Copyright © 2018 Tech Science Press TSP, vol.1, no.1, pp.1-5, 2018

An Optimized Resource Scheduling Strategy for Hadoop Speculative Execution Based on Non-cooperative Game Schemes

Yinghang Jiang[1], Qi Liu[2, 3*], Williams Dannah[1], Dandan Jin[2], Xiaodong Liu[3], Mingxun Sun[4, *]

Abstract: Hadoop is a well-known parallel computing system for distributed computing and large-scale data processes. "Straggling" tasks, however, have a serious impact on task allocation and scheduling in a Hadoop system. Speculative Execution (SE) is an efficient method of processing "Straggling" Tasks by monitoring real-time running status of tasks and then selectively backing up "Stragglers" in another node to increase the chance to complete the entire mission early. Present speculative execution strategies meet challenges on misjudgement of "Straggling" tasks and improper selection of backup nodes, which leads to inefficient implementation of speculative executive processes. This paper has proposed an Optimized Resource Scheduling strategy for Speculative Execution (ORSE) by introducing non-cooperative game schemes. The ORSE transforms the resource scheduling of backup tasks into a multi-party non-cooperative game problem, where the tasks are regarded as game participants, whilst total task execution time of the entire cluster as the utility function. In that case, the most benefit strategy can be implemented in each computing node when the game reaches a Nash equilibrium point, i.e. the final resource scheduling scheme to be obtained. The strategy has been implemented in Hadoop-2.x. Experimental results depict that the ORSE can maintain the efficiency of speculative executive processes and improve fault-tolerant and computation performance under the circumstances of Normal Load, Busy Load and Busy Load with Skewed Data.

Keywords: Distributed computing, speculative execution, resource scheduling, non-cooperative game theory.

1. Introduction

In recent years, from the pace of Internet information technology to the booming trend of

---

[1] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science & Technology, Nanjing, 210044, China.

[2] School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China.

[3] School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh EH10 5DT, UK

[4] Centre for Health Sciences Research, University of Salford, Salford, Greater Manchester, M5 4WT, UK

[*]Corresponding Author: Qi Liu, Mingxun Sun. Email: q.liu@napier.ac.uk, m.sun@salford.ac.uk.

e-commerce, Internet data information has been rapidly expanding, the big data storage and processing platform emerged as the times require [Zafar, Khan, Malik et al. (2017); Hashem, Yaqoob, Anuar et al. (2015); Lee (2013)]. It was hailed as yet another technological revolution in the computer industry following the Internet of Things and Cloud Computing, and it also made the storage and analysis programs based on big data a research hot-spot in academic circles both at home and abroad [Kong, Zhang, Ye et al. (2017); Kong, Zhang and Ye (2015); Stergiou and Psannis (2017)]. As the mainstream open source processing framework for big data, the Hadoop platform has been widely used due to its ease of use. At present, it has become a top project of Apache and has been used by some large Internet companies for customization [Manikandan and Ravi (2014)]. Hadoop was originally part of the open source project “Nutch”, which is implemented by Doug Cutting with reference to Google’s distributed big data storage model “GFS” and distributed parallel computing model “MapReduce” [Ghemawat, Gobioff and Leung (2003); Dean and Ghemawat (2008)]. With the continuous improvement and development of the Hadoop platform, many applications based on HDFS and MapReduce are becoming more and more abundant, such as HBase [Apache hive (2018)] and Hive [Bhupathiraju and Ravuri (2015)] etc., which aim at improving the performance of the cluster and allow people to store and process data more easily. However, these applications are based on the Hadoop distributed storage framework “HDFS” [Chang, Dean, Ghemawat et al. (2008)] and the computing framework “MapReduce” [Dean and Ghemawat (2008)].Therefore, many famous IT companies like Microsoft, Yahoo!, Google, Amazon, have launched their own big data storage and computing platforms such as Storm [Toshniwal, Taneja, Shukla et al. (2014)], Spark [Zaharia, Chowdhury, Franklin et al. (2010)], Dryad [Isard, Budiu, Birrell et al. (2007)], and let the development of big data platform optimization technology as the core development trend in the future [Storey and Song (2017)].

Just as the storage and CPU constitute the bottom of the computer, the distributed storage system HDFS and the distributed computing framework MapReduce also constitute the bottom layer of the Hadoop distributed platform. Therefore, the performance of HDFS and MapReduce directly affects the overall performance of the Hadoop cluster, such as job execution time, cluster throughput, etc. [Glushkova, Jovanovic and Abello (2017)]. MapReduce 2.0 is the core component of the Hadoop ecosystem, although its performance has been greatly improved compared to MapReduce 1.0, the gradual increase in the amount of data has also caused its performance bottleneck. Especially dealing with PB-level data when the data is skewed and the node processing efficiency is low, there will be a certain type of task running at a significantly slower speed than other tasks or even more than five times the average task time, which will slow down the overall running time of the cluster and greatly reduce resource utilization [Naik, Negi and Sastry (2014)]. Hadoop speculative execution mechanism is an efficient way to reduce the impact of this kind of “straggler” on cluster performance. It judges and discovers slow tasks through certain conditions, once a task is confirmed as a “straggler”, Hadoop will start a backup task for it, when the backup task is completed before the original task, the original task is killed, which can reduce the cluster running time and improve the fault-tolerant performance of Hadoop.

The original speculative execution mechanism in Hadoop was called “Hadoop-Naive”, but its performance is poor in heterogeneous environments due to it uses task progress to determine whether the task is a “Straggler”, so many researchers began to optimize the SE from any other aspects and several optimized strategies are proposed [Liu, Jin, Liu et al. (2016)], Such as LATE, MCP, ERUL and so on. These determinations of the “straggler” tasks in the proposed strategy are based on the estimation of the remaining time of the real-time task, but the inaccurate estimation will lead to improper node allocation. At the same time, if there exist multiple “Stragglers” in the cluster, speculative execution performance will greatly affect the overall performance of the cluster, which implies the scheduling strategy of the backup task is very important. Based on the whole study of speculative execution, we proposed an Optimized Resource Scheduling model for Speculative Execution based on a non-cooperative Game theory (ORSE) that introduced the idea of game theory. In the ORSE algorithm, the resource scheduling model of the backup task in execution is transformed into a classic multi-party non-cooperative game problem, the game participants are the backup task group and the game strategies are the node in the cluster, the game’s utility function is the cluster’s overall task execution time, and finally when the game reaches the Nash equilibrium, the task scheduling scheme will be obtained.

2. Related works

The ref. [Liu, Jin, Liu et al. (2016)] lists the three core components in the speculative execution strategy:

- Finding out “straggler” tasks during they are running;
- The selection of a suitable backup node;
- Make sure that the benefit of starting the backup task to the cluster is greater than not enabling it.

Hadoop considered the three components at the beginning of the design, it implied the original speculative execution strategy in Hadoop, which is called “Hadoop-Naive”. Since Hadoop-Nave shows many deficiencies in the heterogeneous cloud environments, Zaharia et al. first proposed the heuristic speculative execution strategy called LATE, this strategy uses the remaining execution time of the task as the priority for the determination of the “straggler” tasks, and also considers the proper backup node [Cheng, Rao, Guo et al. (2014)]. The LATE strategy has been optimized to a certain extent relative to the Nave strategy, but many problems have been found in the application process, such as the estimation error of the task’s remaining time and not considering the impact of real-time workload on task execution. Therefore, the literature [Zhang, Zhang, Li et al. (2016)] further proposes a heuristic strategy “ERUL” by finding the linear relationship between system load and task remaining time. The MCP strategy is proposed [Chen, Liu and Xiao (2014)], which maximizes the benefits of the cluster by establishing the maximum cluster performance model and guarantee that the backup task gains more benefit to the cluster than the original task, but the model does not consider the value of the node itself when calculating the benefits. The Ex-MCP strategy is the optimization of the MCP strategy, which takes the node value into the benefit model [Wu, Li, Tang et al. (2014)].

In addition to these strategies such as LATE, MCP and ex-MCP, domestic and foreign researchers have also conducted research and exploration of speculative execution strategies from different aspects and proposed their own optimization plans. SSE (Smart Speculative Execution) is an optimization strategy based on the node classification which depends on the hardware performance of the node and the amount of computational data in the node [Liu, Cai, Fu et al. (2016)]. Wang et al. [Wang, Lu, Lou et al. (2015)] also proposed a Partial Speculative Execution (PSE), which uses the detection point of the original task to start the speculative execution without restarting the entire process, which enhanced the Hadoop performance. Since speculative execution is a classical space-for-time thinking, most optimization strategies ignore the storage space occupied by backup tasks. Therefore, Liu et al. propose a speculative execution strategy based on space-time optimization for multi-objection, the strategy optimized the load balancing problem during speculative execution based on extreme learning machine and multi-objective space-time optimization algorithm [Liu, Jin, Liu et al. (2016)]. SECDT is a new speculative execution algorithm based on the C4.5 decision tree, which estimates the completion time of the scheduled task based on the C4.5 decision tree [Li, Yang, Lai et al. (2015)]. The ATAS strategy improves the success rate of backup tasks by reducing the reaction time and quickly starting backup tasks [Yang and Chen (2015)]. The data skew of the data itself has always been one of the factors which result in “straggler” tasks, the Flexslot’s strategy can adaptively change the number of slots on each compute node to further mitigate the problem of data skew [Guo, Rao, Jiang et al. (2017)].

3. Model and algorithm

Hadoop resource scheduling refers to different scheduling of tasks to different nodes through a *ResourceManager* (RM). Traditional resource scheduling algorithms such as FIFO, Capacity Scheduler, etc. have the problem of low cluster utilization resulting in low performance of the cluster, so many scholars are committed to Various dimensions of resource scheduling optimization, various optimization strategies based on game theory ideas have also been proposed [Li (2016); Zhang and Zhou (2017)]. This chapter is different from the traditional resource scheduling optimization algorithm in that this paper speculates that the execution of the backup task and the tasks being executed and waiting on the node select different computing nodes through the game, and seeks the Nash equilibrium allocation scheme in the game, thereby improving the cluster computing node, the utilization rate. This article mainly adopts the non-cooperative game model, mainly because its mathematical theory model has been proved and extended many times, especially the existence problems of Nash equilibrium point [Czumaj and Cking (2002)].

The purpose of the game theory introduced in this chapter is to generate backup tasks generated by speculative execution without affecting the execution of normal tasks on cluster nodes. Backup tasks and original tasks can still be evenly distributed to the nodes of the cluster, thereby making greater use of cluster computing resources. The algorithmic model for the speculative execution of resource scheduling in this chapter is based on the following preconditions:

- there are several nodes in the cluster, and different tasks can select different nodes for execution;

- the ultimate goal of optimization is that the backup task can be executed at the fastest speed and the model needs to sense the current task status of the current cluster in real time;
- when the scheduling scheme is optimal, the benefit of the corresponding cluster is the largest;
- there are multiple backup tasks at the same time in the execution process;

In Yarn, users can customize the number of dropped tasks, so the number of backup tasks in Hadoop is limited. Second, a backup task can only be assigned to run on one node. The number of nodes in the cluster is also set in advance, resulting in backup tasks. The number of allocated nodes is limited; finally, when the task is assigned to a node, since the node's operating efficiency and the task's data volume are fixed, the task's running time is also determined, so the model is regular Limited game.

At the same time, there are two concepts in the non-cooperative game: hybrid strategy and Nash equilibrium. Hybrid strategy indicates that a certain task in the cluster has a certain probability when selecting the corresponding node. In the speculative backup task node scheduling, there are multiple tasks and multiple nodes, so each node has a certain probability to be assigned to a fixed node, in line with the hybrid strategy. Nash equilibrium refers to the game process, the game participants cannot obtain higher interests by changing the game strategy, and the core of the game theory introduced in this chapter is to find a balanced backup task node allocation scheme, making the task not to change its own operation the node achieves better benefits and there is also a Nash equilibrium solution. This model satisfies the non-cooperative game model.

In the Hadoop platform, tasks are submitted by users and the number of tasks is limited. It has been explained above that the game model in this chapter is a limited non-cooperative game. Therefore, according to the principle of existence of Nash equilibrium points in non-cooperative game theory, it can be known that there are limited non-cooperative. In the game model, there must be one or more Nash equilibrium points. In this case, the situation that the resource scheduling model has no solution in this paper is eliminated, thereby ensuring that there is a corresponding Nash equilibrium solution for each resource scheduling problem.

3.1. Design of a resource scheduling algorithm model based on a non-cooperative game

From the above it can be seen that this article uses a resource scheduling model based on a non-cooperative game. The input and output of this model under Hadoop speculation execution involve:

Input 1: A set of backup tasks generated by speculative execution. The game focuses on multi-participation. If there is only one task, it is equivalent to a unilateral optimization problem. Therefore, the model needs to have a group of backup tasks $(1, \ldots, i)$ as input.

This group of backup tasks also corresponds to the participants in the game model P. It is assumed that there are backup tasks in the model and the amount of data for each backup task is σ_n.

Input 2: Compute nodes in the cluster. Since there is a backup task, there must be a computing node in the cluster. The computing node corresponds to a game strategy S. Assume that there are computing nodes in the cluster, and the average execution rate of the node processing task is used as the execution rate of the node v_m.

Output: Total task execution time. The RM assigns the backup task to different compute nodes. At this point, the node needs to complete the backup task and the total execution time of the assigned original task on the node.

In the Hadoop platform, each task is assigned by the RM to each computing node in the cluster. Each task has a certain probability to be assigned by the RM to any computing node in the cluster. The probability that the task is assigned to the node k is p_i^k, and the workload of the task i is σ_i. Assuming that the task group assigned by the RM to the node k is denoted as M_k, then the execution time of the node is the ratio of the total task amount of the task to be processed on the node to the node execution efficiency, as shown in the following formula (1).

$$Execution\_Time_k = \frac{\sum_{i \in M_k} \sigma_i p_i^k}{v_k} \tag{1}$$

The model of resource scheduling for the backup task in the speculative implementation is a hybrid strategy and can be represented by this binary group (T, N), where T is the backup task set in the cluster, and N is the compute node set in the cluster. For each backup task t, the data volume is σ_t; whilst for each compute node n, the processing rate is v_n. Therefore, when the RM allocates the backup task t to the node k, the running time of the node k is also the sum of the execution time of the backup task plus the execution time of the original task group waiting on the node, which is shown in formula (2).

$$Time\_C\operatorname{os}t_t^k = \frac{\left(\sum_{i \neq t} \sigma_i p_i^k + \sigma_t\right)}{v_k} \tag{2}$$

Where v_k is the processing rate of the computing node k. p_i^k is the probability that the original task i is assigned to the node k. σ_i is the data quantity of each task in the task set to be completed on the node, and σ_t is the work amount of the backup task t allocated by the RM.

After the scheduler determines all the backup task allocation strategies, a single backup task cannot achieve higher benefits by changing its own computing node. At this point, the Nash equilibrium status of the non-cooperative game-based Hadoop speculative execution resource scheduling model is reached.

For this resource scheduling model, to make it reach the Nash equilibrium point of the game, RM cannot improve the benefit of the whole cluster by changing the scheduling strategy of any current task, then it must satisfy the following conditions, such as formula

(3) shows.

$$p_t^k \Rightarrow time\_\cos t_t^k = \min_{k\in N} time\_\cos t_t^k \tag{3}$$

Where p_t^k is the probability that task t is assigned to node k, and $time\_\cos t_t^k$ is the cost function that task t is scheduled by RM to the node k, which is shown in formula (2). N is the set of compute nodes in the cluster.

Therefore, the resource scheduling model for the speculative backup task can be transformed into a classic non-cooperative game problem. The participants of the game are the set of backup tasks, whilst the game strategy is the different computing nodes in the cluster, and the utility function of the game is the final completion time of the task. In a distributed cloud environment, the task scheduling optimization goal is that the task can be completed as soon as possible, so the game's utility function is the task's completion time; that is, if a task t is scheduled to a computing node k, then the cluster has a gain, whereas the income becomes the profit of task t on the node k. The individual task profit function is shown in formula (4), while the overall cluster's profit function is shown in formula (5).

$$Task\_\Pr ofit_{tk} = f\left(a_{tk}\right) \tag{4}$$

$$Cluster\_\Pr ofit_i = f\left(d_i\right) \tag{5}$$

Where a_{tk} is game strategy, which is the plan that task t dispatches to node k. d_i is the scheduling scheme i.

After the RM schedules backup tasks every time, the metric on the tasks is task completion time, whilst the metric on the scheduling strategy is the overall running time of the tasks on the cluster. These two metrics are calculated following formulae (6) and (7).

$$Task\_Profit_{tk} = f\left(\frac{\left(\sum_{i\in N_t^k}\sigma_i\right)+\sigma_t}{v_k}\right) \tag{6}$$

$$Cluster\_Profit_j = f\left(\max_{p\in M}\left(\frac{\left(\sum_{j\in M_p}\sigma_j\right)}{v_p}\right)\right) \tag{7}$$

Where in formula (6), σ_i is the workload of task i. N_t^k represents the original task set that are uncompleted in node k when backup task t is scheduled to node k. v_k is the execution rate of node k. σ_t is the workload of the backup task t that is scheduled to be on the node k during execution.

In formula (7), σ_j is the workload of task j. v_p is the execution rate of node p. M is the set of compute nodes in the cluster, and M_p represents the task sets assigned to node p in the entire cluster.

In the actual cloud environment, the cluster load is high (the number of nodes is smaller than the current task to be processed) or the cluster load is low. In this strategy, the RM generates a possibility execution node sets for the backup task generated during speculative execution, which are the highest-benefit computing nodes q for the backup task t, where q is the minimum value between the number of tasks and nodes; that is called the set of possible running nodes of the backup task, as shown in the formula (8).

$$Possible\_Node_t = \begin{cases} \arg\max\limits_{i=1}^{N} P_q & M > N \\ P_{i=1}^{M} & M \le N \end{cases} \tag{8}$$

Where $\arg\max_{i=1}^{N}\left(P_q\right)$ represents the most profitable computing nodes N of the backup task t. $P_{i=1}^{M}$ is M nodes in the cluster, whereas N is the number of backup tasks that need to be scheduled. At this time, due to the limited number of tasks and the number of nodes, the scheduling strategy for this backup task is a classical limited non-cooperative game problem. Therefore, according to the definition of non-cooperative games, there must be a Nash equilibrium solution, i.e. the Nash scheme. When there are intersections in the set of possible processing nodes for two or more backup tasks, these two become a conflicting task set, named $Conflict\_Tasks$, as is shown in formula (9).

$$Conflict\_Tasks = \overset{n}{\cup} Task_j \overset{n}{\cap} Node_j \neq \varnothing \tag{9}$$

3.2. Implementation and critical steps of the resource scheduling algorithm model

The overall flow chart of the Resource Scheduling Algorithm model is shown in Fig. 1, where critical steps of resource scheduling scheme on the purposes of speculative implementation are as follows.

- Step 1: The RM determines the number of backup tasks that need to be started based on the cluster resource operating status and the current number of "Straggler" queues in the cluster. At the same time, the RM can estimate the remaining time $time\_cost_i^k$ of backup task i on each node k and the benefit $Task\_Profit_{ik}$ of each task in each node;
- Step 2: Arrange the set of possible running nodes $Possible\_Node_i$ for each task in a descending order according to the $Task\_Profit_{ik}$.
- Step 3: If conflicts of two or more tasks happen in potential execution node sets, a $Conflict\_Tasks$ set is generated.
- Step 4: A non-cooperative game strategy is applied to $Conflict\_Tasks$ to find a Nash equilibrium solution, involving multiple scheduling schemes.
- Step 5: The most benefit scheme is determined from the Nash equilibrium solution according to $time\_cost$ and $Cluster\_Profit$.

- Step 6: According to step 5, the RM schedules backup tasks in the cluster. Corresponding nodes execute the tasks following the instruction of the $Application - Manager(AM)$.
- Step 7: Repeat Step 1-6 until all tasks are completed.

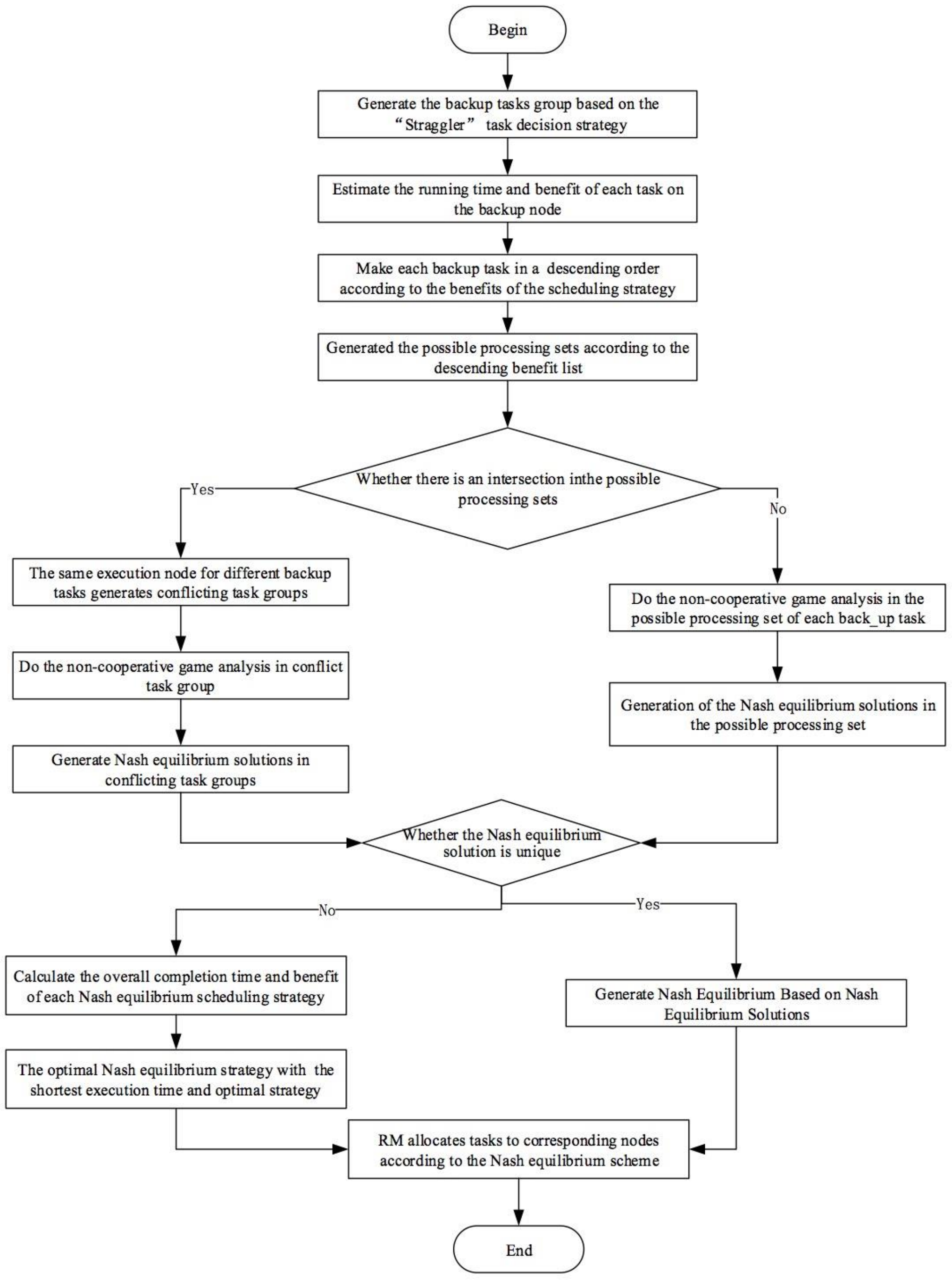
Begin
Generate the backup tasks group based on the "Straggler" task decision strategy
Estimate the running time and benefit of each task on the backup node
Make each backup task in a descending order according to the benefits of the scheduling strategy
Generated the possible processing sets according to the descending benefit list
Whether there is an intersection inthe possible processing sets
Yes
No
The same execution node for different backup tasks generates conflicting task groups
Do the non-cooperative game analysis in conflict task group
Generate Nash equilibrium solutions in conflicting task groups
Do the non-cooperative game analysis in the possible processing set of each back_up task
Generation of the Nash equilibrium solutions in the possible processing set
Whether the Nash equilibrium solution is unique
No
Yes
Calculate the overall completion time and benefit of each Nash equilibrium scheduling strategy
The optimal Nash equilibrium strategy with the shortest execution time and optimal strategy
Generate Nash Equilibrium Based on Nash Equilibrium Solutions
RM allocates tasks to corresponding nodes according to the Nash equilibrium scheme
End

Figure 1: Overall flow chart of ORSE strategy

4. Experiments and evaluation

In order to fully analyse the performance of the ORSE, a series of comparative experiments has been designed and conducted in a heterogeneous distributed environment, including job execution time, cluster throughput, and speculative execution accuracy. The experimental heterogeneous environment is mainly built on the servers in the lab, where eight nodes were created in the cluster, as shown in Tab. 1. A Hadoop-2.6 system was installed on them. LATE and MCP strategies have been deployed in the cluster to meet the contrasting experimental requirements. Each group of experiments was run five times in order to ensure the accuracy of the results, and then the performance comparison results were obtained.

Table 1: The detailed information of each node

NodeID	Memore (GB)	Core Processors
Node 1	10	8
Node 2	8	4
Node 3	8	1
Node 4	8	8
Node 5	4	8
Node 6	4	4
Node 7	18	4
Node 8	12	8

The data sets used in this experiment are all provided by Purdue University's performance testing benchmark suite including *WordCount* and *Sort*. Among them, *WordCount* input data volume is 50G, of which the Map task volume is 200, and the Reduce task volume is 16; the input data volume of *Sort* is 30G, of which the Map task volume is 200 and the Reduce task volume is 15.

4.1. Performance evaluation metrics

In this paper, major metrics are chosen for performance evaluation, including job execution time and cluster throughput.

- Task Execution Time: Task Execution Time is the completion time of a task, as an important indicator to indicate the performance of an optimized Algorithm in the Hadoop system.
- Cluster Throughput: Cluster Throughput is defined as the number of jobs that the cluster runs per unit of time.

Three scenarios are designed to examine potential performance of the ORSE strategy in a heterogeneous distributed environment, including Normal Load Scenario, Busy Load Scenario and Busy Load with Data Skew Scenario.

4.2. Performance of the ORSE strategy in the heterogeneous environment

Normal load scenario

In the normal load scenario, a low-load cluster has been configured with efficient resources. Using the original task initialization strategy in the Hadoop system, a file was split into several file blocks with the size of each block setting to 64MB and acting as a Map task. Data skew has been avoided by setting the input file size to be an integer multiple of 64MB. The execution time of each job and the cluster throughput running *WordCount* and *Sort* datasets are calculated, with experimental results shown in Fig. 2 and Fig. 3, which respectively illustrate the implementation of different strategies during the implementation of *WordCount* and *Sort*. As shown in Fig. 2, ORSE is different from LATE and MCP for the execution time of *WordCount* task. The degree of improvement is about 23.2% higher than that of LATE, which is about 6.1% higher than that of MCP. Compared with LATE, ORSE is improved by about 25.8% compared to LATE in terms of cluster throughput, which is about 9.7% higher than MCP. Similar trends can also be seen in the execution results of *Sort*. In the design process, ORSE is based on the LWR-SE's behind-task decision rule, which has optimized the prediction accuracy compared with LATE, MCP and other strategies using the average execution rate of the task to calculate the remaining time of the task. After the corresponding backup task is generated, the non-cooperative game model is used to find the possible execution nodes of each backup task group, and the node resources in the cluster are scheduled according to the Nash equilibrium scheduling scheme to ensure the execution efficiency of the backup task.

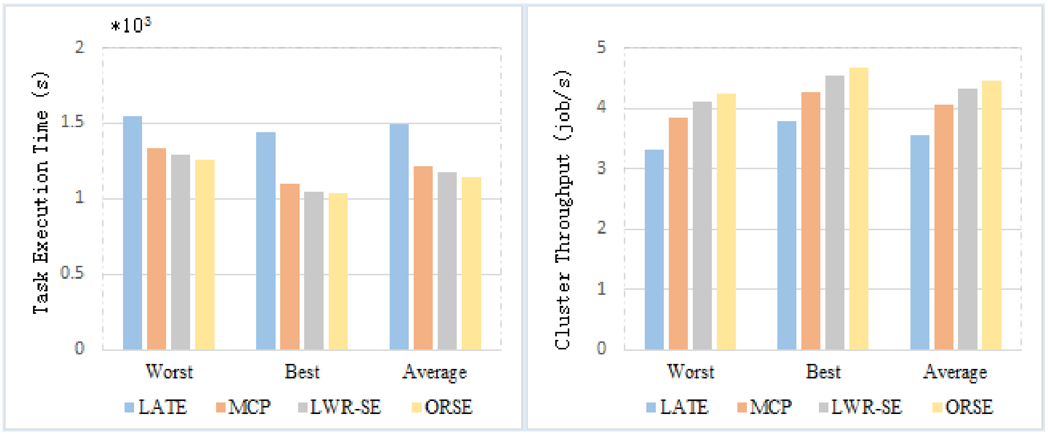

Figure 2: Performance of different SE strategies on *WordCount* jobs in a normal load scenario

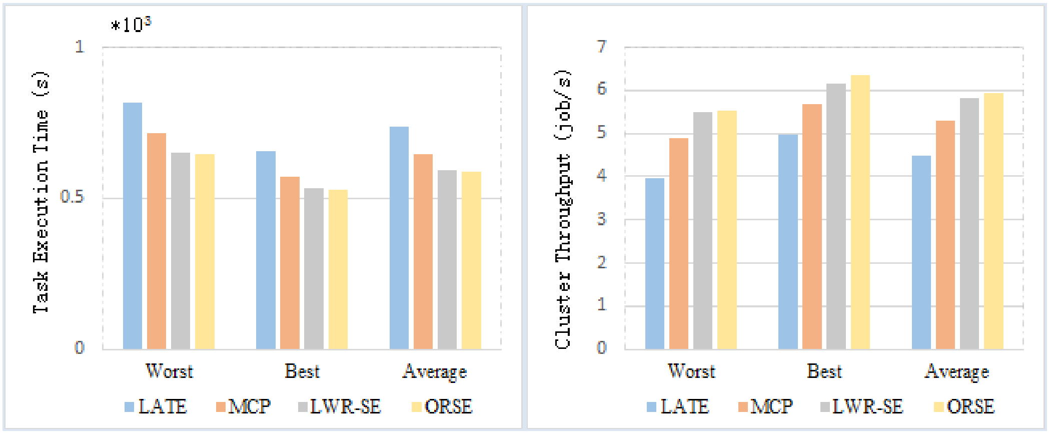

Figure 3: Performance of different SE strategies on *Sort* jobs in a normal load scenario

Busy load scenario

As mentioned earlier, the introduction of game theory is mainly to solve the problem of how to dynamically schedule the cluster resources to maximize the cluster resource utilization when the cluster is under high load. Therefore, in the design of the experiment in this chapter, we select three nodes among the eight nodes in the cluster to perform I/O operations such as file reading tasks to compete for resources in order to simulate the cluster high load scenario. At this time, the accuracy of the monitoring performed and the accuracy of backup node selection become particularly important. The experimental results are shown in Fig. 4 and Fig. 5. As can be seen from Fig. 4, For *WordCount*, on average, ORSE consumes the task execution time 27.3% less than LATE and 13.4% less than MCP and 6.1% less than LWR-SE. Moreover, ORSE improves cluster throughput by 43.2% over LATE and 15.6% over MCP and 9.5% over LWR-SE.

Similarly, we can see from Fig. 5, ORSE gains a corresponding degree of optimization in *Sort* comparing with MCP, LATE and LWR-SE. On average, LWR-SE executes jobs 31.9% faster than LATE and 18.3% faster than MCP and 9.4% faster than LWRSE, whereas ORSE improves cluster throughput by 49.1% over LATE and 21.9% over MCP and 11.1% over LWR-SE.

When the cluster resources are in shortage, “Stragglers” can be misjudged and performed in the slow node by LATE. MCP optimizes the performance through the cluster benefit guarantee strategy, but in Hadoop 2.x and 3.x, the concept of Slot is replaced by Container. All the resources in a cluster are Container resources, where Maps and Reduce types are not divided. Therefore, the accuracy of the MCP cannot be satisfied. The LWR-SE avoids partial misjudgement on the basis of the MCP; however, the efficiency of the selection will be partially insufficient. ORSE dynamically schedules the node computing resources of backup tasks based on LWR-SE; that is, the LWR-SE calculates the execution time and benefit of the backup task, and then the ORSE finds the set of possible nodes for each task to reach the Nash balance, so as to ensure that free computing resources in a cluster are maximized when the cluster appears a set of backup tasks.

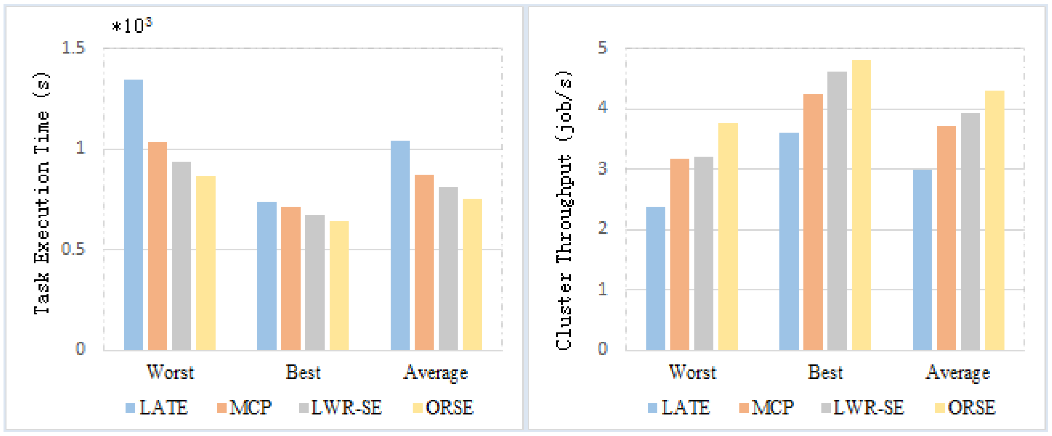

Figure 4: Performance of different SE strategies on *WordCount* jobs in a high load scenario

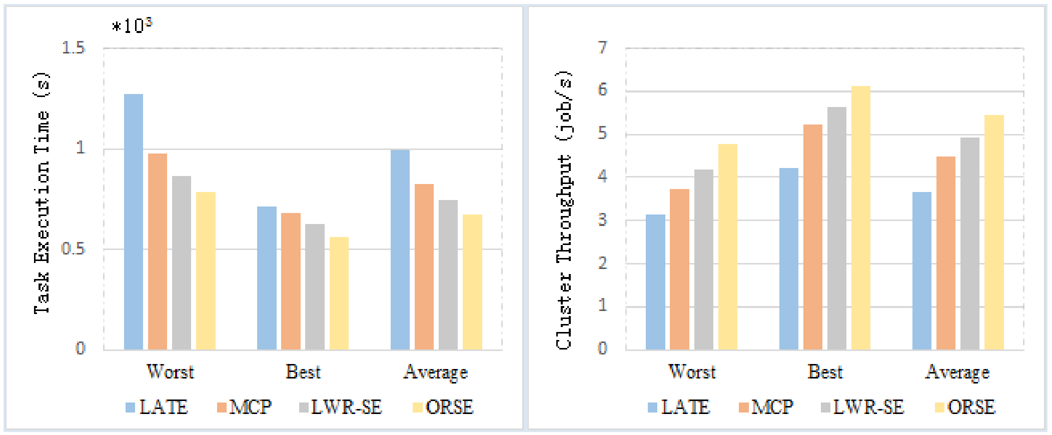

Figure 5: Performance of different SE strategies on *Sort* jobs in a high load scenario

Busy load with data skew scenario

In a practical cloud environment, data skew situation is common, especially in the Map stage, which will result in Stragglers misjudgements due to the different size of input data. In order to create a data skew scenario, the *WordCount* and *Sort* jobs have been proposed with 30GB of their datasets in total and 100MB of input data in each. The input data were divided into two data blocks due to the Hadoop self-split strategy, which are 64MB and 36MB. Similar to the previous Busy Load Scenario, *WordCount* and *Sort* jobs were set up to be submitted every 150 seconds with the task execution time and cluster throughput being counted to be compared with the LATE, MCP and LWR-SE.

According to Fig.6, the performance of the ORSE has been significantly improved, which consumes the job execution time 46.9% less than the LATE, 18.4% less than the MCP and 8.9% less than the LWR-SE. Moreover, the ORSE improves the cluster throughput by 47.2% over the LATE, 23.1% over the MCP and 11.8% over the LWRSE.

Similarly, as can be seen from Fig.7, LWR-SE also gets improvements when executing

Sort jobs in a busy cluster with data skew. In terms of the task execution time, the ORSE finishes jobs 37.8% faster than the LATE, 21.3% faster than the MCP, and 12.1% faster than the LWR-SE. As for the cluster throughput, the ORSE increases by 45.9% over the LATE, 22.9% over the MCP, and 7.3% over the LWR-SE.

When the data skew exists in the cluster, some tasks with slow execution speed appear in the cluster. The reason for their slow speed is not that the execution rate in the real sense is not high, but is due to the influence of the data volume. Therefore, these tasks are not really straggler tasks. The performance of LATE and MCP is significantly reduced when the data is skewed, because it uses the remaining progress and the average rate to calculate the execution time error. The ORSE strategy proposed in this chapter improves the accuracy of LWR- SE. Even if a task with a large amount of data is misjudged as a slow task, the *RM* will find the Nash equilibrium solution according to the possible processing set and conflict set of each task. The task is evenly distributed to the cluster's compute nodes to ensure that the task's execution time and cluster throughput are not affected.

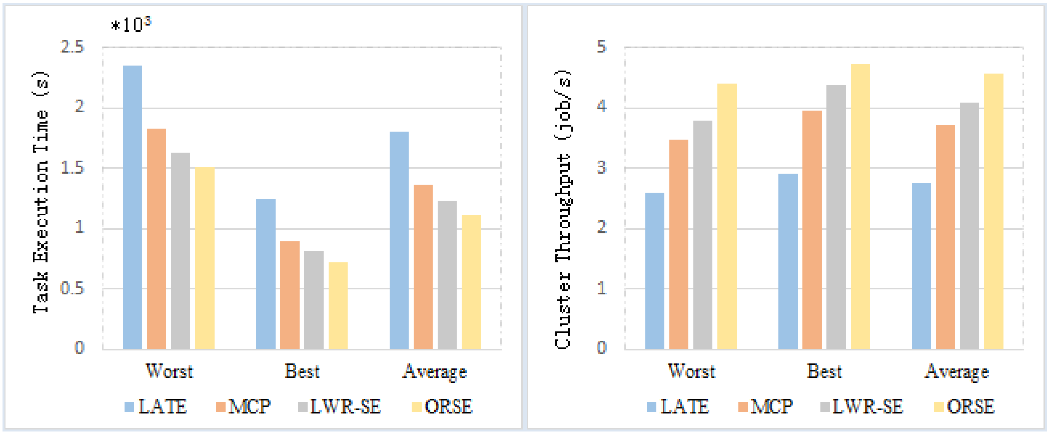

Figure 6: Performance of different SE strategies on *WordCount* jobs in a busy load with data skew scenario

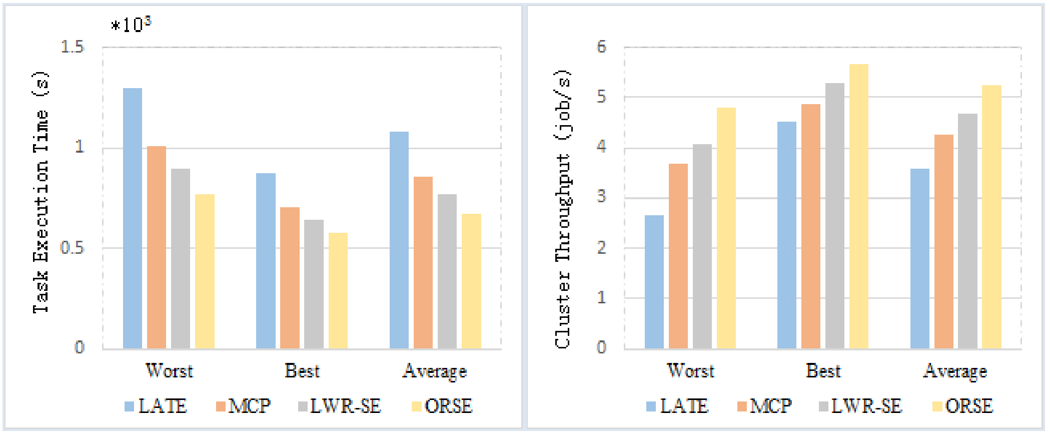

Figure 7: Performance of different SE strategies on *Sort* jobs in a busy load with data skew scenario

5. Conclusion

In order to solve the problem that the efficiency of the backup node allocation strategy of classical speculative execution algorithm is not high, this paper proposes a speculatively executed hybrid resource scheduling strategy ORSE based on non-cooperative game, which transforms the resource scheduling problem of backup task into a classic non-cooperative game problem. The participants of the game are the set of backup tasks. The game strategy is the computing node in the cluster. The utility function of the game is the final completion time of the task. The experimental results show that ORSE has better performance than LATE, MCP, and LWR-SE strategies, and can make greater use of the cluster's computing resources, improve cluster throughput, and estimate the efficiency of execution.

Acknowledgement: This work has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement no.701697, Major Program of the National Social Science Fund of China (Grant No.17ZDA092), Basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20180794) and the PAPD fund.

References

Apache hive. (2018): https://hive.apache.org/, last accessed 2018/10/16.

Bhupathiraju, V.; Ravuri, R. P. (2015): The dawn of big data - hbase, in: It in Business. *Industry and Government*, pp. 1-4.

Chang, F.; Dean, J.; Ghemawat, S.; Hsieh, W.; Wallach, D. A.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, T. (2008): Bigtable:a distributed storage system for structured data. *Acm Transactions on Computer Systems*, vol. 26, no. 2, pp. 1-26.

Chen, Q.; Liu, C.; Xiao, Z. (2014): Improving mapreduce performance using smart speculative execution strategy. *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 954-967.

Cheng, D.; Rao, J.; Guo, Y.; Zhou, X. (2014): Improving mapreduce performance in heterogeneous environments with adaptive task tuning. *Proceedings of the 15th International Middleware Conference*, ACM, pp. 97-108.

Czumaj, A.; Cking, B. V. (2002): Tight bounds for worst-case equilibria. *Thirteenth Acm-Siam Symposium on Discrete Algorithms*, pp. 413-420.

Dean, J.; Ghemawat, S. (2008): Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, vol. 51, no. 1, pp. 107-113.

Ghemawat, S.; Gobioff, H.; Leung, S. T. (2003): The google file system. *Nineteenth ACM Symposium on Operating Systems Principles*, vol. 37, pp. 29-43.

Glushkova, D.; Jovanovic, P.; Abello, A. (2017): Mapreduce performance model for hadoop 2. X. *Information Systems*, pp. 1-37.

Guo, Y.; Rao, J.; Jiang, C.; Zhou, X. (2017): Moving hadoop into the cloud with flexible slot management and speculative execution. *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 3, pp. 798-812.

Hashem, I. A. T.; Yaqoob, I.; Anuar, N. B.; Mokhtar, S.; Gani, A.; Khan, S. U. (2015): The rise of big data on cloud computing: Review and open research issues. *Information Systems*, vol. 47, pp. 98-115.

Isard, M.; Budiu, M.; Yu, Y.; Birrell, A.; Fetterly, D. (2007): Dryad:distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 59-72.

Kong, Y.; Zhang, M.; Ye, D. (2015): An auction-based approach for group task allocation in an open network environment. *The Computer Journal*, vol. 59, no. 3, pp. 403-422.

Kong, Y.; Zhang, M.; Ye, D.; Zhu, J.; Choi, J. (2017): An intelligent agentbased method for task allocation in competitive cloud environments. *Concurrency & Computation Practice & Experience*, vol. 30, no. 6, pp. e4178.

Lee, J. (2013): A view of cloud computing. *Communications of the Acm*, vol. 53, no. 4, pp. 50-58.

Li, Y.; Yang, Q.; Lai, S.; Li, B. (2015): A new speculative execution algorithm based on c4.5 decision tree for Hadoop. *International Conference of Young Computer Scientists, Engineers and Educators*, pp. 284-291.

Li, Z. (2016): A non-cooperative differential game model for resource allocation in cloud computing. *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 9, pp. 5993-5998.

Liu, Q.; Cai, W.; Fu, Z.; Shen, J.; Linge, N. (2016): A smart strategy for speculative execution based on hardware resource in a heterogeneous distributed environment. *International Journal of Grid and Distributed Computing*, vol. 9, no. 2, pp. 203-214.

Liu, Q.; Jin, D.; Liu, X.; Linge, N. (2016): A survey of speculative execution strategy in mapreduce. *International Conference on Cloud Computing and Security*, pp. 296-307.

Naik, N. S.; Negi, A.; Sastry, V. N. (2014): A review of adaptive approaches to mapreduce scheduling in heterogeneous environments. *International Conference on Advances in Computing, Communications and Informatics*, pp. 677-683.

Manikandan, S. G.; Ravi, S. (2014): Big data analysis using apache Hadoop. *IT Convergence and Security (ICITCS), 2014 International Conference on, IEEE*, pp. 1–4.

Stergiou, C.; Psannis, K. E. (2017): Recent advances delivered by mobile cloud computing and internet of things for big data applications: a survey. *International Journal of Network Management*, vol. 27, no. 3, pp. e1930.

Storey, V. C.; Song, I.-Y. (2017): Big data technologies and management: What conceptual modeling can do. *Data & Knowledge Engineering*, vol. 108, pp. 50–67.

Toshniwal, A.; Taneja, S.; Shukla, A.; Ramasamy, K.; Patel, J. M.; Kulkarni, S.; Jackson, J.; Gade K.; Fu, M.; Donham, J.; Bhajnt, N.; Mittal, S.; Ryaboy, D. (2014): Storm@twitter. *ACM SIGMOD International Conference on Management of Data*, pp. 147-156.

Wang, Y.; Lu, W.; Lou, R.; Wei, B. (2015): Improving mapreduce performance with partial speculative execution. *Journal of Grid Computing*, vol. 13, no. 4, pp. 587-604.

Wu, H.; Li, K.; Tang, Z.; Zhang, L. (2014): A heuristic speculative execution strategy in heterogeneous distributed environments, in: Sixth International Symposium on Parallel Architectures. *Algorithms and Programming*, pp. 268-273.

Yang, S. J.; Chen, Y. R. (2015): Design adaptive task allocation scheduler to improve mapreduce performance in heterogeneous clouds. *Journal of Network & Computer Applications*, vol. 57, pp. 265-270.

Zafar, F.; Khan, A.; Malik, S. U. R.; Ahmed, M.; Anjum, A et al. (2017): A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers & Security*, vol. 65, pp. 29-49.

Zaharia, M.; Chowdhury, M.; Franklin, M. J.; Shenker, S.; Stoica, I. (2010): Spark: cluster computing with working sets. *Usenix Conference on Hot Topics in Cloud Computing*, pp. 10-10.

Zhang, L.; Zhang, L.; Li, R.; Wan, L.; Li, K. (2016): Novel heuristic speculative execution strategies in heterogeneous distributed environments. *Computers & Electrical Engineering*, vol. 50, no. C, pp. 166-179.

Zhang, L.; Zhou, J. H. (2017): Task scheduling and resource allocation algorithm in cloud computing system based on non-cooperative game. *IEEE International Conference on Cloud Computing and Big Data Analysis*, pp. 254-259.