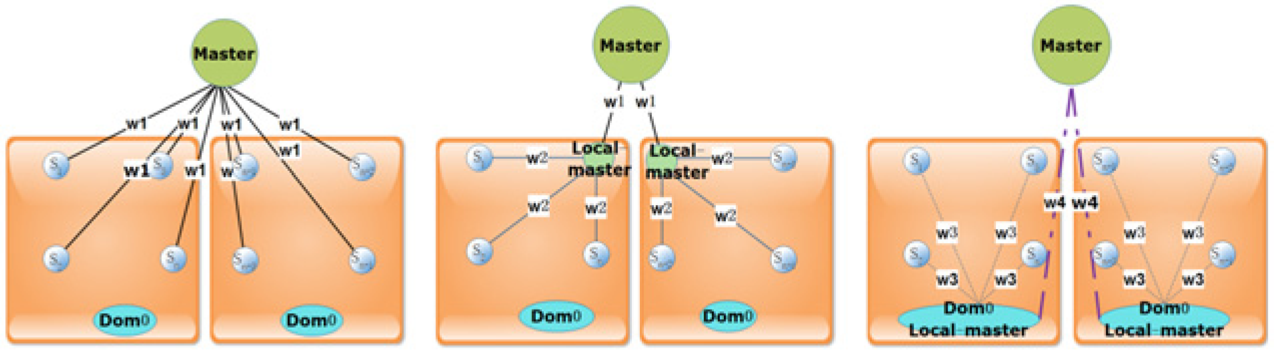


a) The connected graph of Original virtual MapReduce cluster

b) The connected graph of Grouped virtual MapReduce cluster with domain0 excluded

c) The connected graph of Grouped virtual MapReduce cluster with domain0 included

Figure 6. The connected graph of virtual MapReduce cluster



a) The mininum-weight spanning tree of Original virtual MapReduce cluster

b) The mininum-weight spanning tree of Original virtual MapReduce cluster with domain0 excluded

c) The mininum-weight spanning tree of Original virtual MapReduce cluster with domain0 included

Figure 7. The minimum-weight spanning tree of virtual MapReduce cluster

In domain0 exclude situation, we choose a domainU as new added vertex with w_1. The first chosen vertex is set as local-master. Then, scanning the adjacent vertices of local-master, the minimum weight edge w_2 is add to the $key[v]$, the vertices connected local-master with w_2 edge are added to $\pi[v]$.

In domain0 include situation, the minimum weight vertex adjacent to the master is domain0. The weight of the new vertex edge is w_4. The first chosen vertex is set as local-master. Then, scanning the adjacent vertecies of local-master, the minimum weight edge w_3 is add to the key[v], the vertices connected local-master with w_2 edge are added to $\pi[v]$.

The spanning tree results of these cases show as Figure 7b) and Figure 7c). We would have performance analysis in next section.

6. Evaluation

Assume that every virtual MapReduce cluster is made of n virtual machines (not including the number of domain0) and have g groups with g local-master.

Under the situation of original MapReduce cluster as Figure 8a) shows, the weight of tree T shows as:

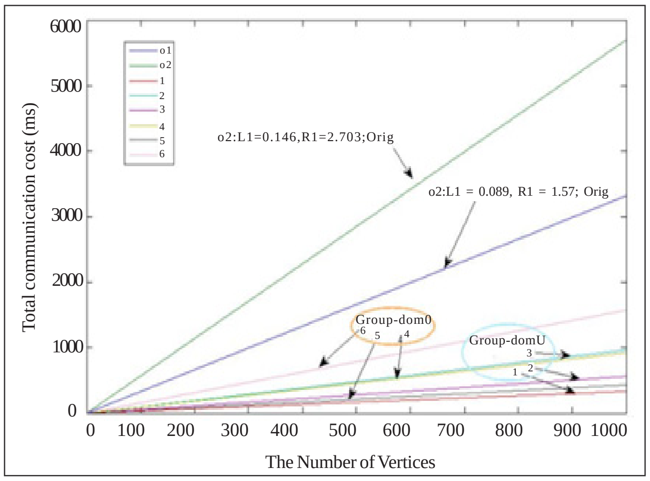


Figure 8. Simulation results for communication costs

$$w_{T_O} = n * w_1 \quad (6)$$

Under the situation of group-independent MapReduce cluster as Figure 7b) shows, the weight of tree T shows:

$$w_{T_{G1}} = n * w_2 + g * (w_1 - w_2) \quad (7)$$

Under the situation of group-independent MapReduce cluster as Figure 7c) shows, the weight of tree T shows:

$$w_{T_{G2}} = n * w_3 + g * w_4 \quad (8)$$

Taken these (1) ~ (4) and (6) ~ (8) together, we get new equations (8) ~ (10):

$$w_{T_O} = 2 * n * Cost_L + n * Cost_R \qquad (9)$$

$$w_{T_{G2}} = 2 * n * Cost_L + g * Cost_R \qquad (10)$$

$$w_{T_{G2}} = (n + g) * Cost_L + g * Cost_R \qquad (11)$$

Figure 8 shows the communication costs simulation results of the three structures of virtual MapReduce cluster. Each line represents the weight of the T with specific configuration. Table 2 shows the configuration of simulation experiments results in Figure 8.

Line-ID	o1	o2	1	2	3	4	5	6
Topo.	T_o	T_o	T_{g1}	T_{g1}	T_{g1}	T_{g2}	T_{g2}	T_{g2}
$Cost_L$	L2	L1	L1	L1	L2	L1	L2	L2
$Cost_L$	R2	L1	L1	L1	R2	L1	R2	R2
g	-	-	0.1n	0.5n	0.1	0.5n	0.1n	0.5n

Table 2. The parameter in the simulation

Variable *Topo*. in Table 2 means the architecture of three types of virtual mapreduce cluster which discussed in Section 2. In order to measure the value of $Cost_L$ and $Cost_R$ affects, we have different combination of these variables. The value of g in Table 2 means the ratio of group/slaves which decide the number of computation nodes in a group.

The structure types of O, $g1$ and $g2$ in Table 2 represent the three structure of Original, Grouped with domain0 and Grouped without domain0 virtual MapReduce cluster. The value of L and R are getting from the experiment results as Figure 5 show. The $L1$ and $R1$ are the value of and with the 0.5kB packets' size and the $L2$ and $R2$ are that with the 10kB packets' size. From the Equation (9)~(11), if the scale of the virtual MapReduce cluster n has been specified, the weight of T is proportional to g, which is less than 0.5 times of n (Because no one would specify local-master in two vertices group). So we Specify $g2 = 0.5 * n$ with the comparison group which $g1 = 0.1 * n$.

The $o1$ and $o2$ lines costs much more than the others, which mean the original structure of the virtual MapReduce clusters are much higher than both of our grouped virtual MapReduce cluster.

The circle including lines 4, 5 and 6 shows the weight comparison in domain0 included structure. And the circle including lines 1, 2 and 3 shows the weight comparison in domain0 excluded structure. It obvious that the bigger g induce higher communication costs since Line 2 is above Line 1 and Line 6 is above Line 5. In Figure 8, the phenomenon that Line 2 is above Line 1, Line 6 is above Line 4 and Line o1 is above Line o2 help us conclude that the larger value of $<L, R>$ pairs are, the higher communication costs are. An easy-ignored detail, which Line 4 is above Line 3, tells that the factor of $<L, R>$ pairs plays a decisive role in all of the impacts we have discussed.

Simulation results are verified an ideal model. Figure 9 and Figure 10 shows the real experiments results. We record the execution time of put and get operations in Hadoop Distribute File System (HDFS) with fixed size.

The real tests we have are with the hardware configuration of our test systems are physical machines which is equipped with Intel Q9400 quad-core CPU, DDRII-800 2GB memory. The software of Hadoop 0.20.2 cluster configurations are 2000MB Hadoop heapsize, 3 replications of files, maximum of map or reduce number on the same slot is 2 and speculative execution is on. The virtual node configurations are 256M memory, 1 vcpu, unpinned.

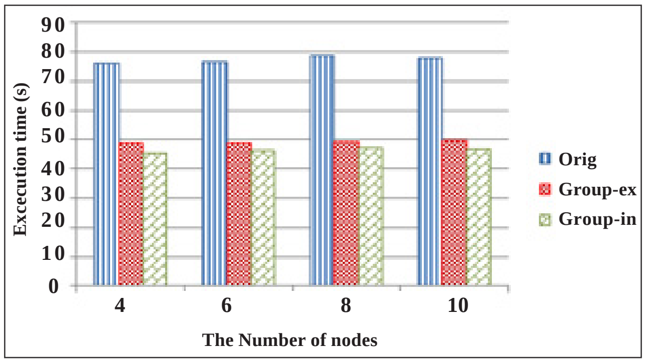


Figure 9. Execution time of Hadoop put operations

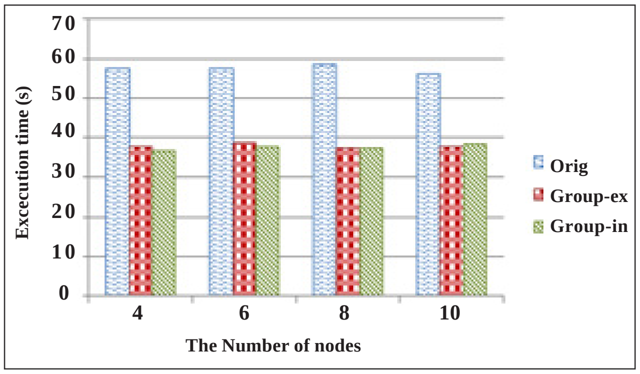


Figure 10. Execution time of Hadoop get operations

From these two figures, the original MapReducce clusters have the longest execution time. The virtual mapreduce clusters exclude dom0 have a little better performance than that of the virtual mapreduce cluster include dom0. This is because domain0 is also the bottleneck in virtualized system, too more workloads in domain0 may cause the overload of domain0 and reduce the performance.

The results tells that the performance improvements of our grouped virtual mapreduce cluster is up to ~40.4% in put operation and 36.3% in get operation. There is no explicit between the number of nodes and performance because that every physical machine has its fixed bandwidth.

7. Conclusion and Future work

In this work, we propose a novel minimum-weight spanning tree algorithm in grouped virtual MapReduce clusters which fully take advantages of virtual machines' data locality in the same physical machine and reduce the communication costs. The minimum-weight spanning tree algorithm help us construct an efficient grouped virtual MapReduce cluster which greatly reduce the I/O waiting time of file read and write. The improvement would benefit the whole performance of MapReduce applications.

The next step of us is that considering the node fault costs which frequently happen. If any fault nodes happen, it slows progress rate down seriously. Tasktracker failure would cause re-computing and speculative execution and Jobtracker failure would lead to system dump which is disaster for mapreduce clusters. We would pay more attentions on fault costs in future. We should comprehensive take communication cost and node invalid into consideration.

Refrences

[1] Jeffrey Dean, Sanjay Ghemawat. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1, 107-113, Jan.

[2] Ibrahim, S., Hai Jin, Lu Lu, Song Wu, Bingsheng He, Li Qi, LEEN. (2010). Locality/Fairness-Aware Key Partitioning for MapReduce in the Cloud, Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on , 17-24, Nov. 30-Dec. 3.

[3] http://aws.amazon.com/elasticmapreduce/.

[4] Yifeng Geng, Shimin Chen, YongWei Wu, Wu, R., Guangwen Yang, Weimin Zheng. (2011). Location-Aware MapReduce in Virtual Cloud, Parallel Processing (ICPP), International Conference on, p.275-284, 13-16 Sept.

[5] Cong Xu, Sahan Gamage, Pawan N. Rao, Ardalan Kangarlou, Ramana Rao Kompella, Dongyan Xu. (2012). vSlicer: latency-aware virtual machine scheduling via differentiated-frequency CPU slicing. *In*: Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing (HPDC '12). ACM, New York, NY, USA, 3-14.

[6] Jin, Hai. (2010). From Grid Computing to Cloud Computing: Experiences on Virtualization Technology. Future Generation Information Technology: Second International Conference, Lecture Notes in Computer Science,V. 6485/2010, 41, FGIT, Jeju Island, Korea, December, 13-15, Proceedings.

[7] Yanyan Hu, Xiang Long, Jiong Zhang, Jun He, Li Xia. (2010). I/O scheduling model of virtual machine based on multi-core dynamic partitioning. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10). ACM, New York, NY, USA, 142-154.

[8] Hammoud, M., Rehman, M. S., Sakr, M. F. (2012). Center-of-Gravity Reduce Task Scheduling to Lower MapReduce Network Traffic. Cloud.

[9] Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong, Runqun Xiong. (2011). BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing, Cluster, Cloud and Grid Computing (CCGrid), 11th IEEE/ACM International Symposium on, 295-304, 23-26 May.

[10] Ibrahim, S., Jin, H., Lu, L., Qi, L., Wu, S., Shi, N. (2009). Evaluating MapReduce on Virtual Machines: The Hadoop Case, Proc. Conf. Cloud Computing (CloudCom 2009), Springer LNCS, Dec, p.519-528.

[11] Jun Fang, Shoubao Yang, Wenyu Zhou, Hu Song. (2010). Evaluating I/O Scheduler in Virtual Machines for Mapreduce application, Grid and Cooperative Computing (GCC), 9th International Conference on, p.64-69, 1-5 Nov.

[12] Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, Li Qi. (2009). CLOUDLET: towards mapreduce implementation on virtual machines. *In*: Proceedings of the 18th ACM international symposium on High performance distributed computing (HPDC '09). ACM, New York, NY, USA, 65-66.

[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. (2003). Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). ACM, New York, NY, USA, 164-177.

[14] Cormen, T. H., Leiserson, C. E., Rivest, R. L. (1989). Introduction to Algorithms, MIT Press. 561-579.

[16] http://hadoop.apache.org/common/.