A Hadoop Job Scheduling Model Based on Uncategorized Slot

Tao Xue and Ting-ting Li
Department of Computer Science, Xi'an Polytechnic University, Xi'an 710048, China
Email: xt73@163.com; 810978269@qq.com

Abstract —As Job scheduling is becoming an important part of Hadoop framework at present, a job scheduling model based on uncategorized slot was researched on this paper. It could eliminate the limitation of Job Task type and didn't distinguish between Map slot and Reduce slot any more, however there was only one type of slot left which could be assigned to execute the Map tasks and to run the Reduce tasks. By adopting Reduce dynamic partitioning, it can realize switching smoothly the slot between two types of tasks, meanwhile, compared with the FIFO algorithm which need distinguish the type of slot, the experimental result shows that the model not only improves the resources utilization and betters load balancing, but also enhances the parallelism of tasks and shortens the execution time of the Job Tasks.

Index Terms—Dynamic partitioning algorithms, Hadoop, job scheduling algorithms, MapReduce

I. Introduction

With the high-speed development of the Internet and the rapid growth of customer number, the amount of data growth rate also presents exponential growth, which makes "Big Data" become the most popular search term currently. Facing such a large amount of data, the traditional approach cannot handle it with the aid of existing tools or techniques. Along with the rapid development of software technology, distributed platforms have sprung up. In numerous distributed processing platforms, Apache Hadoop [1], due to the powerful distributed computing framework of MapReduce, is by far the most active and widely used platform. By adopting the idea of divide and conquer, MapReduce is a kind of low cost and high efficiency approach. It connects a large number of low-cost PC each other formed a cluster to deal with mass data storage and analysis. In a nutshell, the traditional processing is replaced by distributed processing in the field of big data processing.

Hadoop platform hides the underlying implementation details for developers. In this platform, you only need to write the Map and the Reduce functions according to the demand, which greatly simplifies the development of distributed applications. As a result, many cloud operators began using Hadoop, which makes the number of users and active frequency constantly rising; therefore the performance of Hadoop platform promotion is more and more important. In the study of Hadoop, job scheduling is always a key point of the research. This article puts forward a new kind of Hadoop job scheduling model.

The remainder of this paper is organized as follows: Section II reviews the related work. Section III describes the algorithm and implementation of job schedule model based on uncategorized slot. Experiment and performance analysis are studied in Section IV. Section V, concludes the work of this paper and outlines the future work.

II. Related Work

In order to simplify the resources allocation, Hadoop adopts the model based on the slot resources. Slot is the basic unit of the mission in the MapReduce, which maps from the multi-dimensional resources (CPU, MM, IO, etc.) to one dimensional in TaskTracker node. Although there has been many scheduling algorithms, none have unified resource allocation model. When the Map tasks run, it takes up the Map slot. While the Reduce tasks execute, it occupy the Reduce slot. In other words, the two different types of slot occupy their own resources respectively and no sharing consideration. On TaskTracker node configuration, each node has two Map slots and one Reduce slot by default, and the Map slot is only responsible for the Map tasks and the Reduce slot is only responsible for executing the Reduce task, which largely decrease the task parallelism and resources utilization.

Hadoop platform possesses three algorithms of scheduling, FIFO Scheduler [2], Fair Scheduler [3] and Capacity Scheduler [4]. The default scheduling algorithm is FIFO in this platform. In addition, numerous of scheduling algorithms are coming into being. For example, Delay Scheduler [5], which is based on enhancing data locality; Dynamic Proportional Scheduler [6], which is based on user preferences and changing the way of task allocation proportion and dynamic priority; Constraint-Based Scheduler [7], which is based on deadline of Real-time; [8] proposed a scheduling algorithm which considers when the task scheduler can't choose the Data-local task whether it allows to assign the Non-local task or not; [9] proposed a scheduling algorithm

Manuscript received May 24, 2015; revised September 23, 2015.
This work was supported by the China's National Development and Reform Commission (NDRC) and High-tech industrialization projects (Shaanxi NDRC [2009] no. 1365); Xi'an Polytechnic University doctoral research start-up fund (BS0725).
Corresponding author email: 810978269@qq.com
doi:10.12720/jcm.10.10.778-783

which is based on the number of Map task node and data sheet replication mode; LATE [10] (Longest Approximate Time to End) and SAMR [11] (Self-adaptive MapReduce Scheduling Algorithm), which are based on under heterogeneous environment how to improve the scheduling efficiency; [12] and [13] proposed a scheduling algorithm which are based on job types classifying and the load dynamic of heterogeneous; [14] proposed a scheduling algorithm which is based on the adaptive node capacity; [15] proposed a scheduling algorithm which is based on matching rules; beyond above, the scheduler based on the intelligent algorithm and simulated annealing algorithm [16] and artificial fish algorithm [17] and genetic algorithm [18]; etc. however all of these scheduling algorithm don't consider from the resources allocation model, their task scheduling strategy is to distinguish the slot type.

This paper presents a scheduling model based on uncategorized slot, using only one type of resource slot. The slot can be either assigned to run the Map task or the Reduce task, improving the utilization rate of slot and tasks parallelism and resources utilization.

III. Scheduling Model Based on Uncategorized Slot

In the scheduling model based on uncategorized slot, resources model expressed by one kind of slot, which can run both of the Map tasks and Reduce task, specific allocate what kind of slot according to the execution of job progress and the task scheduler as shown in Fig. 1.

The key problem of slot scheduling model based on uncategorized slot is how to make the slot in smooth switching between two different types of tasks. Because the demand resources for the Map tasks and the Reduce tasks are different, so we need to design the unified slot resources representation model. According to the above reasons we need to do following four steps: 1) scheduling model related definitions; 2) slot resources model representation; 3) reduce dynamic partitioning; 4) scheduling algorithm implementation.

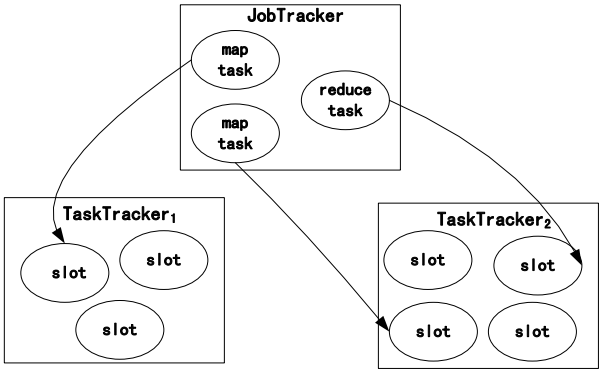

Fig. 1. Task allocation model based on uncategorized slot.

A. Scheduling Model Related Definitions

Because of the Reduce task input source data is from the Map phase output data, so the Reduce phase has data dependencies with the Map phase. When the Map task process is completed, the data buffer storage to the local disk to next sorting work. In the scheduling model to distinguish the slot type, *R* is a static value and represents the number of Reduce slots. The partition function divided the data into *R* pieces of data blocks then *R* represents the number of Reduce Tasks and each Reduce task shuffling middle results of data submitted to the corresponding Reduce function. The MapReduce data flow is shown in Fig. 2.

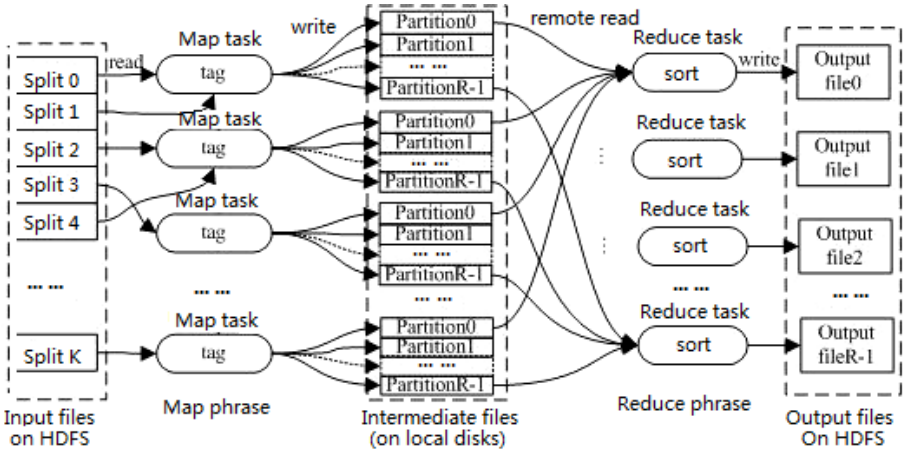

Fig. 2. MapReduce data flow.

In the uncategorized slot model, the initial value of *R* is 1. Because the value of *R* will change along with the number of slots assigned to the Reduce task, this will change the direction of data flow which was generated by the Map task. In order to balance each direction of data flow and data skew situation doesn't appear, so we need to get resources information and data through Ganglia in job runtime. Ganglia is an open source cluster monitoring software, which can monitor and display various kinds of status information in the cluster nodes, such as CPU, Memory, hard disk utilization, I/O workload, Network traffic situation, etc. Because Ganglia does little to the consumption of computing resources and which is fully compatible with Hadoop, so deploy RRDTool and gmetad

which summary of cluster resources on the JobTracker node and deploy gmond which collection of each node resources on the TaskTracker node is very convenient.

We can define the concept of the rate ratio σ of Map and Reduce from the information collected above. Getting the S_m which is total internal data output per unit time in the Map phase and the S_r which is the amount of data processed per unit time in Reduce phase, so σ is the speed ratio. Parameter σ is the key determinants of the slot whether to execute the Map tasks or the Reduce tasks.

$$\sigma = \frac{S_r}{S_m} \tag{1}$$

In case the intermediate data to be covered by running too many Map tasks or processing speed too slow by running too many Reduce tasks, we have to define two variable parameters, which are the range of σ, *LowLimit* (*LL*) [19] and *UpperLimit* (*UL*), through the two parameters to adjust the number of slot which execute different types of tasks.

B. Slot Resources Model Representation

In the process of task execution, not only the Map phase and the Reduce phase demand for different resources, but also the child in the process demands for different resources too at same time. Such as, copy the block of data mainly using the network I/O and disk I/O resources at the beginning of the Reduce phase and in the process of the shuffle. Then the CPU is the main use of resources in the process of the Reduce phase. So we can control the resources by changing various parameters.

In the following three equations, set *R* represents the model of the slot with five dimensions group resources, ρ_i represents the elements in the resources model, ω_i represents corresponding element weight value of resources.

$$R = \{CPU, MM, Disk, IO, DiskIO\} \tag{2}$$

$$slot = \sum_{i=1}^{5} \omega_i \rho_i \tag{3}$$

$$\sum_{i=1}^{5} \omega_i = 1 \tag{4}$$

ω_i 's value will dynamic change in the different phase of job execution along with the historical statistics of JobTracker and the current implementation. Here introduce slot Dynamic Stability Control (DSC) [19], which adjust the number of TaskTracker slots according to the node load and the ω_i value dynamically.

C. Reduce Dynamic Partitioning

In the same job, the data dependencies between Map tasks and Reduce tasks mainly displays in the middle of the block data block replication. The Reduce task can be scheduled after the completion of a certain percentage of the Map tasks, so in uncategorized slot model task execution is still divided into Map phase and Reduce phase.

When a slot has been processed the Map task then request task scheduler assigned a new task, the scheduler allocate the slot to Map task or Reduce task according to the work of implementation. If the node allocated Reduce task slot, the slot went into the Reduce phase, which directly cause the *R* value and the direction of intermediate data flow change.

For each task, the first step is sampling {*key*, *value*} of the map data and establishing an index tree B-tree, as shown in Fig. 3. Not a leaf node represents the *key* "bytepath", while each leaf node represents the former two *key* "bytespath", and the range of the partition number is saved in the leaf node, at the mean time, the same prefix *key* was assigned to the same leaf node.

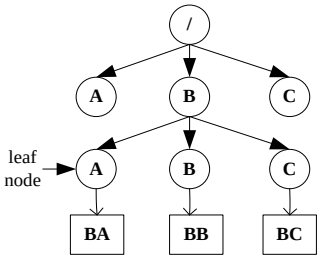

Fig. 3. A partition index tree.

For a *key*, the process of partition: 1) judge the first byte of the *key*, then find the not leaf node of first layer; 2) according to the second byte of the *key*, each leaf node may correspond to multiple sampling (i.e., multiple reducer), then compare with each samples one by one and determine which reducer can be assigned to.

In the case of not distinguish the slot, the initial value of *R* is 1, and its value will change along with the number of Reduce tasks. Here is the design of dynamic Hash function *H(R)*:

$$H(key) = key \quad MOD \quad R \tag{5}$$

The value of *R* is synchronous, which can change with the number of distribution to the Reduce task, so the number of partitions and range will change accordingly. The whole model is divided into four stages: initial, split, merge and recovery stage.

The initial stage: sampling the data in the process of Map task execution, and according to the sample point establish B-index tree. After a certain percentage of the Map task, framework automatically assigned to the Reduce task. Due to the initial value of *R* is 1, so there is only one slot executing Reduce task, in other words, all the sampling are mapped to the interval {a, z} of this reducer, as shown in Fig. 4:

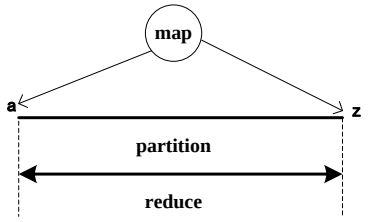

Fig. 4. The initial partition.

The split phase: with the development of task execution, the amount of data have increased in the Map phase and the accumulation of intermediate data, forcing us to add R_x Reduces, at this time R_x slots will be selected to execute Reduce tasks. In order to achieve the copy of the intermediate data block, the scheduler will add R_x sample points and then split the original partition. The selection of sampling points is on the basis of the sorted middle data and then divided into R_x equally, thus tipping point is the new sample point. As shown in Fig. 5, three new sample points d, h, t was added, and the original interval was divided into four parts {a, d}, {d, h}, {h, t}, {t, z}, as a result each interval corresponding a piece of intermediate data. Starting a Reduce task takes up a slot, at the mean time, copy the Map results in the corresponding section of the data, in this way scheduler switches a part of Map tasks to the Reduce task.

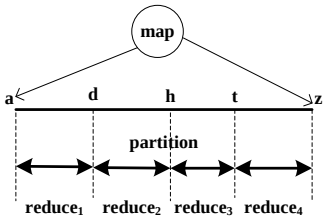

Fig. 5. Divide partition.

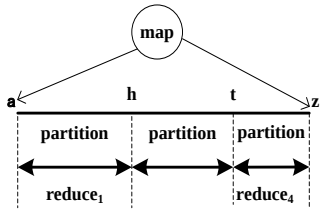

Fig. 6. Partition merge.

The merge phase: when the Reduce phase process to be finished or σ larger than the *UL*, slot will be assigned to a Map task or other Reduce task, result in the current operation of slot will reduce accordingly, so we need to decrease the sampling points and merge interval data. As shown in Fig. 6, when removing the sample point d, reducing *R* to 3, merging interval {a, d} and {d, h} to interval {a, h}, corresponding reduce will mapped to a new interval period of {a, h} block of data, these lead to a slot to be released the request of other tasks.

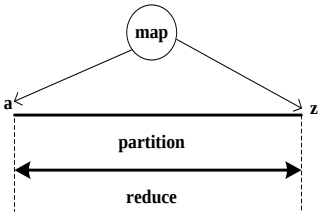

Fig. 7. Partition recovery.

The recovery phase: as the last job execution is completed, sample point is reduced to zero, and *R* is reset to the initial value 1, so all small intervals merge into a big interval, as shown in Fig. 7, the slot will deal with the final finishing operation of Reduce phase.

D. Implementation

As scheduling model has been defined in the previous, the last problem is the timing of starting the Reduce task. Due to the similarity of data in the same job, so that we can assess the rest tasks by the resources which have been operated the missions.

Using the ceiling *LL* and lower limit *UL* of parameter σ, if σ large than *UL*, for the current slot allocation reduce task, if σ lower than *LL*, for the current slot allocation map task, somewhere in between, continued in accordance with the distribution of task type before. Flowchart is shown in Fig. 8.

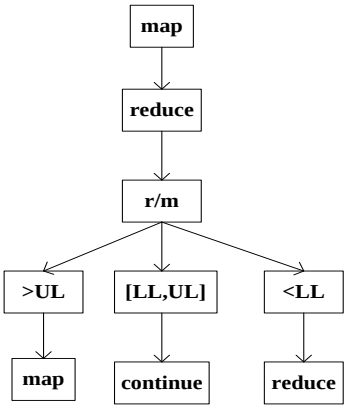

Fig. 8. flowchart of entirety Task allocation.

Algorithm1: scheduling based on uncategorized slot
Begin
1: Determine the number of slots according to the node load and the value of ω_i;
2: Allocate the Map task for all the slots;
3: **If** (Map phase completed more than set threshold or σ less than the *LL*)
{
4: Allocate the reduce task;
5: Select the free slot;
6 *R* + 1, add the sampling points, dynamic partitioning division;
}
7: **else if** (σ is greater than the *UL*)
{
8: Allocate the Map task;
9: Select the free slot;
10: *R* - 1, reduce the sampling point, dynamic partitioning merge;
}
11: **else** (σ between *LL* and *UL*)
12: **continue**;
13: **end**

IV. Experiment and Results

E. Experimental Environment

Experimental environment is a Hadoop cluster, which is made up of nine compute nodes and these nodes distribution on the two hosts, one containing the name node and four data nodes, the other include the remaining nodes. Two hosts via gigabit switches interconnected, as

shown in Table I, Table II and Table III.

TABLE I: EXPERIMENT ENVIRONMENT

Version	Hadoop1.1.2
File system	HDFS
Test program	Sort program
Scheduling model	FIFO/uncategorized slot model
Data	Use RandomWriter randomly generated

TABLE II: JOBTRACKER NODE CONFIGURATION

CPU	Intel Xeon(R)CPU E5-2620 @2.00GHZ×2
OS	Ubuntu12.04
Kernel	Linux 3.2.0-60-generic
Memory	4G

TABLE III: TASKTRACKER NODE CONFIGURATION

CPU	Intel Xeon(R)CPU E5-2620 @2.00GHZ×2
OS	Ubuntu12.04
Kernel	Linux 3.2.0-60-generic
Memory	2G

The Ganglia installed in the cluster is to monitor each TaskTracker node and the JobTracker node, meanwhile, according to the monitoring statistics [19], it will set the *UL* to 0.8 and set the *LL* to 0.6 and ω_i initial set to 0.2.

F. Results Analysis

At first two models for the utilization of resources by the number of slots in the process of job execution are measured. As a contrast, to distinguish the slot in the model, we choose the most typical and widely used FIFO algorithm.

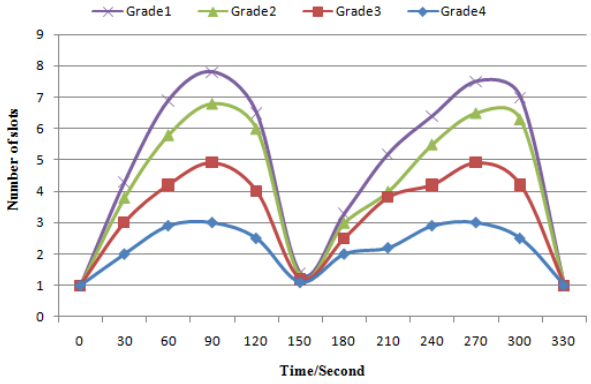

Fig. 9. Slot number on FIFO model.

The slots number varies with job executing under different size of clusters as shown in Fig. 9 and Fig. 10. The "*Grade x*" represents different scale of data. Fig. 9 illustrates at the Map phase of one task, the number of Map slots increase at first and then decrease, when the Map task execution is finished, Map slots decrease to initial value, and then reduce phase start and the trend is as same as the above. Obviously the Map phase and the Reduce phase resources is under a state of overload, so the resources load in the whole process is not balanced, while in the two stages of cohesion resources utilization rate is very low.

Fig. 10 shows the slots number changes in the uncategorized model. In Fig. 10, during the entire process of execution, the change tendency of the slot number is different from Fig. 9 under different scale of data, while the fluctuations are in a small range, until the completion of job scheduling and the recovery of the initial value of the slot number.

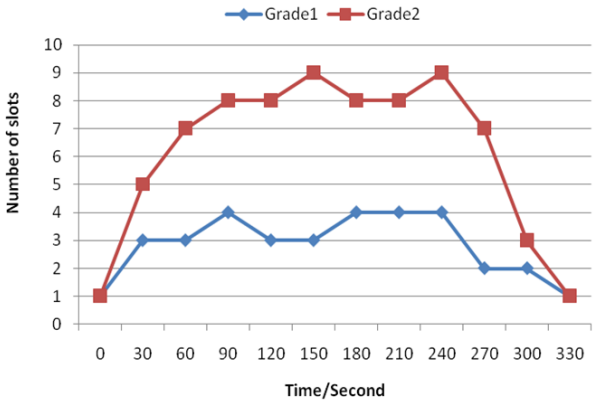

Fig. 10. Slot number on uncategorized slot model.

Uncategorized slot improves resources utilization can also be reflected in the improvement of job execution time. Under the different data size (MB), two kinds of scheduling model execution time contrast in the sorting test has been shown in Fig. 11. In the proposed scheduling model, job selection strategy is consistent with the FIFO, except no distinguish slot type. In Fig. 11 we can see uncategorized scheduling model is faster than the FIFO scheduling in execution time. This is because when uncategorized scheduling model is scheduling with higher parallelism of tasks, more slots can be participate in the task, thus improving the execution efficiency is inevitable.

Some Common Mistakes

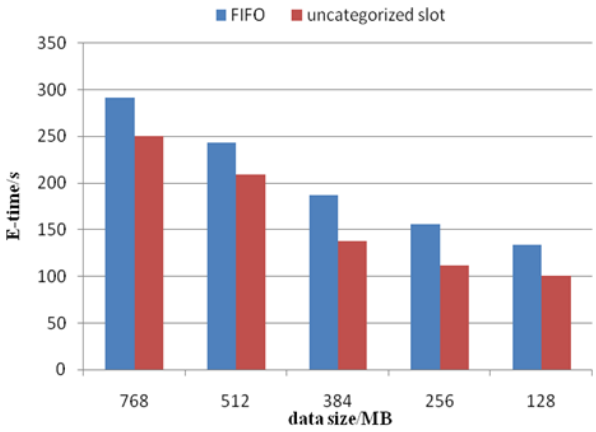

Fig. 11. execution time contrast in two models.

V. CONCLUSION

This paper proposes a Hadoop job scheduling model based on uncategorized slot. This model can eliminate the limitation of the slot on the map tasks and reduce tasks in task allocation, which makes the slot assigned to any types of task and it not only can improve the resources utilization and task parallelism, but also shorten the execution time of the operation.

For future work, we will continue on studying the performance stability of module Job scheduling as well as

the other modules in our uncategorized slot scheduling. The system level parallelization and scheduling also will be studied.

REFERENCES

[1] Apache Hadoop, Hadoop. (May 2015). [Online]. Available: http://hadoop.apache.org

[2] Chen He, Y. Lu, and D. Swanson, "Matchmaking: A new mapreduce scheduling technique," in *Proc. IEEE Third International Conference on Cloud Computing Technology and Science*, Nov. 2011, pp. 40-47.

[3] M. Zaharia, D. Borthakur, and J. S. Sarma, "Job scheduling for multi-user MapReduce clusters," EECS Department, University of California, 2009.

[4] A. Raj, K. Kaur, and U. Dutta, "Enhancement of hadoop clusters with virtualization using the capacity scheduler," in *Proc. Third International Conference on Services in Emerging Markets*, 2012, pp. 50-57.

[5] M. Zaharia, D. Borthakur, and J. S. Sarma, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proc. 5th European Conference on Computer Systems*, 2010, pp. 265–278.

[6] T. Sandholm and K. Lai, "Dynamic proportional share scheduling in Hadoop," *Job Scheduling Strategies for Parallel Processing*, pp. 110-131, 2010.

[7] K. Kamal and K. Anyanwu, "Scheduling Hadoop jobs to meet deadlines," in *Proc. IEEE International Conference on Cloud Computing Technology and Science*, 2012, pp. 388–392.

[8] X. Zhang, Y. Feng, and S. Feng, "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments," in *Proc. International Conference on Cloud and Service Computing*, 2011, pp. 235-242.

[9] Ibrahim, "Maestro: Replica-aware map scheduling for MapReduce," in *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012, pp. 435-442.

[10] S. Selvalakshmi and S. Vinodkumar, "Heuristic based heterogeneous longest approximate time to end algorithm of scientific workflows in computational clouds," *International Journal of Research in Advent Technology*, vol. 2, no. 10, pp. 2321-9637, Oct. 2014

[11] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "SAMR: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment," in *Proc. IEEE International Conference on Computer and Information Technology*, 2010, pp. 2736–2743.

[12] S. Patil and S. Deshmukh, "Survey on task assignment techniques in Hadoop," *International Journal of Computer Applications*, pp. 15-18, 2012.

[13] J. Dhok, N. Maheshwari, and V. Varma, "Learning based opportunistic admission control algorithm for mapreduce as a service," in *Proc. 3rd India Software Engineering Conference*, 2010, pp. 153-160.

[14] W. Kai and Z. Hou, "A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds," *Journal of Grid Computing*, vol. 10, no. 3, pp. 521-552, Sept. 2012.

[15] C. Tian, H. Zhou, and Y. He, "A dynamic MapReduce scheduler for heterogeneous workloads," in *Proc. Eighth International Conference on Grid and Cooperative Computing*, Aug. 2009, pp. 218-224.

[16] E. Torabzadeh, "Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop," *Advances in Engineering Software*, pp. 1243-1258, 2010.

[17] G. S. Sadasivam and D. Selvaraj "A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids," in *Proc. 2nd World Congress on Nature and Biologically Inspired Computing*, Dec. 2010, pp. 377-382.

[18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, pp. 755-768, Apr. 2011.

[19] H. Mao, S. Hu, Z. Zhang, L. Xiao, and L. Ruan, "A load-driven task scheduler with adaptive DSC for MapReduce," in *Proc. IEEE/ACM International Conference on Green Computing and Communications*, Aug. 2011, pp. 28-33.

Tao Xue was born in Shaanxi Province, China, in 1973. He received the B.S. degree in mechanical engineering from the Xi 'an Jiaotong University of China (XJTU), Xi 'an, in 1995 and the M.S. degree in computer science from the Northwest University of China (NWU), Xi 'an, in 2000 and the Ph.D. degree in computer software and theory from the XJTU, in 2005. He is currently an Associate Professor in the Computer Science Dept at Xi'an Polytechnic University. His research interests include Cloud Computing, Big Data and Content-based Networking.

Ting-ting Li was born in Zhejiang Province, China, in 1990. She received the B.S. degree in network engineering from the Xi'an Polytechnic University of China, Xi 'an, in 2013. She is currently pursuing the M.S. degree with the Department of Computer Science, Xi'an Polytechnic University. Her research interests include Cloud Computing, Big Data and Content-based Networking.