K-Means Method for Grouping in Hybrid MapReduce Cluster

Yang Yang, Xiang Long, Bo Jiang†
School of Computer Science and Engineering
Beihang University
Beijing, China
{yangyang, long, bojiang }@les.buaa.edu.cn

***Abstract*—In hybrid cloud computing era, hybrid clusters which are made of virtual machines and physical machines would be seen more and more generally. Hybrid clusters need more careful organization for finer resource allocations. Another problem of big data in this era is that database system can not well-handled the semi-structured and unstructured data. Luckily, MapReduce is a good weapon to solve the increasing big size and quickly-increased data at this social computing and multimedia computing time. One of the biggest challenges in hybrid mapreduce cluster is I/O bottleneck which would be aggravated under big data computing. In this paper, we take data locality into consideration and group slave nodes with low intra-communication and high inter-communication. After introducing the architecture and implementation of our grouped hybrid mapreduce cluster(GHMC), we give our method of k-means algorithm to group in our GHMC system and evaluate it with reality environments. The results show that there is a nearly 34.9% performance improvement in our GHMC system which are deployed by our K-means algorithm. What's more, it also shows good scalability.**

***Index Terms*—MapReduce; Hybrid cloud computing; K-means algorithm.**

I. INTRODUCTION

MapReduce program model is emerging technology resurgence in 2006[1]. With the development of social network and multimedia, the size of unstructured data and semi-structured data increase at fantastic speed. There are remarkable features in MapReduce, such as simplicity, easy and efficient fault tolerance, and good scalability. It is by far the most successful realization of data intensive cloud computing platform[2]. Since Amazon EC2 has spanned up Hadoop framework on Amazon EC2 instances supplying Elastic MapReduce services[3], MapReduce program model tightly combines with cloud computing and virtualization technology.

In cloud computing era, virtualization technology[5] become a foundation technology with the features of easy-deployed, highly-utilized and well-isolated[6]. In Gartner' report, the trends of hybrid cloud computing would keep in mainstream in next 5 to 10 years. It means more and more applications may deploy in hybrid cluster consisted by virtual machine(VM) and physical machine(PM). In hybrid mapreduce cluster, virtual machines, which "slice" a single physical machine to multiple relatively separated VMs and are assigned own resources, co-exist with physical machines.

The performance of a large scale of hybrid mapreduce cluster may degrade because of I/O bottleneck[7]. The integration of mapreduce and hybrid cloud computing have worse performance of I/O. The reasons are below:

- MapReduce applications have clear bounds among the processing stages. In I/O-intensive stages, large numbers of I/O requests are sent and data transfer performance degrades sharply.
- The architecture of MapReduce is master/slave which limits the I/O performance on the space. All of the slaves communicate with the master. It means that the more slaves are, the more packets would be dealt with. The performance of the master node reduces when it overload[17].

It was pointed out[1] that locality is an important issue affecting performance in a shared clusters environment, due to limited network bisection bandwidth. The current researches[2][4][8][11][12][17] on reduce I/O communication costs are take use of data locality. Grouped Virtual Mapreduce Cluster(GVMC)[17] group slave nodes and select local-masters which manage the nodes in every group. However, all of the researches are based on homogeneous mapreduce clusters.

In hybrid mapreduce cluster, we take use of the way which groups low intra-communication nodes together and select local master. We name this hybrid mapreduce cluster as grouped hybrid mapreduce cluster(GHMC). GVMC is one part of GHMC. Using the minimum-weight spanning tree method to construct GVMC minimum communication tree is not suitable for our GHMC because of the more complex situations.

To take fully use of data locality and alleviate the I/O bottleneck in hybrid mapreduce cluster, we make use of clustering method to group the slave nodes with low intra-communication and high inter-communication. Considering the characteristic of hybrid mapreduce cluster, the advantage of clustering method is that the

This research is supported in part by the grants of the National Science and Technology Support Project of China (project no. 2011BAH04B05), National Natural Science Foundation of China (project no. 61202077) and the Fundamental Research Funds for the Central Universities (project no. YWF-12-LXGY-008).

† Corresponding author.

doi:10.4304/jcp.8.10.2648-2655

scale of the problem is relatively small and has low-dimension.

The rest of the paper is organized as follows. Section II is related work. Section III gives a introduction of grouped hybrid mapreduce cluster(GHMC) overview and formal the question. In section IV, we propose our K-means Method Algorithm for Clustering. We verify our algorithm in real GHMC in Section V. Section VI discusses our evaluation results and some related issues.

II. REALTED WORK

A. MapReduce Overview

MapReduce is a parallel program model. The input data stores in <key, value>. The data is split into *map*, *shuffle* and *reduce* consecutive phase. The most popular open-source implementation of MapReduce is Hadoop[18] which is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Hadoop follows the architecture of Google MapReduce[1]. The basic MapReduce framework consists by HDFS and MapReduce. HDFS is a distributed file system supplying high-throughput access to application data which makes of Namenode and Datanodes. Jobtracker has responsibilities for MapReduce task scheduling and tasktrackers management. This typical master/slaves architecture aggravates I/O bottleneck. There are two types of communications in MapReduce working flow:

- 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

Study[11] explores the effect of I/O scheduling on performance of MapReduce running on VMs. Study[19] use weighted round-robin scheduling algorithm into the task scheduling of Hadoop and puts forward the weight update rules through analyzing all the situations of weight update.

All of these works take use of data locality and reduce the communication costs in either pure physical mapreduce cluster or pure virtual mapreduce cluster. These researches on their mapreduce cluster are based on the assumption that mapreduce cluster are in homogeneous environments. It's obviously that mapreduce clusters are in heterogeneous environments when they deploy physical and virtual hybrid machines. Making use of the idea of GVMC, we group the slave nodes with low intra-communication costs and high inter-communication costs.

B. Data locality method overview

In order to reduce the communication costs and alleviate I/O performance degradation, data locality is fully used. Studies[2][8] make Hadoop's reduce task scheduler aware of partitions' network locations and sizes to reduce the communication costs and execution time. BAR[9] allocates tasks dynamically according to network state and clusters workload. Studies[4][11][12] take VMs data locality to improve the performance. Study[4] has builds a model that defines metrics to analyze the data allocation problem. Study[12] proposes a MapReduce framework on virtual machine which take full advantage of data locality, virtual machine live migration and checkpoint.

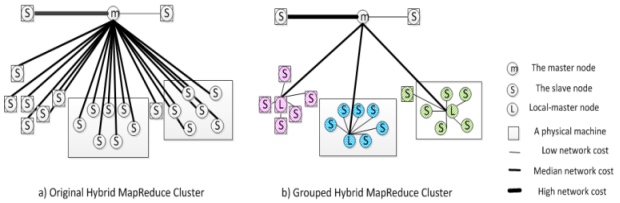

Figure 1. A case of Original mapreduce cluster and GHMC

C. Clustering Algorithm Overview

Cluster[28] is the process to collect similar data objects to one another within the same cluster and dissimilar to the objects in other clusters. The popular methods for clustering are classified by partitioning methods, hierarchical methods, density-based methods and grid-based methods.

K-means algorithms[29] is one the most popular algorithms. It's a partitioning method to construct a partition of a database D of n objects into a set of k cluster[20]. It's efficient algorithm for clustering. The process is implemented in 4 steps: firstly, it partition objects into k subsets; secondly, k seed points as the centroids of the cluster; thirdly, each object is assigned to the cluster with the nearest seed point. The process lasts until no more new assignment happen.

III. DESIGN OF GHMC

A. Grouped Hybrid Mapreduce Cluster

At the original mapreduce cluster, every slave connects the master nodes and sends/receives messages to/from it. We take advantages of data locality; group the low intra-communication slave nodes to reduce data transfer costs in mapreduce process. Local-masters are also introduced to manage group members.

Figure 1 shows a case of GHMC topology. Figure 1.a) shows original hybrid mapreduce cluster. b) shows GHMC. The black line with HH Type means the high communication costs of the slave node when it connects with both the master and the other slaves. The finer line means the connection of HL Type slave and the master. The HL Type slaves represent the slave nodes with high-communication costs connecting the master and low-communication costs connecting other slaves. The finest line means the lowest communication costs with the master which we call LO type communication. Since HH type nodes are hard to optimal by grouping and LO type nodes have no need to optimal, we take the method of grouping to optimal on HL type slaves communication costs.

In GHMC, HH type nodes or LO type nodes are directly connected with the master. Others are connected with local-masters which have 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 LocalClientProtocol and LocalDatanodeProtocol communication protocols. The LocalClientProtocol defines local namespace operations, including the operations of adding, creating, deleting blocks and getting block locations. The LocalDatanodeProtocol defines the interface between local-NameNode and NameNode.

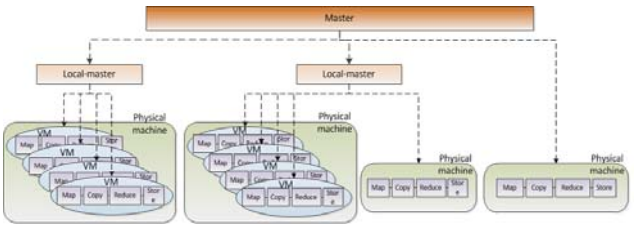

Figure 2. The Architecture of Group Hybrid Mapreduce Cluster

B. *Grouped Hybrid Mapreduce Cluster*

Figure 2 shows the architecture of GHMC. There are three types of organization for group:

- Pure virtual machine group: the nodes in this type of group are all virtual machines.
- Pure physical machine: the nodes in this type of group are all physical machine.
- Hybrid virtual machine and physical machine: the nodes in this group are made of virtual machine and physical machine.

In these three types of GHMC, there are three types of execution overview and local-master should have little different functions.

The original execution overview of mapreduce cluster in the virtual machines shows as Figure 3 where nodes deploy in virtual machines and no structure are optimal on it. Data are divided into many splits and every mapper process on a split, then, merge the intermediate data on every virtual machine, stores the results into local disk. Every reducer copies the intermediate data from virtual disk and go on reduce process.

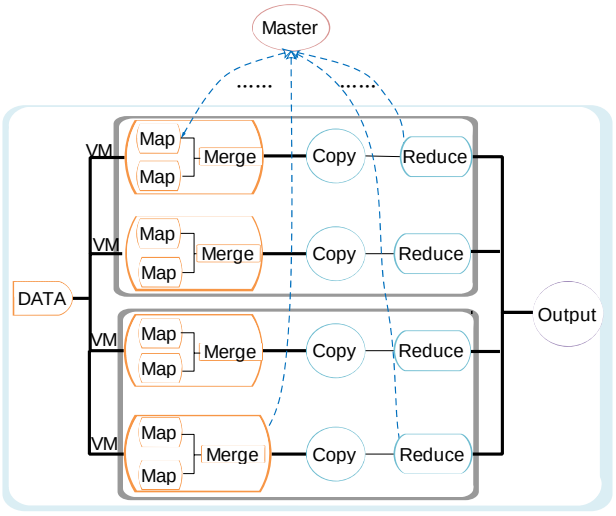

Figure 3. The execution overview of original Mapreduce Cluster

In pure virtual machine group, the merge step is merge all of the intermediate data in the same physical machine. Then, the *reduce* process of intermediate results is similar with the original one.

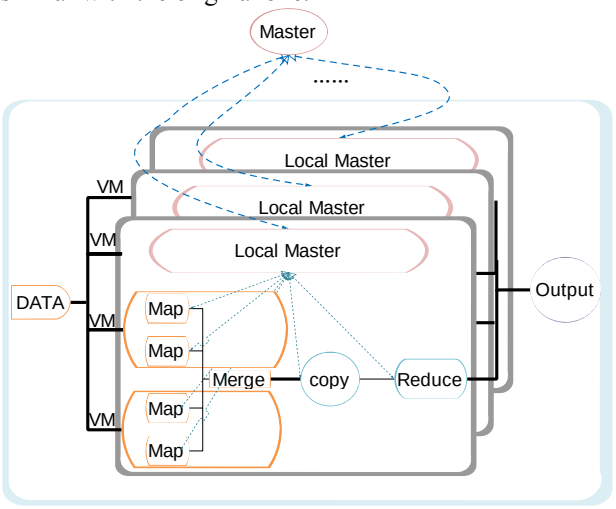

Figure 4. The execution overview of pure virtual Mapreduce Cluster

In pure physical machine group, the overview execution of this type is divided into 4 steps: the process is firstly *map* in every physical machine nodes, then *merge* the intermediate results as original mapreduce cluster, do *reduce* in group and the last step is that have *global reduce* in the whole systems.

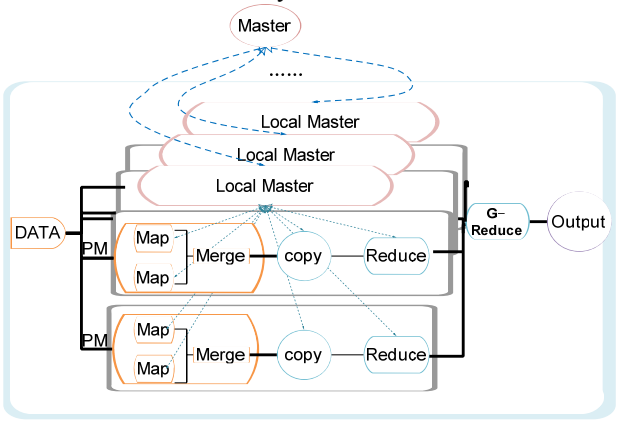

Figure 5. The execution overview of physical Mapreduce Cluster

In hybrid virtual machines and physical machines, the overview execution should do both the processes of pure physical machine groups and pure virtual machine groups. So the local-master should know the topologies of its group. And merge intermediate data have G-reduce in every physical machine.

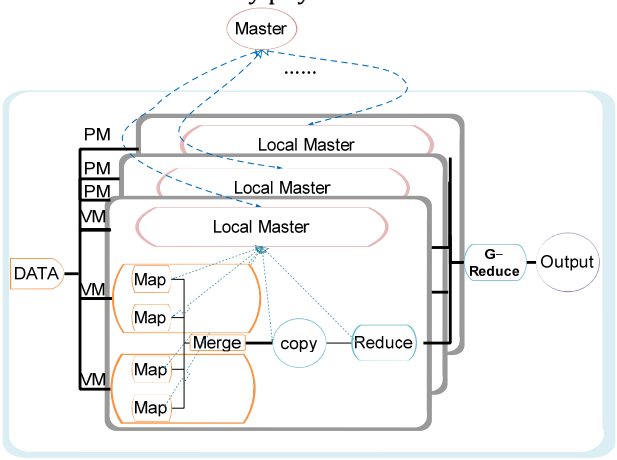

Figure 6. The execution overview of hybrid Mapreduce Cluster

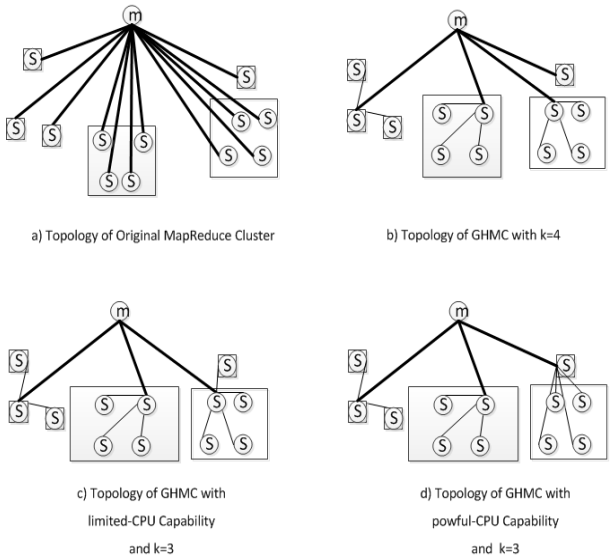

Figure 7. The Experiments Environments of Group Hybrid Mapreduce Cluster

In these structures, the most important implementation is the local-master. So, local-masters should know the information of slaves whether on nodes of physical machines or on nodes of virtual machines and control the execution flow of every type.

IV. K-MEANS METHOD ALGORITHM

A. Algorithm Describtion

In order to complete the operation of grouping and selecting the local-master, we need to build a Low and Flow Tree(LFTree). The conditions are satisfied that T is a connected , undirected graph $T = (V, E)$ where V is its vertex set which represents nodes and E is its edge set which stands for the connection between nodes. For each edge $(i, j) \in E$, we have weight $w(i, j)$ specifying the cost to connect i and j and put $w(i, j)$ into the matrix D_{ij} .

The input of the algorithm is D_{ij} matrices and the number of groups specified by the administrator k . The output is the LFTree built by our algorithm.

K-MEANS-IN-GROUPEDHMC(D_{ij}, k)

1: initial $n = length(D_{ij})$

2: initial $CQueue$ which stores the centroid V_x ;

3: initial $NCQueue$ which stores the non-centroid V_y ;

4: initial combination number of $l = C(n, k)$;

5: initial control variety of loop:

$$S_0 = S_1 = S_2 = tmp = 0$$

6: let $HLMatrix = w(i, j)$ which $i \in \{x \mid x \in HL\}$;

7: **while** $S_0 < l$

8: let $CQueue^{s0} = combination(length(HLMatrix), k)$;

9: let $V_i \in CQueue^{s0}$;

10: let $V_j \in NCQueue^{s0}$;

11: **while** $S_2 < length(CQueue^{s0})$

12: **while** $S_1 < length(NCQueue^{s0})$

13: let $V_o \in Group_p^{s0}$ when $D_{po}^{s0} = min(D_{ij})$;

14: let $S_1 = S_1 + 1$;

15: **end**;

16: let $S_2 = S_2 + 1$;

17: **end**;

18: let Weight$(T^{s0}) = \sum_{p=0}^{k-1}\sum_{o=k}^{n-1} D_{po}^{s0} + \sum_{i=0}^{k-1} D_{ij}^{s0}$;

19: record the β th minWeight(T^{s0});

20: record the group $situationGroup\_\beta_\alpha^{s0}$ where $a \in CQueue^{s0}$

21: let $S_0 = S_0 + 1$;

22: **end**;

23: Choose suitable centroid based on CPU cap;

24: Restore LFTREE;

We firstly initial the temporary variables. Lines 1-5 show this procedure. Then, choose the HL Type nodes by scanning MNC matrix and build HLMatrix consisted by edges' weight connecting HL type nodes. It shows in Line 6. Lines 7-22 are improved k-means clustering algorithm to find out the clusters and centroids which are groups

and local-master in our GHMC environment. Lines 23-24 find out the relatively optimal group method for the cluster. Line 8 is the combination function which produces l initial centroid way. Lines 7-22 have triple loop. The outermost loop is the l different initial centroid. The inner loops are the processes which group the non-centroid nodes into the minimum distance centroid group. Lines 18-20 are choose the βth minimum sums of weight in certain situation. Then, take use of group function to complete the process of group operation and produce LFTree.

V. Evaluation

A. Experiments Environment

The hardware configuration of our test systems are physical machines 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.

We use hybrid cluster to verify our GHMC performance. The cluster is made of 12 slave nodes and 1 master as Figure 7 shows. 8 slave nodes are put on 8 virtual machines which allocate on 2 physical machines equally. We use adjust the network bandwidth and simulate the different network condition in reality situation. The network bandwidth which is set 5MB/s in thick black lines and 200 MB/s thin lines in Figure 3. The network bandwidth of disconnected nodes is set to 10 KB/s. Figure 3 shows the results after running our GHMC algorithm and reconstructing the cluster. We also run Wordcount, Grep and BBP to measure the performance.

B. Experiments and Discussion

We have discussed 4 problems of pure virtual machines cluster and 2 problems of hybrid mapreduce cluster. The problems are list:

1) Virtual machine costs introduced by local-master.
2) The impact of VM number on performance.
3) Performance results under the no fault situation.
4) Performance results on different type of applications.
5) Performance results under the four different way of group.
6) Scalability of GHMC.

Let's look at these experiments results and analysis below:

1) How much does the local-master cost?

Table 1
CPU usage of Special Nodes

	Ovmc.mv	Gvmc1.lm	Gvmc1.mv	Gvmc2.lm
2	1.9%	0.3%	3.0%	3.4%
3	2.3%	0.3%	3.0%	3.4%
5	3.0%	0.3%	3.1%	3.9%
7	3.2%	0.4%	3.3%	4.2%
9	4.6%	0.5%	3.4%	4.8%

It's difficult to measure the local-master's working cost itself. The service of local-master may be affected by many factors. The approximate value of real cost to collect static value in our work.

Table 1 shows the CPU usage of special nodes in VMC when they linked with different number of slaves. The special nodes in our paper includes main-VM in OVMC(*OVMC.mv*), local-master in GVMC1(*GVMC1.lm*), main-VM in GVMC1(*GVMC1.mv*) and local-master in GVMC2(*GVMC2.lm*). Because the local-master is put on main-VM in GVMC2 framework, local-master in GMVC2 is also main-VM in GVMC2. The value of *OVMC.mv* is baseline of *GVMC1.mv* and *GVMC2.mv* because all of these special nodes should have the basic service working on OVMC. *GVMC1.lm* is the pure local-master costs compared to *GVMC2.lm* which also contains main-VM CPU usage costs.

The percent of CPU usage in Table 1 are collected in the same physical machine in VMC which means they have the same CPU computation capability. From Table 1, we can see that the GVMC1.lm keeps low percentage. And the difference between the values of OVMC.mv and GVMC2.lm, which means the local-master costs of GVMC2, keeps low too. The largest difference value between OVMC.mv and GVMC2.lm is 1.2%. So, we conclude that both GVMC1.lm and GVMC2 maintains low level CPU usage for local-master services.

2) How many VMs should be grouped?

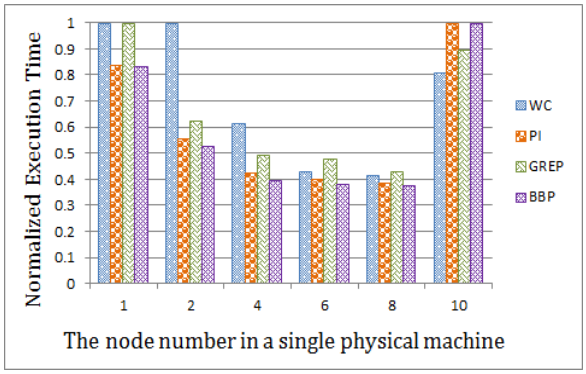

Figure 8. The number of the nodes effects on execution time

It's hard to measure the exact mathematical relations between the VMs' number and physical machines' number. Bin packing algorithm is often used to get the approximate solutions.

In our solution, there is a conflict on virtual node mapreduce architecture. Higher parallelism for applications means better performance, but resource costs for tasktracker are higher. The problem of VMs number is complex and is related to many factors. Luckily, we needn't get the exact optimal value of the VMs' number because the resource would be dynamic determined at the running time.

We test the performance with different VMs' number in single physical machine. Figure 8 shows the normalized results of MapReduce performance with different VMs' number which run wordcount, pi, grep and bbp. In GVMC1 virtual mapreduce clusters, when VMs number equals to 8, which is 2 times of CPU cores, all of the mapreduce applications perform the best. The results

are similar for both CPU-intensive applications and I/O-intensive applications. Based on these empirical data, we suggest that the VMs number can be more than CPU cores and less than 2 times of CPU cores. Because too little VM numbers result in resource waste and too much VM numbers lead to much switch costs. So, we suggest set VM_threshold equals to 2 times of CPU number.

3) Whose performance is better in no fault situation?

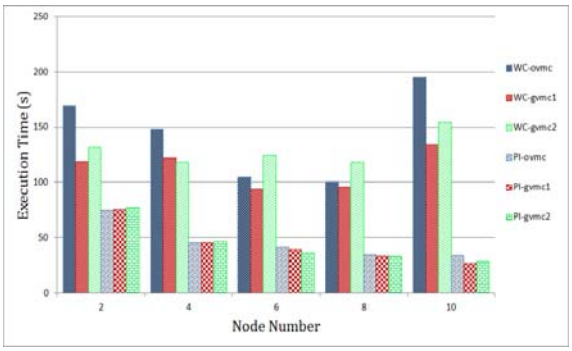

Figure 9. The Performance comparison under the no fault situation

Figure 9 compares the performance of three architecture when no node fault occurs. We observe the execution time on little-sized wordcount and pi, which represent I/O-intensive and CPU-intensive applications respectively. It's clear that I/O-intensive applications are more sensitive to the master locations. It tells us the communication costs are the main factor affects the performance of I/O-intensive applications. The execution time of OVMC, which is the original implementation of Hadoop, is more than that of GVMC1, but the execution time comparison between GVMC2 and GVMC1/OVMC shows stochastically. This is because that the double duty of local-master would cause fault and lead to unstable performance.

4) How do different types of applications perform?

TABLE 2
EXECUTION TIME AND MEAN-VARIANCE OF CPU-INTENSIVE APPLICATIONS

	BBP1	BBP2	BBP3	PI1	PI2	PI3
Mean	57.295	95.859	75.727	36.701	55.689	76.817
v.ovmc	3.71	2.425	1.259	1.126	0.955	2.310
v.gvmc1	0.112	0.502	0.656	0.306	0.021	0.507
v.gvmc2	3.603	2.927	1.915	1.432	0.977	1.803
Var.ovmc	62.55	99.29	77.51	38.29	57.04	80.08
Var.gvmc1	57.14	96.57	76.66	37.13	55.72	76.10
Var.gvmc2	52.20	91.72	73.02	34.68	54.31	74.27

The last tests explain the impacts on performance of I/O-intensive applications and CPU-intensive applications separately. All of the results have been filtered with no node faults happen. Figure 10 I/O-Intensive has compared the execution time of wordcount and grep applications, which stand for I/O-intensive applications in the three types of structures. The processed data is of three different sizes: 1G Bytes, 2G Bytes and 3G Bytes. The results tell us the larger the data is, the longer execution time it has. This is because more communication costs would spend longer time on the larger data transfer. The execution time of OVMC structure is higher than the others. That's to say, our algorithm would work faster than the original assignment of virtual mapreduce clusters. GVMC2 gets a little better result than that of GVMC1, this is for the communication costs. However, considering the single point of failure, we suggest deploying the virtual nodes in physical machine like GVMC1.

The Table time is the execution time of virtual mapreduce clusters, which run CPU-Intensive applications. Without statistics of the node faults situation, the GVMC2 has the best performance and the Table mean, which is the Mean-Variance table shows little difference in CPU-Intensive workload. From this table mean, all of the variances from GVMC1 and GVMC2 are small. It means that there is no big difference on CPU-Intensive applications.

From last experiments, our grouped virtual mapreduce clusters have achieved a performance improvement of up to 16.5% on CPU-intensive applications and 36.2% on I/O-intensive applications. The BBP, Pi, Wordcount and Grep performance have been improved by 10.0%, 7.2%, 24.9% and 14.0% on average, respectively.

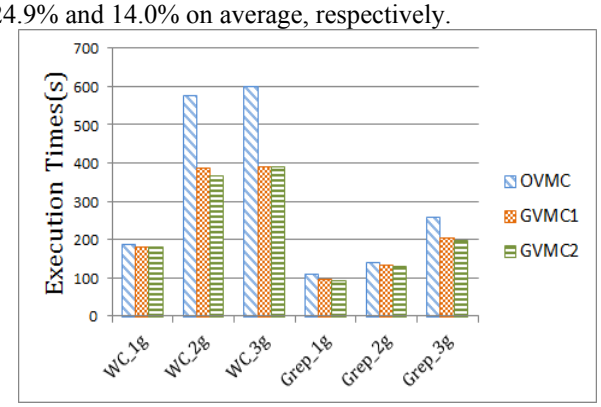

Figure 10. Performance comparison on I/O-intensive applications

5) Performance results under the four different way of group

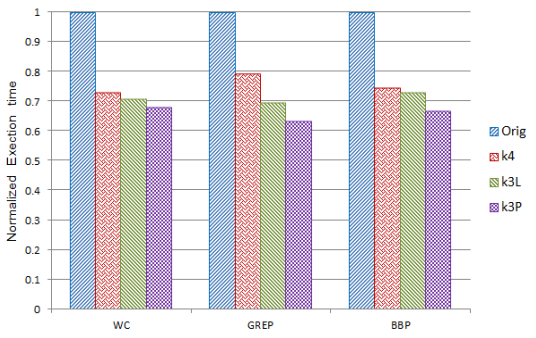

Figure 11. The Normalized Execution Time on four

Figure 11 shows the experiments results under the four environments as Figure 3 shows. To show more visually appealing way, we take the average value on 50 times execution time of each applications and normalize the last results. The results tell that the improvement of our grouped virtual mapreduce cluster is up to ~34.9%. It's obviously that set the powerful CPU machines as local-masters have shorter execution time than that of selection with weaker local-master. The parameter of *k* value also affects the execution time; it needs more study on further research.

6) Scalability of GHMC.

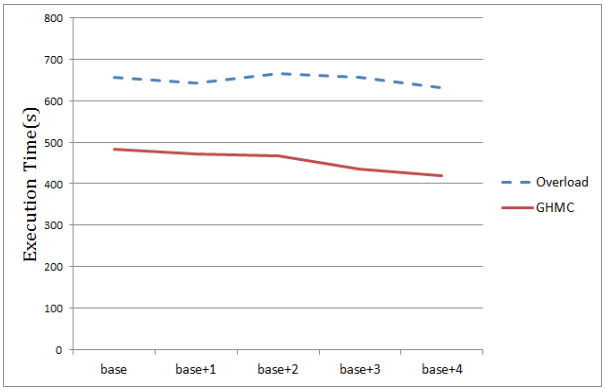

Figure 12. Scalability comparison on Overload original cluster

In GHMC scalability tests, we add four physical machines and use GHMC algorithm to compute the topology each time. Figure 12 shows the results. Comparing the instability of overloaded original mapreduce cluster, the execution time of GHMC is shorter when the nodes number increased. It says that our GHMC algorithm has good scalability.

VI. Conclusion and Future Work

In this work, we propose a novel method to group the hybrid mapreduce cluster which improves the performance by fully taking advantages of data locality. We use K-means method for clustering algorithm in GHMC. The reality experiments show that not only our grouped virtual mapreduce cluster costs low and perform better performance, but also our GHMC which uses K-means algorithm has better performance compared with original hybrid mapreduce cluster and shows good scalability.

Acknowledgment

This research is supported in part by the grants of the National Science and Technology Support Project of China (project no. 2011BAH04B05), National Natural Science Foundation of China (project no. 61202077) and the Fundamental Research Funds for the Central Universities (project no. YWF-12-LXGY-008).

References

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

[2] Ibrahim, S.; Hai Jin; Lu Lu; Song Wu; Bingsheng He; Li Qi; LEEN: Locality/Fairness-Aware Key Partitioning for MapReduce in the Cloud, Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on , vol., no., pp.17-24, Nov. 30 2010-Dec. 3 2010

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

[4] Yifeng Geng; Shimin Chen; YongWei Wu; Wu, R.; Guangwen Yang; Weimin Zheng; , "Location-Aware MapReduce in Virtual Cloud," Parallel Processing (ICPP), 2011 International Conference on , vol., no., pp.275-284, 13-16 Sept. 2011

[5] Cong Xu, Sahan Gamage, Pawan N. Rao, Ardalan Kangarlou, Ramana Rao Kompella, and 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.From Grid Computing to Cloud Computing: Experiences on Virtualization Technology. Future Generation Information Technology: Second International Conference, Lecture Notes in Computer Science, 2010, Volume 6485/2010, 41, FGIT 2010, Jeju Island, Korea, December 13-15, 2010. Proceedings.

[7] Yanyan Hu, Xiang Long, Jiong Zhang, Jun He, and 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. and Rehman, M.S. and Sakr, M.F. Center-of-Gravity Reduce Task Scheduling to Lower MapReduce Network Traffic. Cloud2012

[9] Jiahui Jin; Junzhou Luo; Aibo Song; Fang Dong; Runqun Xiong; , "BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing," Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on , vol., no., pp.295-304, 23-26 May 2011

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

[11] Jun Fang; Shoubao Yang; Wenyu Zhou; Hu Song; , "Evaluating I/O Scheduler in Virtual Machines for Mapreduce application," Grid and Cooperative Computing (GCC), 2010 9th International Conference on , vol., no., pp.64-69, 1-5 Nov. 2010

[12] Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, and 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, and 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] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, 1989 :MIT Press. 561-579

[15] Cüneyt F. Bazlamaçcı, Khalil S. Hindi, Minimum-weight spanning tree algorithms A survey and empirical study, Computers & Operations Research, Volume 28, Issue 8, July 2001, Pages 767-785

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

[17] Yang Yang, Xiang Long, Bo Jiang. “An Efficient Grouped Virtual Mapreduce Cluster” . the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013)

[18] http://hadoop.apache.org/#What+Is+Apache+Hadoop%3F

[19] Dan Wang, Jilan Chen, Wenbing Zhao. A Task Scheduling Algorithm for Hadoop Platform.Journal of Computers, Vol 8, No 4 (2013), 929-936, Apr 2013.

[20] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". 1. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and

Probability. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.
[21] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.
[22] J. Clerk Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[23] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[24] K. Elissa, "Title of paper if known," unpublished.
[25] R. Nicole, "Title of paper with only first word capitalized", *J. Name Stand. Abbrev.*, in press.
[26] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[27] M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, 1989.
[28] Han J. W., Kamber M. Data mining: concepts and technique. Morgan Kaufmann, 2001.
[29] Juanying Xie, Shuai Jiang, Weixin Xie, Xinbo Gao.An Efficient Global K-means Clustering Algorithm.Journal of Computers, Vol 6, No 2 (2011), 271-279, Feb 2011

Yang Yang received the Bachelor degree in Computer Science and Technology from the National University of Defense and Technology, in Changsha. Currently, she is a Ph.D. student at Beihang University, in Beijing. Her research interests include virtualization technology and cloud computing.

Dr. Xiang Long received the Bachelor degree in Mathematics from Peking University, in Beijing. He received the master and Ph.D. degree in Computer Science and Technology in Beihang University. Currently, he is a professor at Beihang University, Beijing. His research interests include High performance Computing, computer architectures, Reliability.

Bo Jiang is an Assistant Professor at Institute of Computer Architecture, School of Computer Science and Engineering, Beihang University. He got his Ph.D. from Department of Computer Science of The University of Hong Kong. His research interests involve service and cloud Computing, embedded software testing, and program debugging. http://bojiang.buaa.edu.cn/