2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control

A Survey on MapReduce Scheduling in Cloud Computing

Li Liu
School of Automation and Electrical Engineering
University of Science and Technology Beijing, China

YingQi Zhai
School of Automation and Electrical Engineering
University of Science and Technology Beijing, China

***Abstract*—A large-scale data processing is increasingly leveraging MapReduce frameworks on Cloud computing platform. We aimed to study various job schedulers improved in MapReduce, and identify research directions in this area. In this paper, the related techniques of MapReduce and Hodoop are described briefly. Then a variety of MapReduce schedulers are reviewed and classified, also the features of these schedulers are expressed.**

***Keywords*—Hadoop; MapReduce; Scheduling; Clould Computing**

I. INTRODUCTION

Inspired by the power of large scale data processing, on-demand configuration of scalable and reliable computing services, along with a cost model which charges to consumers based on actual usage of service, has been a goal in distributed computing industry and research for a while[1]. Cloud computing has become a very attractive computing model and grown rapidly, which focus on delivering computing power to the end-user directly, no matter when and where[2].

By means of virtualization technologies, Cloud computing offers to users a wide variety of services enclosing the whole computing stack, from the application to the hardware level. Cloud computing is expected to achieve the goal of allowing consumers to deploy applications and store data, and access them via Web protocols from anywhere in the world based on users Quality of Service (QoS) requirements and paying-per-use[3].

Large-scale data processing hosted on Cloud computing platform has become an important research problem. Typically, these applications leverage parallel programming model like MapReduce frameworks for developing on cloud. MapReduce is a programming model for processing large data sets in a highly-parallel way [4].

Scheduling for MapReduce job in Cloud environments is an open issue [5]. The job in MapReduce is divided into two types of task: map task and reduce task. The number of tasks can be executed concurrently on a node. In this paper, the existing MapReduce schedulers are classified in five categories: Locality aware Scheduling, Real-time aware (Deadline) Scheduling, Resource aware Scheduling,Cost/Performance aware Scheduling and Workload aware Scheduling.

II. OVERVIEW OF HDFS AND MAPREDUCE IN HODOOP

Hadoop is an open-source implementation for MapReduce, which has become the leading platform for large-scale data processing [6]. Hadoop hide the details of parallel processing, including the distribution of the data to the processing nodes, restarting failed subtasks, and merging calculation results. The framework allows developers to write parallel processing programs to deal with large scale data, without to understand the underlying details of the distributed processing system.

Hadoop consists of two core parts, which are Hadoop Distributed File System (HDFS) and distributed computing framework named MapReduce. HDFS is a distributed file system that store large amount of data with high throughput access to data on clusters. MapReduce is a software framework for distributed processing of data on clusters.

A. HDFS Architecture

As shown in Figure 1, HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode and a number of DataNodes. NameNode is a master server that manages the file system namespace and control access to files by clients. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode is the arbitrator and repository for all HDFS metadata, which executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. They also perform block creation, deletion, and replication upon instruction from the NameNode. [7]

B. MapReduce structure

In Cloud, a MapReduce service is used to analyze large scale data cost-effectively. By means of virtual machines (VMs) and storage hosted on Cloud, we can simply build virtual MapReduce clusters to process data [8].

MapReduce is described by defining a "mapper" and a "reducer". The "mapper" processed by "map" function is used to generates a number of intermediate <key, value> pairs from the input <key/value> pair. The "reducer" executed by "reduce" function is used to merge all the intermediate <key/value> pairs with the same intermediate key.

978-1-4673-7723-2/15 $31.00 © 2015 IEEE
DOI 10.1109/IMCCC.2015.363

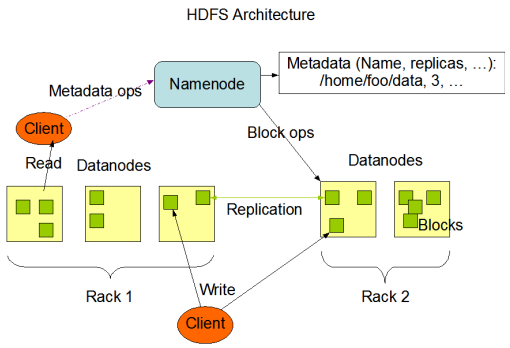

Figure 1 HDFS Architecture

MapReduce implementations (e.g., Apache Hadoop and Google's MapReduce [9]) are based on a master-slave structure shown as figure 2. A job is submitted by a Client to a JobTracker and then JobTracker selects idle TaskTrackers and assigns each one to some map or reduce tasks. When all map and reduce tasks have been completed, the JobTracker returns the result to the Client. If a TaskTracker failed, another TaskTracker will re-execute the failed task. Jeffrey Dean et al [4] describe the basic programming model in detail and give several examples and refinements of the programming model.

The process of MapReduce implementation is a unified with sequential and parallel processing. As Figure 3 shown, the Map phase is executed before the Reduce step. However, in each phase many Map or Reduce processes are executed in parallel [10].

Figure 4 presents an outline of the job execution steps in MapReduce framework. An initial large data set is split into processing blocks via a split function, which defines the data processing splits, such as a column, a database row or a line of a file. This function must generate <key, value> pairs, such as the position and the content of the line in a file. Each result from the split phase is assigned to a worker. Then they execute the map function and reduce function. As a result, each worker generates a partial output of the data which is automatically stored in an output file. A job will be divided into a number of Map tasks and Reduce tasks. Note that users only have to specify the split, map, reduce and functions without having to deal with the aspects related to parallel programming. The framework also is able to schedule, monitor and checkpoint the individual jobs.

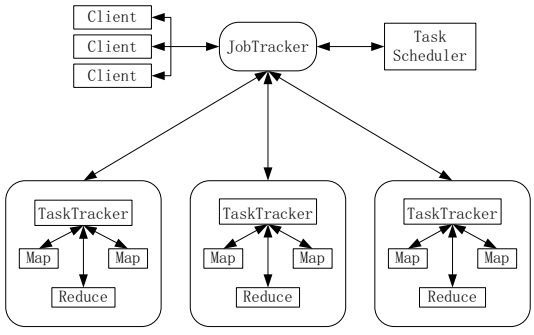

Figure 2 master-slave structure of MapReduce

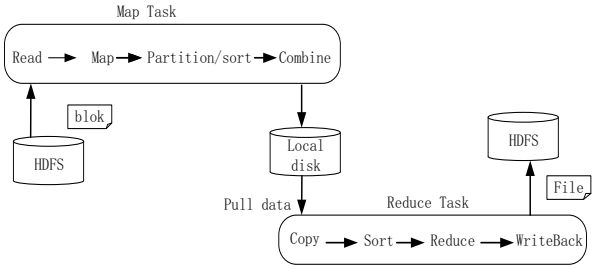

Figure 3 Components in Map and Reduce task and execution sequence

III. TAXONOMY FOR MAPREDUCE SCHEDULING

A. *Default MapReduce scheduling in Hadoop*

Hadoop provides three MapReduce scheduling algorithms. The default Scheduling algorithm is based on FIFO where jobs were executed in the order of their submission [11]. Then, the priority of a Job was set based on the ability. Facebook and Yahoo contributed significant work in developing schedulers such as Fair Scheduler [12] and Capacity Scheduler [13].

- *Default FIFO Scheduler*

The default Hadoop scheduler uses a FIFO queue algorithm. After a job is divided into series tasks, they are loaded into the queue and assigned to free workers. Typically, each job would use the whole cluster, so the other jobs had to wait for their turn. So sharing resources fairly between users requires a better scheduler, Even though a shared cluster offers great potential of offering large resources to many users. So it is disadvantageous for Production jobs to be completed in a timely manner.

- *Fair Scheduler*

For the deficiencies of the Default FIFO Scheduler, the Fair Scheduler [12] [14] was developed at Facebook, which is aimed to manage the access to their Hadoop cluster and subsequently published to the Hadoop community. The Fair Scheduler aims to give every user a fair opportunity to share the cluster resources. There are some resources pools to be installed, and each pool is allocated an appropriate number of Map and Reduce slots (workers).

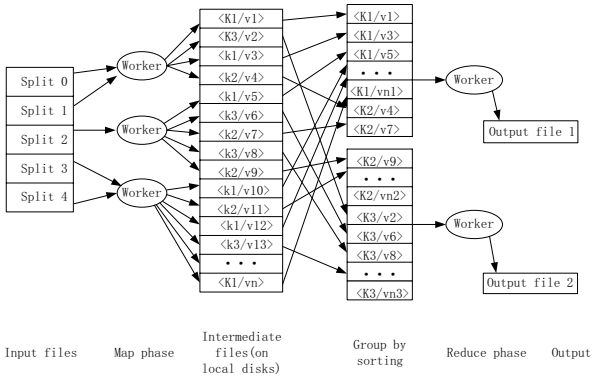

Figure 4 Job Execution Steps in MapReduce

Free slots in idle pools may be allocated to other pools, while excess capacity is shared among jobs. The Fair Scheduler supports preemption mechanism. In addition, administrators can enforce settings priority on some pools. So Tasks are scheduled based on their priority within pool, the cluster capacity and usage of their pool.

• *Capacity Scheduler*

Capacity Scheduler [13] was developed at Yahoo, where the number of users is large, and a fair allocation of computation resources is to be ensured. The Capacity Scheduler allocates jobs according to the users who are queueing and the configurable numbers of Map and Reduce slots. Queues which contain jobs are configured their capacity, while free capacity in a queue is shared among the others. Scheduling operates on a queue with different priority jobs. The priorities are adjusted based on the time a job was submitted, and the class of user or job. The queue with the lowest load is chosen from the oldest remaining job. When a Task Tracker slot becomes free. Tasks are then scheduled from that job.

B. Taxonomy of Improved Schedulers

Many researchers are working for improving the scheduling policies in Hadoop. Recently the improved Schedulers is mainly as following five aspects: *1) Locality aware Scheduling; 2) Real-time (Deadline) Scheduling*, such as Delay Scheduler and the Deadline Constraint Scheduler; *3) Resource aware Scheduling* which was proposed with considering the resource availability to schedule jobs; *4) Cost/Performance aware Scheduling;* and 5) *Workload aware Scheduling* where the schedulers try to allocate capacity to users or jobs fairly among resources.

• *Locality aware Scheduling*

Due to limited network bandwidth and high data transfer cost, locality is a crucial performance issue in a shared resources environment. There is a conflict between fairness in scheduling and data locality (placing tasks on nodes that contain their input data). Matei Zaharia et al [15] illustrate this problem through designing a fair scheduler for a 600-node Hadoop cluster at Facebook. They proposed a simple algorithm called delay scheduling, i.e. when the job that should be scheduled next cannot launch a local task according to fairness, it waits for a time to let other jobs launch tasks instead. The delay scheduling achieves nearly optimal data locality in a variety of workloads and can increase throughput by up to 2 times while preserving fairness. Mohammad Hammoud et al. proposed a practical strategy which data locality is improved by using network locations [8]. Jiahui Jin et al. proposed a heuristic task scheduling algorithm which can adjust data locality dynamically according to network state and cluster workload [16].

• *Real-time (Deadline) Scheduling*

K. Kc et al. [17] extend real time cluster scheduling approach to account for the two-phase computation style of MapReduce. They present criteria for scheduling jobs based on user specified deadline constraints and discuss the implementation and evaluation of a Deadline Constraint Scheduler for Hadoop, which ensures that jobs are scheduled and accomplished within the specified deadlines. After a job is submitted, the ability of scheduler is to be estimated whether the job can be accomplished within the specified deadline or not. Based on the different deadlines of jobs, the scheduler will deploy different number of slots to them to satisfy the specified deadline.

Verma A et al. [18] introduced a novel framework to address the deadline problem and offer a new resource sizing and provisioning service in MapReduce environments. For a job needs to be completed within a certain time, the profile can be built from its past executions or by executing the application on a smaller data set using an automated profile tool. Then, by applying scaling rules combined with a fast and efficient capacity planning model, they generated a set of resource provisioning options. In [19], authors proposed mechanisms that enhance workload management decisions for processing MapReduce jobs with deadlines. The three mechanisms they considered are: a policy for job ordering in the processing queue; a mechanism for allocating a tailored number of map and reduce slots to each job with a completion time requirement; a mechanism for allocating and reallocating spare resources in the system among the active jobs. They implemented a novel deadline-based Hadoop scheduler that integrates all these three mechanisms.

Phan Linh T.X. et al. [20] explored the feasibility of enabling the scheduling of mixed hard and soft real-time MapReduce applications. They first presented an experimental evaluation of the popular Hadoop MapReduce middleware on the Amazon EC2 cloud. Their evaluation revealed tradeoffs between overall system throughput and execution time predictability, as well as highlights a number of factors affecting real-time scheduling, such as data placement, concurrent users, and master scheduling overhead. They formulated the offline scheduling of real-time MapReduce jobs on a heterogeneous distributed Hadoop architecture as a constraint satisfaction problem (CSP), and proposed heuristic method for the online scheduling.

Cho B et al. [21] presented Natjam, a system that supports arbitrary job priorities, hard real-time scheduling, and efficient preemption for resource-constrained Mapreduce clusters. Their contributions include: a) exploration and evaluation of smart eviction policies for jobs based on resource usage, task runtime, and job deadlines; b) a work-conserving task preemption mechanism for Mapreduce. They integrated Natjam into the Hadoop YARN scheduler framework.

• *Resource aware Scheduling*

Running MapReduce programs in the public cloud introduces the important problem: how to optimize resource provisioning to minimize the financial charge for a specific job? Fengguang Tian et al. [10] researched the MapReduce processing and presented a cost function explicitly modeling the relationship between the input data and the available system resources (Map and Reduce slots). Based on this cost model, resources could be optimized to minimize the cost under a deadline or minimize the time under certain budget.

Mark Yong et al.[22] proposed two more efficient resource aware scheduling schemes that minimize contention for CPU and IO resources on worker machines and can give better performance on the cluster.

Polo J, et al. [23] presented a resource-aware scheduling method for MapReduce multi-job workloads which aimed at improving resource utilization across machines while satisfying completion time goals. Their method leveraged job profiling information to dynamically adjust the number of slots on each machine, as well as workload placement across them, to maximize the resource utilization of the cluster.

• *Cost aware Scheduling*

Zhang Z. et al [24] offered a framework for evaluating and selecting the right underlying platform (e.g., small, medium, or large EC2 instances) and achieving the desirable Service Level Objectives (SLOs). Users can define a set of different SLOs: a) achieving a given completion time for a MapReduce jobs while minimizing the cost (budget), or b) for a given budget select the type and the number of instances that optimize the MapReduce workload performance. Their evaluation study and experiments in Amazon EC2 platform revealed that for different workload mixed the optimized platform choice might result in 37-70% cost savings for achieving the same performance goal when using different option.

Bampis E. et al. [25] proposed power-aware MapReduce scheduling scheme. They focused on the minimization of the total weighted completion time of MapReduce jobs under a given budget of energy. A polynomial time constant-factor approximation algorithm was proposed based on linear programming. A convex programming formulation was presented to combine with standard list scheduling policies.

Palanisamy Bet al [26] proposed a new MapReduce cloud service model, Cura, for data analyzing in the cloud. Cura leverages MapReduce profiling to automatically create the best cluster configuration for the jobs so as to obtain a global resource optimization from the provider perspective. Secondly, to better serve MapReduce workloads with a large proportion of interactive real-time jobs, Cura uses a unique instant VM allocation technique to reduce response times by up to 65%. Cura also benefits from a number of additional performance enhancements including cost-aware resource provisioning, VM-aware scheduling and online virtual machine reconfiguration. Experimental results showed that there was more than 80% reduction in cost of the cloud data center along with response time improvements.

• *Workload aware Scheduling*

Wang Zhe, et al [27] put forward a scheduling strategy, i.e. CICS (CPU and I/O Characteristic Estimation Strategy), according job type, which was based on classical FCFS and had been improved intensively on fairness. This scheduling strategy included two parts: dividing the job dynamically into two types based on cluster historical operating data, i.e. CPU-intensive and I/O-intensive; removing the influence of noise data on the reliability of historical data.

Mude R. G. et al. [28] proposed a scheduler which could capture the node resource status, classify jobs into two types of CPU bound and IO bound, and assign task to the node having less CPU/IO utilization. The experimental result showed an improvement of 15%-20% on heterogeneous and around 10% of homogeneous cluster with respect to Hadoop native scheduler.

Zhang Q. et al [29] addressed several key challenges of dynamic workload management in heterogenous Cloud environments. They firstly presented a scheme that deploy service application across geographically distributed data centers to meet service demand while minimizing total resource usage cost. Then, they designed a heterogeneity-aware dynamic application provisioning technique to minimize energy consumption while satisfying performance objectives. They also presented a novel scheme to leverage heterogenous run-time task usage characteristics. Experiments and simulations showed that their proposed solutions can significantly reduce data center energy consumption, and achieve better application performance in terms of service response time and job completion time.

From mentioned above, we summarize the type and features of existing MapReduce scheduler in Table I.

TABLE I. TYPE OF EXISTING MAPREDUCE SCHEDULER IN CLOUD

Method \ Features		Advantage	Heterogeneous environments	Speculated mechanism	Dynamic deployment
Locality aware Scheduling	[20]	increase throughput improve response times	✓	-	-
	[8]	enhancing performance reducing network traffic improve scheduling delay	-	✓	-
	[16]	reduce completion time	-	✓	✓
Real-time aware	[21]	Increase of performance	-	-	-
	[17]	Improve utilization,	-	✓	-
	[19]	Increase of performance	-	✓	-

(Deadline) Scheduling	**[20]**	reduce completion time increase throughput	✓	-	-
	[18]	Increase performance predicted completion times accurately	-	✓	✓
Resource aware Scheduling	**[22]**	Reduce contention for CPU resources Increase performance	-	-	✓
	[23]	Increase resource utilization	✓	-	✓
	[10]	cost model performs well minimize the financial cost with a time deadline or minimize the time under certain financial budget	-	-	-
Cost/Performance aware Scheduling	**[24]**	37-70% cost savings for achieving the same performance objectives	✓	✓	-
	[25]	minimization of the total weighted completion time	✓	-	-
	[30]	reduces the data transfer and the speculative tasks degrades the job finish time	✓	✓	✓
	[31]	Increase performance Decreases the execution time	✓	✓	✓
Workload aware Scheduling	**[27]**	Decreases the execution time	✓	-	✓
	[28]	Reduce job completion time	✓	-	✓
	[29]	Reduce data center energy consumption, response time and job completion time.	✓	✓	✓

IV. Conclusion

Traditional data processing and storage approaches are facing many challenges in meeting the continuously increasing computing demands of Big Data. This work focused on MapReduce, one of the key enabling approaches for meeting Big Data demands by means of highly parallel processing on a large number of commodity nodes.

This paper summarizes pros and cons of scheduling policies of various Hadoop Schedulers developed by different communities. We classify these schedulers from five aspects: Locality aware Scheduling, Real-time aware (Deadline) Scheduling, Resource aware Scheduling, Cost/Performance aware Scheduling and Workload aware Scheduling.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No 61370132, No.61472033, No.61272432) and the State High-Tech Development Plan (No.2013AA01A601)).

References

[1] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, “A comparison of approaches to large-scale data analysis,” in Proceedings of ACM SIGMOD Conference, 2009.
[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computing System vol.25, 2009, pp.599–616.
[3] Klems M, Nimis J, Tai S. Do clouds computer a framework for estimating the value of cloud computing [M]//Designing E-Business Systems. Markets, Services, and Networks. Springer Berlin Heidelberg, 2009: 110-123.
[4] Dean, J., Ghemawat, and S.: MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, vol. 51 n. 1 (2008) 107-113.
[5] K. H. Lee, Y. J. Lee et al., Parallel data processing with MapReduce: a survey", ACM SIGMOD Record, 2011, Vol. 40, No. 4, pp. 11-20
[6]Apache Hadoop. http://hadoop.apache.org/ (Visited: February 2015).
[7]http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction. (Visited: February 2015).
[8] M. Hammoud, M. F. Sakr.Locality-Aware Reduce Task Scheduling for MapReduce,Cloud Computing Technology and Science, 2011 IEEE Third International Conference, 2011, pp. 570-576.
[9]K. Morton, A. Friesen, M. Balazinska, and D. Grossman, “Estimating the progress of mapreduce pipelines,” in Proceedings of IEEE International Conference on Data Engineering (ICDE), 2010.
[10]Tian F, Chen K. Towards optimal resource provisioning for running MapReduce programs in public clouds[C]//Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011: 155-162
[11]Chen J, Wang D, Zhao W. A Task Scheduling Algorithm for Hadoop Platform [J]. Journal of Computers, 2013, 8(4).
[12]Hadoop’s Fair Scheduler http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html
[13] Hadoop’s Capacity Scheduler: http://hadoop.apache.org/core/docs/current/capacity_scheduler.html.
[14]Lee G, Chun B G, Katz R H. Heterogeneity-aware resource allocation and scheduling in the cloud[C]//Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud. 2011, 11.
[15] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In EuroSys ’10: Proceedings of the 5th European conference on Computer systems, pages 265–278, New York, NY, USA, 2010. ACM.
[16]Jin J, Luo J, Song A, et al. BAR: an efficient data locality driven task scheduling algorithm for cloud computing[C]//Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2011: 295-304.
[17]K. Kc and K. Anyanwu. Scheduling Hadoop Jobs to Meet Deadlines, Proc. CloudCom, 2010, pp.388-392.
[18]Verma A, Cherkasova L, Campbell R H. Resource provisioning framework for mapreduce jobs with performance goals. Middleware 2011. Springer Berlin Heidelberg, 2011: 165-186.
[19] Verma A, Cherkasova L, Kumar V S, et al. Deadline-based workload management for mapreduce environments: Pieces of the performance puzzle[C]//Network Operations and Management Symposium (NOMS), 2012 IEEE. IEEE, 2012: 900-905.
[20] Phan L T X, Zhang Z, Loo B T, et al. Real-time MapReduce scheduling. http://www.utdallas.edu/~cxl137330/courses/spring14/AdvRTS/protected/slides/26.pdf , 2010.
[21] Cho B, Rahman M, Chajed T, et al. Natjam: Design and evaluation of eviction policies for supporting priorities and deadlines in mapreduce clusters[C]//Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013: 6.

[22]Mark Yong, Nitin Garegrat, Shiwali Mohan. Towards a Resource Aware Scheduler in Hadoop. Proc. ICWS, 2009, pp:102-109
[23]Polo J, Castillo C, Carrera D, et al. Resource-aware adaptive scheduling for mapreduce clusters. Middleware 2011. Springer Berlin Heidelberg, 2011: 187-207.
[24] Zhang Z, Cherkasova L, Loo B T. Optimizing Cost and Performance Trade-Offs for MapReduce Job Processing in the Cloud[C]//Proc. of IEEE/IFIP NOMS. 2014.
[25] L.Mashayekhy, M.M.Nejad,D. Grosu, et al. Energy Efficient Scheduling of MapReduce Jobs for Big Data Applications. IEEE Transactions on Parallel & Distributed Systems 1402.2810, 2014.
[26] Palanisamy B, Singh A, Liu L, et al. Cura: A Cost-optimized Model for MapReduce in a Cloud. Parallel & Distributed Processing (IPDPS), IEEE 27th International Symposium on. IEEE, 2013: 1275-1286
[27] Wang, Zhe, et al. "A New Schedule Strategy for Heterogenous Workload-aware in Hadoop." China Grid Annual Conference (China Grid), 2013 8th. IEEE, 2013, pp. 80-85.
[28] Mude R G, Betta A, Debbarma A. Capturing Node Resource Status and Classifying Workload for Map Reduce Resource Aware Scheduler. Intelligent Computing, Communication and Devices. 2014: 247-257.
[29] Zhang Q, Boutaba R. Dynamic workload management in heterogeneous Cloud computing environments[C]//Network Operations and Management Symposium (NOMS), 2014sz IEEE, 2014: 1-7.
[30] Tao Y C, Shi L. Optimization of MapReduce Performance in Heterogeneous Resource Environments. Journal of Chinese Computer Systems, 2013, 34(2): 287-292
[31]Chen Q, Zhang D, Guo M, et al. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment[C]//Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. IEEE, 2010: 2736-2743.