Open Access

A New MapReduce Framework Based on Virtual IP Mechanism and Load Balancing Strategy

Song Yang, Hao Pingting, Hu Jiejun, Hu Liang and Che Xilong*

School of Computer Science and Technology, Jilin University, Changchun Jilin, 130012, China

Abstract: MapReduce is an important method for large-scale data processing on parallel architecture. In Hadoop ecosystem, MapReduce runs on the application-level, thus it provides system with flexibility. MapReduce is good at offline batch processing and it could accelerate the whole execution time. The deficiency of the MapReduce architecture is a lack in balancing and scalability, thus leads to low efficiency when dealing with large-scale data. In this paper, we propose a new MapReduce framework that is more suitable for Hadoop ecosystem. The framework is based on the virtual IP mechanism and load balancing strategy. Comparative experiments indicate that the new framework achieve twice the performance compared to the original MapReduce. Besides, the framework fully meets the environment of Hadoop ecosystem, and provides a stable and efficient data processing.

Keywords: Load balancing strategy, parallel batch processing, virtual IP mechanism.

1. INTRODUCTION

MapReduce [1] is a distributed high-performance computing framework for analyzing and processing massive data [2]. It manages data flow and control flow while coordinating with HDFS [3] in the Hadoop [4] ecosystem. Scalability, reliability and high-performance are the main existing defects in MapReduce. On one hand, the load-balancing algorithm in the TaskScheduler does not apply for most submitted job for an even dispatch. On the other hand, it is the users' concern to explicitly allocate the IP addresses of servers, thus lead to an exposure of server information and a lose in executing transparency.

Load balancing [5] is a critical technique to provide higher quality of service and better performance for Cloud computing. It endows the system with the capability of job assignment concerning availability, cost and flexibility. The key problem is how to introduce load balancing into the MapReduce in a dynamic manner so that there is no need to make static reservation at the submission stage while maintaining the transparency of server information.

In order to address this problem, we propose a new MapReduce Framework which incorporate virtual IP mechanism and load balance mechanism together. The main contributions of this literature can be summarized as follows.

(1) The virtual IP (VIP) mechanism is introduced to decouple the static mapping between the MapReduce jobs and the physical resources so that there is no need to make reservation at the submission stage.

(2) The load balance mechanism is employed to dynamically identify underload resources while dispatching MapReduce jobs at execution stage.

(3) Typical load balance algorithms are evaluated through comparative study, and a prototype implementation is validated.

2. RELATED WORKS

MapReduce is the master/slave (M/S) framework. As is shown in Fig. (**1**), it is consisted of Client, JobTracker, TaskTracker. Users submit programs to Client, and then Client sends jobs to JobTracker. JobTracker assigns jobs to each TaskTracker. At the same time, it feedbacks the heartbeat in real time. The TaskTracker uses "slot" to strip resources.

In [6] presents a hierarchical MapReduce framework. The framework supports cross-domain, so that it could gather computation resources from different clusters and run jobs by using these resources. The framework adds two scheduling algorithms, which are Compute Capacity Aware Scheduling (CCAS) and Data Location Aware Scheduling (DLAS). The scheduling algorithm is used to promote the performance of computing. But it does not concern about performance and stability of MapReduce in this paper.

In [7] presents a system that improves the job scheduling strategy of MapReduce for allocating resources reasonably. The system uses the strategy of requesting priority to decide the assignment of resources allocation, as well as the sequence of auto-detection. However, the tactic is not appropriate for massive data sets in virtual environment. The request priority strategy may react load balancing. In addition, it could increase the degree of coupling.

Load balancing strategy [8] mainly includes Round-Robin Scheduling (RR), Weighted Round-Robin Scheduling (WRR) [9], Least-Connection Scheduling(LC) [10], Weighted Least-Connection Scheduling(WLC) [11], Locality-Based Least-Connection Scheduling(LBLC) [12], Locality-Based Least-Connection with Replication Schedul-

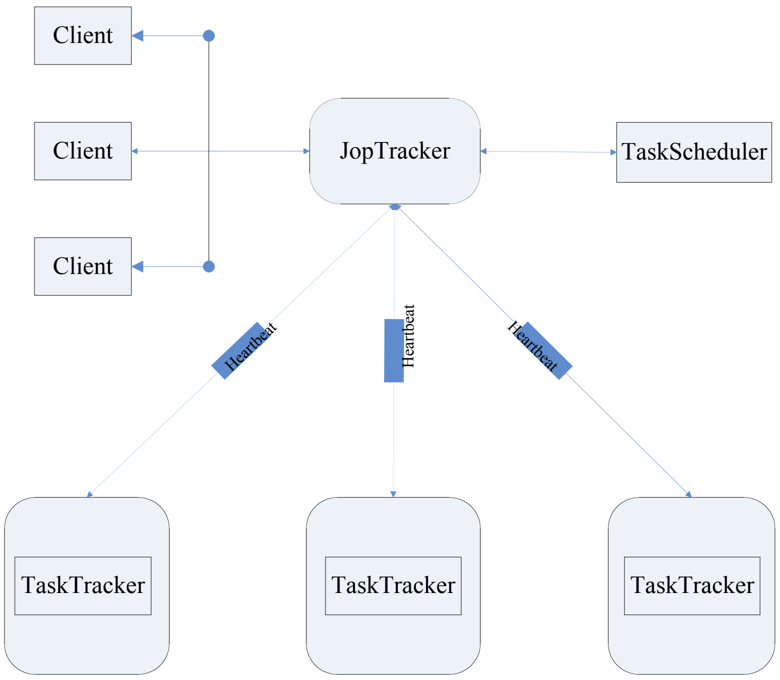

Fig. (1). Hadoop MapReduce Framework.

ing(LBLCR), Destination Hashing Scheduling(DH), Source Hashing Scheduling and so on. Aiming at practical situation or experimental questions, algorithms can be restructured to optimize job assignment, such as appending dynamic detection based on the state of node, or adding algorithms based on some effective math conclusion into load balancing strategy.

Combining with actual circumstance, virtual IP can be designed into different architecture of the cluster. For example, Master/Slave, C/S or etc. One node of the cluster can be chosen as Leader to handle requests, and it can also schedule and distribute works in specific situation. Other nodes are responsible for accepting requests from Leader or disposing of works.

[13] introduced VIP mechanism into the cluster system to improve the throughout and efficiency. It uses one address to name all the cluster nodes for building a virtual cluster with large amount of low-cost resources. The VIP mechanism is proved to be not only scalable and available, but also multi-node fault-tolerance [14]. The VIP works on IP layer, thus the users and the servers do not need to handle the virtual traffic in the application level.

3. FRAMEWORK

3.1. Topology

The new MapReduce framework is presented in Fig. (**2**). It includes Virtual IP mechanism and Load balancing strategy. In what follows, we explain key components.

Master node manages the metadata and the cataloging tree of HDFS, it receives the heartbeat and execution results from the Slave node.

Slave node runs the tasks and submits the results to the Master node.

JobTracker receives commands issued by users and sets up parameters according to the dynamic state of the network. It also divides each job into one or more blocks.

TaskTracker executes tasks dispatched by the JobTracker.

Leader VIP takes charge of all the tasks. When a task is sent from client to cluster, it is firstly gathered by the node where Leader VIP runs. Then the Leader VIP issues the tasks to working nodes with VIPs.

Super VIP woks like a Leader VIP, it sends tasks to Leader VIPs instead of working nodes. We will not go into details about super VIP.

3.2. VIP Mechanism

VIP mechanism is a core design of our framework. It is an IP address that is virtually assigned to multiple domain names or servers. It could improve redundancy by providing alternative failover options on the servers evolved.

While a client generates a task, it explicitly nominate a VIP address to execute its mappers and reducers. The VIP is not an physically available IP, but all the cluster nodes are aware of the mapping relationship between the virtual and physical address.

When the Super VIP receives tasks from a client, it transfers the virtual address into a physical address of Leader VIP, and selects one Leader VIP to run it. The selection is performed according to inter-cluster load balancing strategy. When the Leader VIP receives a task from a Super VIP, it dispatches them to VIPs according to the intra-cluster load balancing strategy. Therefore, both inter-cluster and intra-cluster are concerned to ensure that the proposed framework makes fully use of available resources.

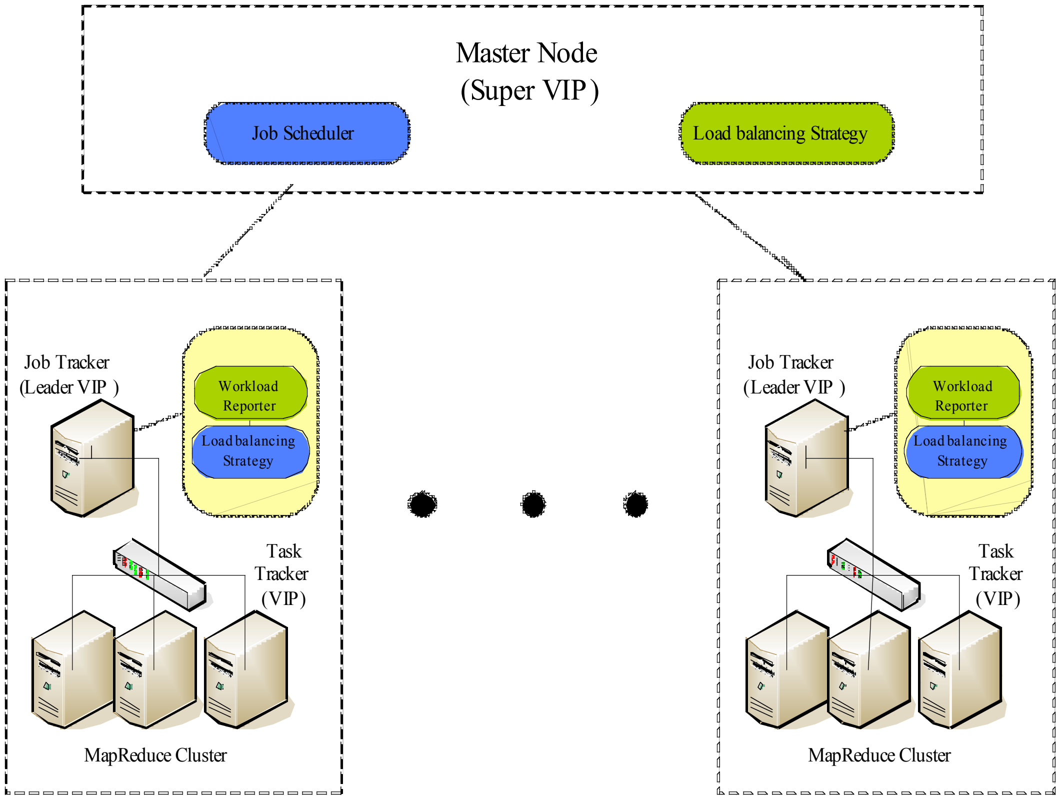

Fig. (2). A new MapReduce architecture.

Since the virtual IP are dynamically mapped to physical IP in the dispatch stage, it is more easy to balance the task distribution and robust execution.

3.3. Heartbeat

The original MR by sending a heartbeat [15] packet to determine whether TaskTracker survival, understanding the space, it can run JobTracker assigned tasks.

As is shown in Fig. (**3**), the original MapReduce sends heartbeat according to the five steps as follows:

1. To determine whether to send heartbeat time, according to the TaskTracker to dynamically adjust the number of sending heartbeat time.
2. Determine whether the TaskTracker is just the start, need to check the TaskTracker and JobTracker versions are consistent.
3. Check whether the damaged disk
4. Send a heartbeat packet
5. To receive and execute the command

In our new MapReduce framework, as is shown in Fig. (**4**), JobTracker according to the VIP mechanism to control to which sub node sends a heartbeat packet, dynamic deployment according to the real time condition, achieve dynamic extensible, so do the precise location in the subsequent calculation, improve the call rate, improve the ability of fault tolerance.

The new method has six step as follows:

1. According to the configuration of the VIP, set the cluster
2. Whether to send heartbeat time
3. To find the corresponding VIP address according to the VIP configuration file, to judge whether the TaskTracker has just started
4. According to the SVIP address to check whether there is damage to the hard drive
5. Send a heartbeat packet to the specified VIP address, send heartbeat to TaskTracker
6. To execute the command and return results to the SVIP.

3.4. Loadbalance

In the proposed framework, load balancing strategy is inevitable to achieve good performance. We take the following scheduling algorithms under concern.

DH Scheduling is a static algorithm that mappes tasks to target IP addresses according to hash function on IP addresses. In our framework, this algorithm is used b the JobTracker to select a TaskTracker for dispatching each task.

LC Scheduling is a dynamic algorithm that assigns requests to the servers with least connections. A director records the connection count for each server. More specifically, the count is added by one if one task assigned, and vice versa. If two servers have same connection count, then the first place in logic is chosen. This algorithm will lead to uneven distribution if the execution time of tasks are rather uneven. In that case, it can lead to low efficiency.

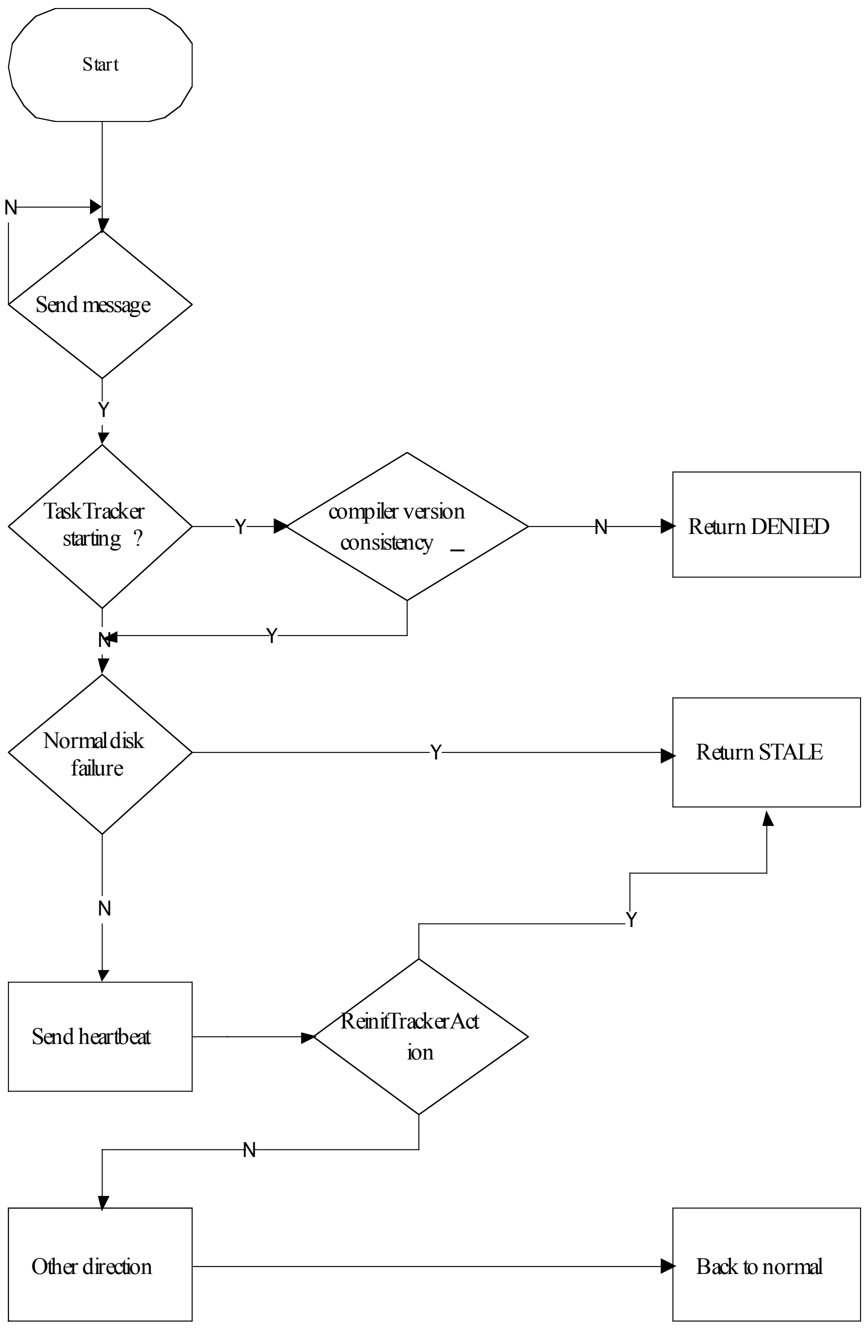

Fig. (3). The original process of MapReduce heartbeat.

LBLC Scheduling is originally designed for cache use in the cluster system but is widely used in a great many situations. This algorithm combines LC and DH together. When there is an available server under loaded, then DH is adopted, otherwise, LC runs in turn.

3.5. Workflow

Clients will send request to JobTracker's Leader Virtual IP, JobTracker can monitor every TaskTracker and Job's health, and according to the load balance strategy to allocate the resources for task. And then JobTracker could perform the job according to the node distribution by TaskTracker's Virtual IP. JobTracker will send the heartbeat to TaskTracker to confirm the TaskTracker is alive, and TaskTracker replies it to JobTracker. This action can be completed by virtual IP mechanism, to guarantee the stability and scalability. By using Virtual IP mechanism can achieve the task distribution and recovery when the node failures immediately transfer or recovery.

In the new framework, we use virtual IP mechanism to improve the fault-tolerance performance and computational performance. JobTracker finds its own virtual IP layer through the previous configuration file. Load balancing strategy and network traffic is decided by the current network environment.

The process of a task execution request is consisted of 8 steps, as is shown in Fig. (**5**):

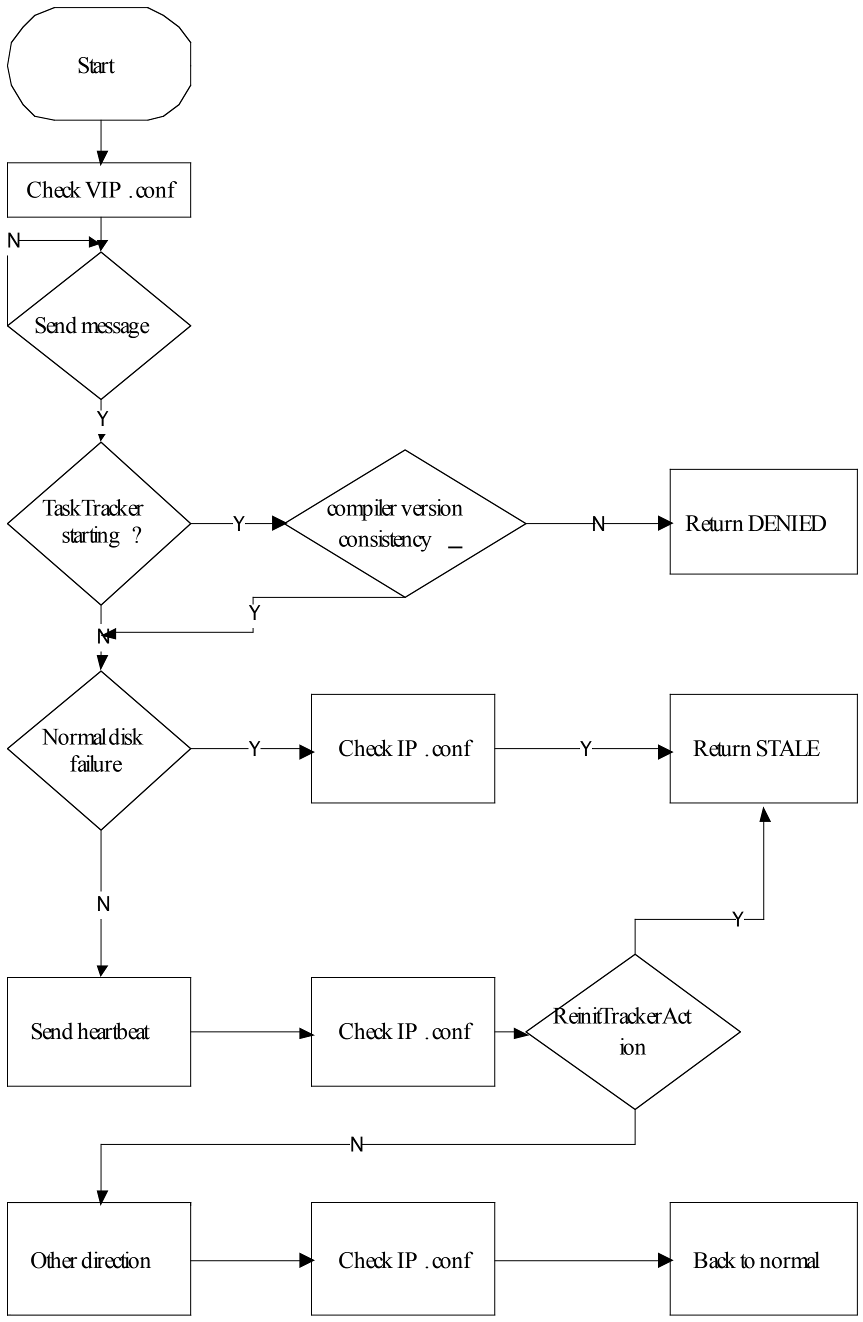

Fig. (4). The new process of MapReduce heartbeat.

1. Client proposes a request about dealing with jobs and submitting jobs
2. If the submitted IP by the client is equivalent with VIP of cluster, the node with leader VIP in the cluster starts to send packet to check the health of Task Tracker.
3. If Task Tracker is alive, it responds the node with leader VIP to prove its existence.
4. The node with leader VIP assigns tasks to Load Balance module.
5. Load Balance module returns tasks list to the node with leader VIP. The node with leader VIP gives the list to Job Tracker.
6. Job Tracker passes the list on to Task Tracker, and makes Task Tracker to assign tasks by strategy of Load Balance.
7. Task Tracker launches tasks to Task. The next step is going on the phase of Reduce. After the phase of Reduce, the job is accomplished.
8. Task returns the result to parts of HDFS in Hadoop ecosystem.

As is shown in Fig. (**6**), the data flow is changed in the new MapReduce framework. On account of the relationship between the data transferring in original architecture, the new framework based on the hierarchy, integrated virtual IP mechanism. Each task is divided into blocks, a plurality of

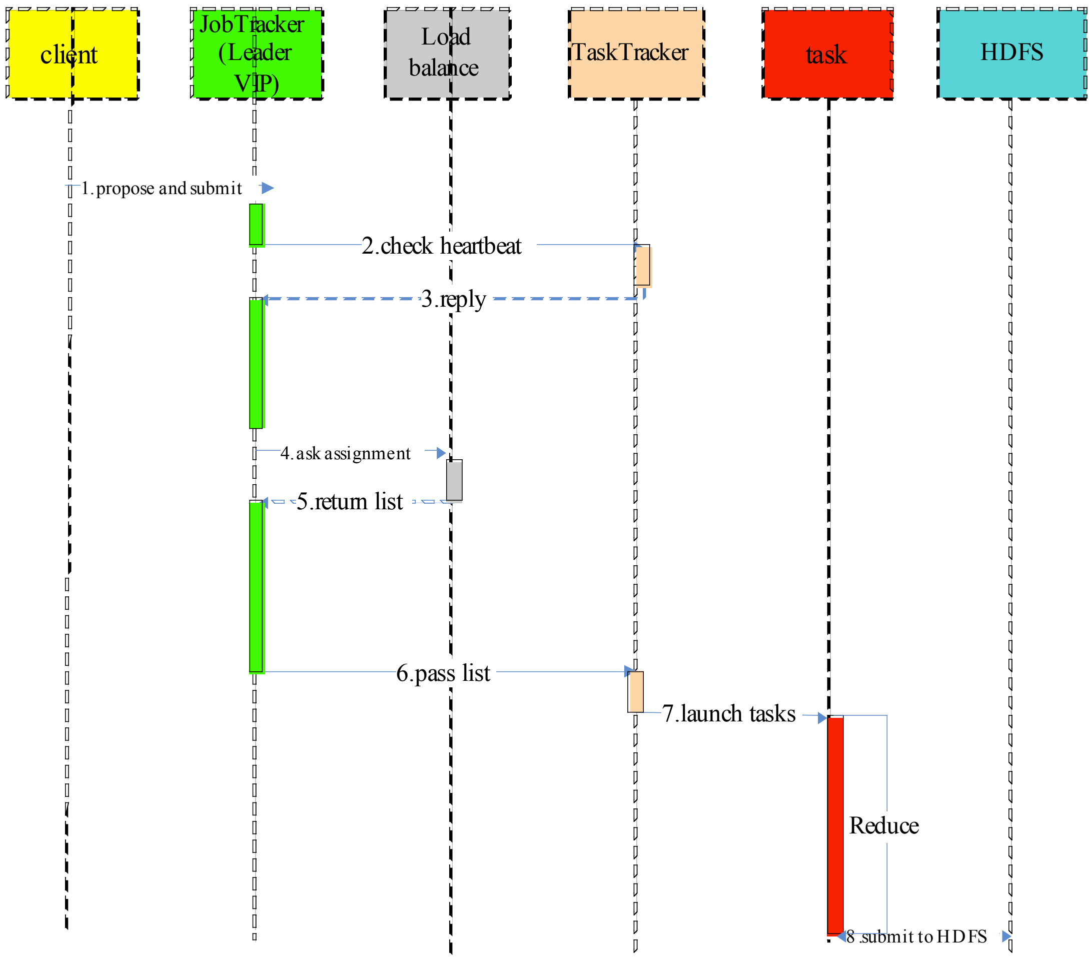

Fig. (5). A new MapReduce work-flow diagram.

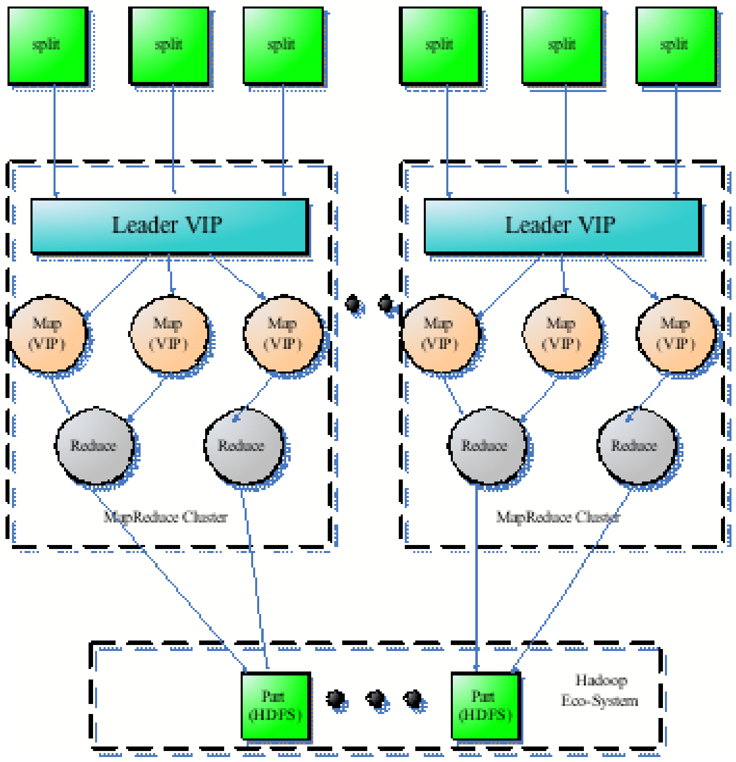

Fig. (6). A new MapReduce data-flow diagram.

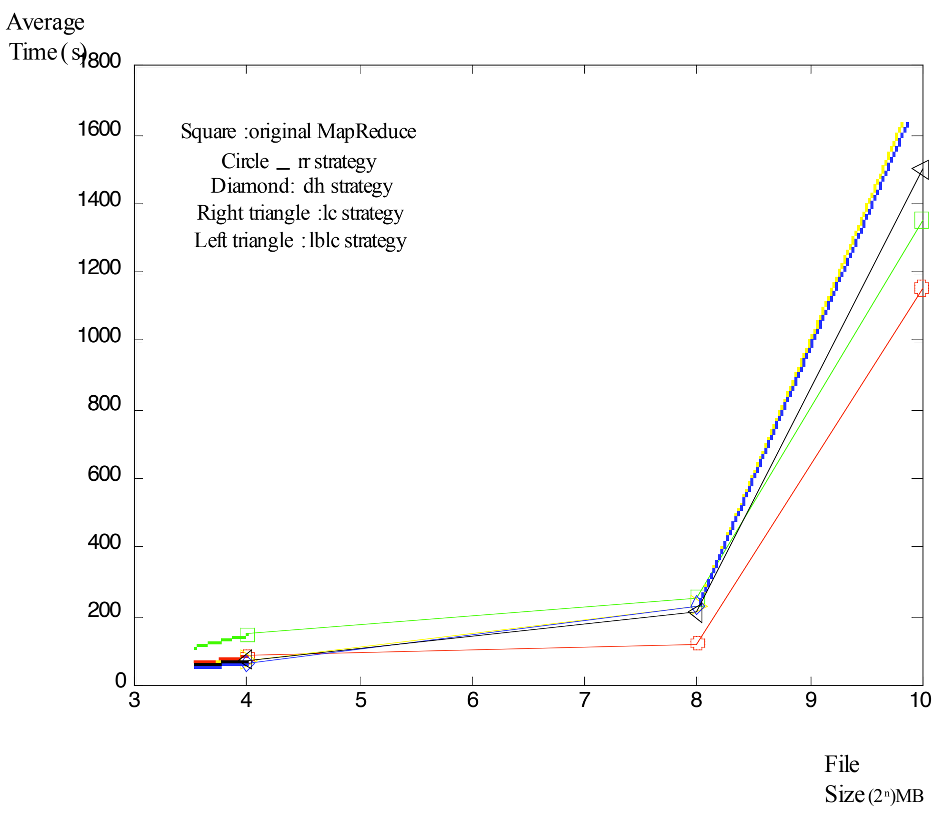

Fig. (7). The average execution time for each size.

data streams entering the system through the way of virtual IP.

4. EXPERIMENT

4.1. Experimental Settings

The unit of Datasets is 1MB, including a single word, special symbols, spaces and so on. Each set of test data is formed by the unit of datasets. The unit of datasets forms each test data set.

The real experimental environment for us is two PC machines, core 2 Duo processor, 8GB memory. One machine established four virtual machines, which are master, slave 1, slave 2 and slave 3. Another established five from slave 4 to slave 8. The network is connected through bridge-connection.

Based on the above in-depth analysis of MapReduce, ensuring the experimental data is accurate and persuasion, we divided into four groups to compare, which are 8 documents, 16 documents, 256 documents and 1536 documents. In the meantime, take four strategies in experiment that is RR strategy, DH strategy, LC strategy, LBLC strategy.

We propose a new framework based on the MapReduce. We modify the related API, specific parameters and add some configuration files to ensure that the new framework can be applied to the Hadoop ecosystem.

Although our architecture is very suitable for the operation in the practical scenario of large-computing, on account of the limitation of our experimental environment, the strategy of RR load balancing is better in this experiment.

4.2. Results and Discuss

We accomplish the prototype system by modifying LVS and MapReduce. Then we take a set of test data which is special for verifying the capacity of offline processing.

When the task comes to TaskTracker, first of all we judge TaskTracker whether it is occupied. If TaskTracker is busy, the task will jump to the next TaskTracker.

As is shown in Fig. (**7**), We analyze the trace of the average execution time for each size of job. Compared to original architecture, the overall speed of job execution will increase 200% by using RR strategy. The improvement percentages are also noticeable in other three load balancing strategies. The job size of 256MB is the turning point in the process of operation. When the file size exceeds 256MB, it will increase computation time. Beyond the ability of computing in the cluster, the average time increases sharply.

In Fig. (**8**) shows that the error rate of new framework is not affected so much. In 1536MB, the error rate rose to a certain degree lower than original MapReduce in DH strategy, LB strategy and LBLC strategy. The failure of RR strategy has the lowest rate of failure among four strategies. You can also find that the original MapReduce increases the rate of failure to a certain degree, and then the number of error is controlled in an average level. The error rate in original MapReduce framework is much higher than our new framework.

We take the job size of 256MB as an example to make a detailed analysis and data display in Fig. (**9**). The result shows that the first experiment of speed efficiency is lower than latter experiment. The reason is that the first reading

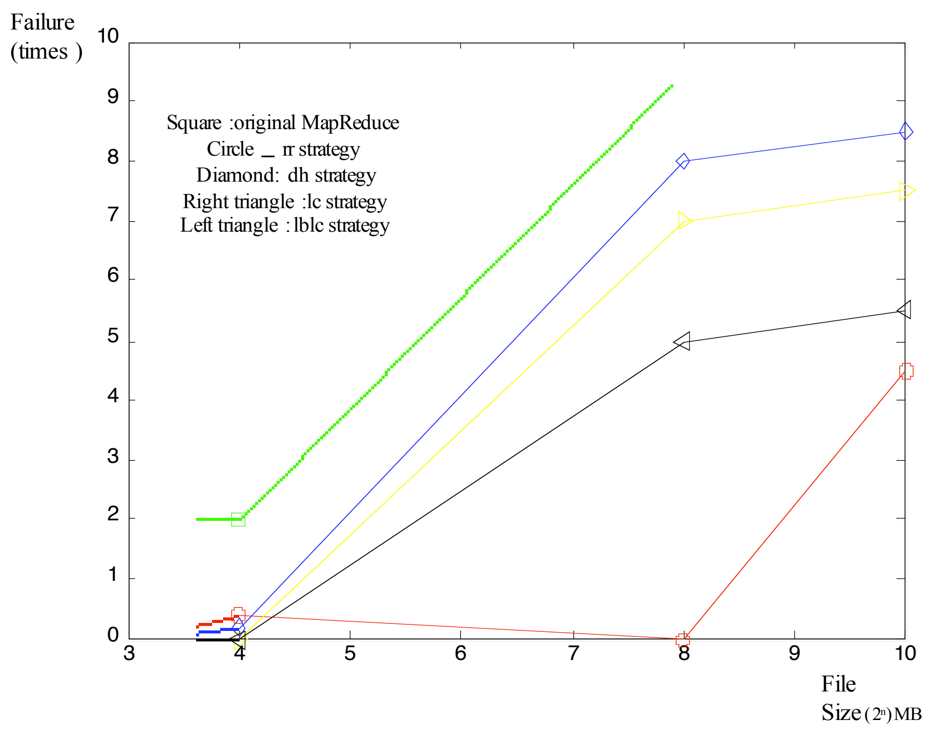

Fig. (8). The average error of multiple test data.

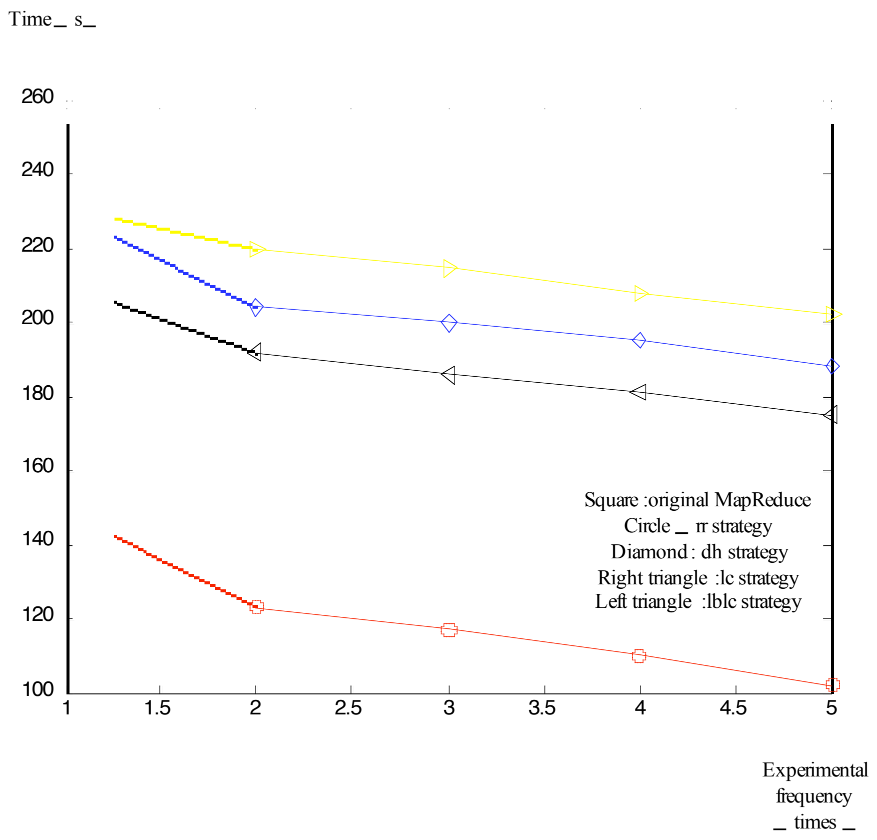

Fig. (9). The running time and the number of tests in 256MB.

from the dataset will slow down the speed of the system. Through the curve can be seen that the calculating rate of original framework did not improve much, while our new framework gradually speeds up the procedure until a stable point.

In conclusion, our architecture improves much better execution efficiency than that of the original MapReduce under different file size. The new framework also achieves a roughly increasement about 200%. Furthermore, we can find that with the same circumstance, the error rate is reduced to some extent.

5. CONCLUSION

We proposed a framework which is fully applicable for Hadoop ecosystem. We add virtual IP mechanism and load balancing strategy into MapReduce. They make a perfect fusion in the new framework to optimize the procedure of MapReduce.

In this paper, we do not combine it into one single aspect of Hadoop. In another word, it not only optimizes the code for one module, but will propose a new computing architecture. Load balancing strategy and Virtual IP mechanism help the entire computing clusters become more scalable and stable. Although the new framework proposed may not be the best in one specific aspect, it can achieve high performance computing by the combination of load balancing and virtual IP mechanism.

In future work, we will improve the load balancing strategy of adaptive strategies that can be suitable for more computing environment. We can consider more popular load balancing strategies, for instance dynamic algorithms, CDN.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is funded by: European Framework Program (FP7) under Grant No. FP7-PEOPLE-2011-IRSES, and by National Natural Science Foundation of China under Grant No. 61073009 & 60873235, and by National High Technology R&D Program 863 of China under Grant No. 2011AA010101, and by National Key Technology R&D Program of China under Grant No. SQ2013GX11E00316, and by National Science and Technology Major Projects of China under Grant No. SinoProbe-09-01-03 & 2012ZX01039-004-04-3, and by Key Science Technology Program of Jilin Province of China under Grant No. 2011ZDGG007.

REFERENCES

[1] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters", In: *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, December 06-08, 2004, San Francisco, CA, 2004, pp. 10-10.

[2] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system" In: *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, IEEE, 2010, pp. 1-10.

[4] The Apache Software Foundation. http://hadoop.apache.org/

[5] A.Y. Zomaya, and Y.H. The, "Observations on using genetic algorithms for dynamic load-balancing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 12, no. 9, pp. 899-911, 2001.

[6] Y. Luo, and B. Plale, "Hierarchical mapreduce programming model and scheduling algorithms," In: *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012),* IEEE Computer Society, 2012, pp. 769-774.

[7] T. Sandholm, K. Lai, "MapReduce optimization using regulated dynamic prioritization," In: *Proceedings of the 11th international joint conference on Measurement and modeling of computer systems, ACM*, 2009, pp. 299-310.

[8] S. Sharifian, S.A. Motamedi, and M. K. Akbari, "A predictive and probabilistic load-balancing algorithm for cluster-based web servers," *Applied Soft Computing*, vol. 11, no. 1, pp. 970-981, 2011.

[9] L.B. Le, E. Hossain, and A.S. Alfa, "Service differentiation in multirate wireless networks with weighted round-robin scheduling and ARQ-based error control", *IEEE Transactions on Communications*, vol. 54, no. 2, pp. 208-215, Feb. 2006.

[10] D. Choi, K. S. Chung, and J. Shon, *An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems*, Springer Berlin Heidelberg, vol. 121, 2010, pp. 127-134.

[11] M. Chiang, C. Wu, Y. Liao, and Y. Chen, "New content-aware request distribution policies in web clusters providing multiple services," *SAC'09 Proceedings of the ACM symposium on Applied Computing*, 2009, pp. 79-83,

[12] H. Kwak, A. Sohn, and K. Chung, "Autonomous learning of load and traffic patterns to improve cluster utilization," *Cluster Computing*, vol. 14, no. 4, pp. 397-417, Dec. 2011.

[13] Wikimedia Foundation. Inc., http://en.wikipedia.org/wiki/Virtual_IP_address/

[14] T. Schroeder, S. Goddard, and B. Ramamurthy, "Scalable web server clustering technologies," *Network, IEEE*, vol. 14, no. 3, pp. 38-45, 2000.

[15] J.S. Manjaly, and V.S. Chooralil, "TaskTracker Aware Scheduling for Hadoop MapReduce,"In: *Advances in Computing and Communications (ICACC),* In: *3rd International Conference on,* IEEE, Aug. 2013, pp. 278-281.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014