Journal of Network and Computer Applications 57 (2015) 61–70

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds

Shin-Jer Yang *, Yi-Ru Chen

Department of Computer Science & Information Management, Soochow University, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 8 August 2014
Received in revised form
3 February 2015
Accepted 15 July 2015
Available online 1 August 2015

Keywords:
Cloud computing
ATAS
LATE
Hadoop Speculative
MapReduce

ABSTRACT

This paper conducted a thorough research on one of the critical technologies in cloud computing, the MapReduce programming model. Some of the past research results showed that one of the methods for enhancing MapReduce performance is the execution through allocating identical tasks to each cloud node. However, such allocation is not applicable for the environment of a heterogeneous cloud. Due to the different computing power and system resources between the nodes, such uniform distribution of tasks will lower the performance between nodes, and hence this paper makes improvements on the original speculative execution method of Hadoop (called Hadoop Speculative) and LATE Scheduler by proposing a new scheduling scheme known as Adaptive Task Allocation Scheduler (ATAS). The ATAS adopts more accurate methods to determine the response time and backup tasks that affect the system, which is expected to enhance the success ratio of backup tasks and thereby effectively increase the system's ability to respond. We conducted three different simulation experiments, in which execution time of the LATE Scheduler and ATAS could be reduced by a maximum of 12% and 30%, increased the average tasks throughput by a maximum of 13% and 33%, and reduced the average tasks latency by a maximum of 25% and 36% compared to Hadoop Speculative; the ATAS can effectively enhance the processing performance of MapReduce to be compared with LATE Scheduler in a heterogeneous cloud environment.

1. Introduction

Cloud computing is the next-generation Internet while MapReduce is the key technology to cloud computing (Armbrust et al., 2010; Pallis, 2010), which was first released by Google at the international conference of Operating Systems Design and Implementation (OSDI) in 2004 (Dean and Ghemawat, 2008). MapReduce is a distributed programming framework that allows service developers to write simple programming using a lot of computing resources to speed up the processing of massive amount of data, whereas the Map program first divides data into a variety of small blocks, followed by allocation to a large number of servers for processing. The Reduce program will compile and export the processed results to customers. MapReduce is similar to a massive application program of "Sort & Merge". Today, the MapReduce framework is widely applied in a number of science fields (Amin et al., 2011; Almeer, 2012; R. Chen et al., 2010), whereas the technicians can quickly select data through Map using the MapReduce framework and divide the data on various servers. Each section after the division can synchronously execute computing and collecting this intermediate data. Finally, the data is handed to Reduce to simplify the procedures by combining all intermediate data as input, followed by exporting the final simplified results (Gu and Grossman, 2011; Zaharia et al., 2008). Recently, some premium researchers in recent years have discovered the special application of the MapReduce framework in the processing of single-function large data. For this reason, MapReduce coincidentally specializes in the application that requires large distributed data processing in cloud computing.

The cloud application development environment allows users to quickly develop and process a large amount of data application through the MapReduce program design; nonetheless, the majority of methods assume nodes under a homogeneous environment and pre-adopting uniform allocation for task allocation. For each node allocated with the same number of tasks, such allocation does not apply to the heterogeneous cloud computing environment

* Correspondence to: Department of Computer Science and Information Management, Soochow University, No. 56, Sec. 1, Kuei-Yang St., Taipei 100, Taiwan. Tel.: +886 2 23111531x3802; fax: +886 2 23756878.
E-mail addresses: sjyang@csim.scu.edu.tw (S.-J. Yang), neilchen131@gmail.com (Y.-R. Chen).

http://dx.doi.org/10.1016/j.jnca.2015.07.012
1084-8045/

(Hu et al., 2010; Tian et al., 2009). In view of the various processing power and remaining resources for each work node, this will result in extra redundant time and execution of unnecessary backup tasks, which leads to the main purposes of this paper as described below:

(1) To take consideration of the heterogeneity of cloud platforms to make proper resource allocations with the node processing power and system resource through analysis (Rao et al., 2011).
(2) To propose Adaptive Task Allocation Scheduler (ATAS) for improving the lack of existing scheduling methods for original Hadoop Speculative and LATE Scheduler in order to solve the lack of original Hadoop scheduling.
(3) To identify the heterogeneity of cloud platform to allocate backup tasks to nodes with excellent performance for execution.
(4) The ATAS will make more reasonable task allocations based on the processing power of each calculated node in order to improve the performance of MapReduce through simulation experiments on the heterogeneous cloud.

The rest of this paper is organized as follows. In Section 2, we will elaborate on relevant literature review to survey the application for existing MapReduce in heterogeneous cloud and issues with the existing scheduling method. In Section 3, we propose the operational procedure of ATAS and design its algorithm. In Section 4, we describe the experimental environments and simulation procedures and list some key benchmarks. Also, we compare our proposed ATAS scheme with Hadoop and LATE and make the results analysis. A conclusion will be drawn in Section 5.

2. Related work

2.1. Hadoop overview

Hadoop was a project developed by Dong Cutting under Lucene within Apache.org. In the history of Lucene, the popularity of Google as a search engine application was referred to in Dong Cutting's published paper on MapReduce and Google File System (GFS) architecture announced by Google (Dean and Ghemawat, 2008). In this project, practice on MapReduce and GFS is conducted to further separate Hadoop. The application developed from this source code has extended capability of rapid computing of large machines to joint computing of multiple machines, and use concept of distributed architecture and gradually mature PC computing to achieve function similar to large mainframe (Kambatla et al., 2009).

Areas that apply MapReduce are as shown below: scientific research projects (Zhou et al., 2010) (e.g., research of climatology, biology, astronomy etc.); search/internet companies (Google (Dean and Ghemawat, 2008), Microsoft, Wikipedia, Yahoo, IBM (Kruijf and Sankaralingam, 2009), etc.); IT companies (Hewlett Packard, Intel, etc.); universities (Carnegie Mellon University, TU Dresden, University of Pennsylvania, University of Central Florida, etc.). Many papers have discussed system function boosts of computing due to different requirements (Rao and Reddy, 2011; Xu et al., 2011), but the problems cannot be completely solved though the new architecture that is proposed.

2.2. Operating principles of MapReduce

MapReduce is a simple programming model, whereas the system helps with sorting and classification of intermediate data when using Map and Reduce functions in programming design, followed by using the Reduce program to simplify the intermediate data (Jiang et al., 2009; Seo et al., 2009). The purpose of MapReduce aims to execute distributed computing on large datasets in a cluster computer, processing parallel computing for a large amount of data. The overall framework of MapReduce is constituted by Map and Reduce functions, whereas when functions are entered into a big group of key/values, the Map function will automatically split original key/value into many pairs of intermediate key/value. Then, the Reduce function merges these intermediate value pairs with identical keys, which is simplified and generated with the final output results (Kim et al., 2011). The execution process of MapReduce is shown in Fig. 1.

2.3. Speculative executing task

The MapReduce model brings us an essential benefit by automatically processing faults. When a node is at fault, it will randomly choose another node to execute tasks of the faulty node. It is equally important when a node is not paralyzed by the task speed executed on the node, it is significantly slower than other nodes, and such node is known as the straggler (Q. Chen et al., 2010). MapReduce also picks another working node to execute the replication of the tasks of the straggler, while such mechanism is

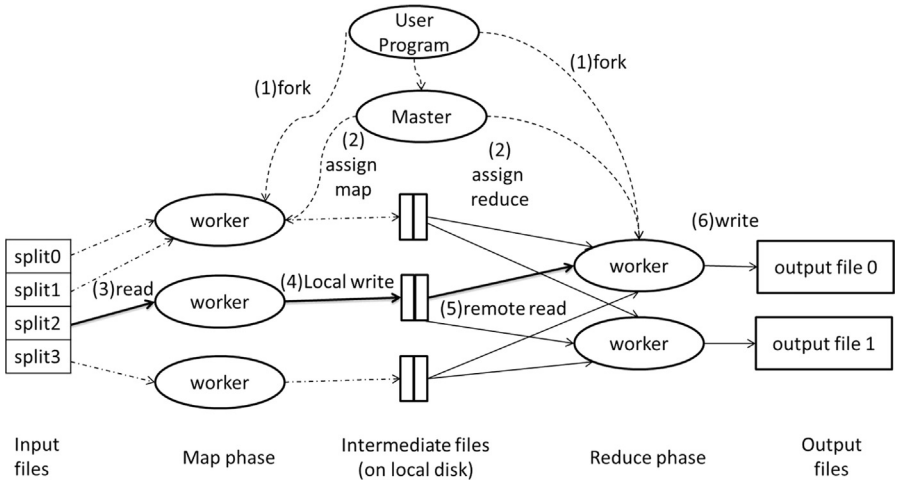

Fig. 1. Execution process of MapReduce.

referred to as the speculative execution (Q. Chen et al., 2010). These speculative execution tasks are known as the backup tasks. The straggler could have resulted from various reasons, such as a hardware malfunction or configuration errors. Google suggested that passing pre-executing tasks will enhance the response time by 44% on the existing system.

The enhancement of performance in speculative execution tasks is valid under homogenous clouds except for the virtual data center of Amazon, which is in a heterogeneous cloud environment (Rao et al., 2011). Virtual resources should be computed in a virtual computing environment (Hu et al., 2010; Visalakshi and Karthik, 2011). However, the performance of virtual machines in terms of processing performance cannot be accurately controlled. Even in a private physical data center, heterogeneity is widely prevalent. Companies or enterprises often mix the use of new and old hardware and equipment, while virtualization technology will bring the data center with benefits of convenient control and integrated servers. Studies show that a heterogeneous environment will lead to inaccurate and excessively speculative execution of tasks, whereas excessive repetition of executing straggler tasks often perform worse under situations without speculative execution. In some experiments, approximately 80% of the tasks are regarded from the straggler and executed once from other nodes.

In practical application, it is quite difficult to confirm whether or not a node or task belongs to a straggler and the main reasons are described below:

(1) Executing backup tasks requires consumption of system resources while competing for system resources such as bandwidth with other operating tasks at the same time.
(2) How to choose the working node for backup task execution is equally important as how to choose backup services. If the backup tasks are allocated to execution on nodes with poor performance, it will not only increase the response time but also add workload to that node and waste system resources.
(3) It is quite difficult to determine the straggler node in a heterogeneous cloud.
(4) Stragglers should be timely detected and their backup tasks executed in order to possibly reduce the response time of the system.

2.4. Speculative execution task in Hadoop

The choice of backup tasks for Hadoop is determined according to progress value (Q. Chen et al., 2010). The estimation for the task progress under the Map phase is shown in Eq. (1).

Map phase:

$$\text{Progress} = M/N \tag{1}$$

where M denotes the number of key/value pairs have been processed in the task. N denotes the number of key/value pairs have been processed in the task. And Progress denotes the progress value. The calculation of the progress value at the Map phase is referred to as the ratio between tasks that have completed processing and the total number of tasks (Kruijf and Sankaralingam, 2009).

At the Reduce phase, this value is divided into three stages, namely the replication stage, sorting stage and Reduce stage, where each account for one third of the total progress value. The calculation is shown in Eq. (2).

Reduce Phase:

$$\text{Progress} \approx 1/3 \times (K + M/N) \tag{2}$$

where the number of tasks having completed processing is denoted by M, total number of tasks that require processing is denoted by N, the serial number of the stage is denoted by K (Reduce task), and the task progress is referred to as Progress.

In addition, the determination of the straggler is shown in Eqs. (3) and (4).

If progress[i] of the task is satisfied

$$\text{avgProgress} = \sum_{i=1}^{P} \text{Progress}[i]/P \tag{3}$$

Launch backup task.

$$\text{Progress}[i] < \text{avgProgress} - 20\% \tag{4}$$

where the avgProgress is the average task progress value. Progress [i] denotes the progress of the ith task and P is the number of tasks. If the progress value, Progress, is smaller than the avgProgress deducted by 20%, then the task is determined to belong to the straggler and the backup task is launched.

In a homogeneous environment (Kruijf and Sankaralingam, 2009), the consideration for Hadoop is reasonable because the start time and the end time of tasks are roughly consistent. Hence, the calculation of Eq. (4) is reasonable. But in a heterogeneous environment, the execution time ratio at different phases of tasks will differ with the nodes, whereas differences in the task execution ratio for different nodes, could not accurately determine which tasks belong to the straggler. In some cases, some tasks will appear with smaller progress but faster execution speed, or appear with greater task progress but slower completed tasks compared with other tasks.

2.5. LATE Scheduler

The previous LATE Scheduler chose the tasks with the longest remaining time in order to launch the backup tasks (Zaharia et al., 2008) and its calculation is shown in Eqs. (5) and (6). Also, the LATE Scheduler will choose the tasks with the longest remaining time to launch the backup tasks.

$$\text{ProgressRate} = \text{ProgressScore}/T \tag{5}$$

$$\text{TimeToEnd} = (1 - \text{ProgressScore})/\text{ProgressRate} \tag{6}$$

where ProgressScore denotes progress of tasks (range from 0 to 1). The time to be operated for the task is T, the processing speed of the task is ProgressRate, and the longest remaining time is TimeToEnd.

Nonetheless, the LATE method contains the following flaws:

(1) For an inaccurate estimation of task remaining time, the LATE Scheduler still applies the existing scheduler of Hadoop to divide the Reduce task into three stages, namely the Replication stage, Sorting stage and Reduce stage. Each stage may have one-third of the task progress, namely is 33.3%, 33.3% and 33.3%. However, the three stages of the Reduce task should have an actual operating progress as 60%, 20% and 20%. When the Reduce task finishes the Replication stage, the remaining time of the Reduce task can be computed by Eq. (6), which is $(1–33.3\%) \times (T/33.3\%) = 2T$, but the actual remaining time of the Reduce task is only $(1–60\%) \times (T/60\%) = 0.67T$.
(2) Slow Nodes and quick nodes are not distinct when Map and Reduce tasks are in being executed. First, the LATE Scheduler is divided into slow and quick nodes on the MapReduce system. Such distinction is necessary, but there exist some flaws. To consider more deeply the heterogeneous systems, the different resources such as CPU, memory and disk controlled by each node will appear on the same Map task; however, their operating time may be different on various nodes. Since the LATE Scheduler is not identified by any slow node between the

Map task and Reduce task, it will lower the system resource utilization.

3. Operations and design issues of ATAS

The speculative execution algorithm proposed by Hadoop (or called Hadoop Speculative) aims to provide additional execution opportunities for tasks with slower executing speeds, so that the backup tasks of stragglers can finish with execution faster than the original tasks, thereby reducing the task response time. Taking into account the existing Hadoop Speculative and LATE Scheduler strategies, the progress value of tasks taken into consideration could not accurately reflect the task progress in the system so that the system can execute unnecessary backup tasks. Even if the speculative execution tasks belong to the straggler, it can simply allocate the task to any working node that requests tasks without considering the difference in heterogeneity on the working nodes. The execution of multiple replications for straggler tasks could allow backup tasks to maintain a higher probability of replacing the execution of original task. Nonetheless, such results lead to a greater demand for system resources. For the inadequacy of Hadoop Speculative and LATE Scheduler, the proposed ATAS focuses on improvements in the determining ways of stragglers and slow nodes.

3.1. Straggler determination strategy

To make the speculative execution more meaningful, the ATAS proposes a new evaluation model, the Approximate End Time (abbreviated as AET). The speculative execution is based on the principle of launching backup tasks for future Tasks with the longest expected execution time during in the computing process. Speculative execution covers the original task with a backup task and it is faster than the original task execution, which reduces the task response time. It is not accurate for Hadoop to utilize Progress in the determination of completed time of Task execution. Therefore, the AvgProgress of this paper represents the average speed of task execution and it is computed as AvgProgress = Progress/T. Also, we assume all the tasks are executed in accordance with the same speed, and k is the serial number of the Reduce stage. Hence, the Approximate End Time (AET) is computed by Eq. (7).

$$\mathrm{AET} = \frac{1 - \sum_{i=1}^{k} \mathrm{Progress}[i]/T)}{\mathrm{AvgProgress}} \tag{7}$$

where Progress[i] denotes progress value of current task. AET is calculated by 1 subtracting the Progress[i] then divided by a AvgProgress to produce AET.

In Eq. (7). the AET estimates how long the Task will complete execution based on the existing Task executing speed. Even if the existing system comes with different batches of Tasks, the newly scheduled Task executing time, T, will be smaller; therefore, its AET is not too large for executing a backup task of the newly scheduled Task. Also, The algorithm sets up an upper limitation value as BackupTaskCap for the backup task concurrently executing in the system. Such upper limit value should remain smaller so that it will not affect the normal system scheduling of operations. The previous experimental results show that the upper limited value of backup task is configured to one tenths of the number of tasks currently under execution (Zaharia et al., 2008). Hence, the speculative execution of the system can yield better results, as shown in Eq. (8).

$$\mathrm{BackupTaskCap} = \mathrm{TaskNum} \times 10\% \tag{8}$$

where TaskNum is the number of executing tasks for the system.

3.2. Slow-node determination strategy

The aforementioned equations introduced the inaccuracy of Hadoop for the allocation of backup tasks, resulting in speculative execution not able to improve the processing power according to the design concepts. The ATAS algorithm of the study adopted a simple improvement strategy with excellent improvement effects; namely, we only allocated backup task to nodes with good performance and fast calculating speed for execution.

To implement this rule, ATAS specified a threshold value, SlowNodeThreshold, to determine whether or not a node is a quick node or a slow node, where a node with a value lower than this threshold value is regarded as a SlowNode or else a QuickNode. When a QuickNode request for tasks and the system has a backup task that requires executing, the backup task will be prioritized for allocation with executing to a QuickNode, whereas a SlowNode will not be allocated with a backup task when requesting tasks. The previous authors conducted many experiments to propose that a well performing value of SlowNodeThreshold is 25% (Zaharia et al., 2008). According to that value, the SlowNodeThreshold is refined to 25% of the average progress of nodes in our ATAS algorithm.

The level of node progress is not only determined by the task progress operating on existing nodes, but the sum of tasks successfully execute from the beginning of the system initialization and the task progress values currently executing. The task progress value of successful execution is denoted as 1, or else denoted with the progress value, so the processing power and computing speed of the nodes can be better determined, and thereby more accurately adjust the computing speed of the nodes. Hence, the SlowNodeThreshold is shown in Eq. (9).

$$\mathrm{SlowNodeThreshold} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{ProgressScore}[i](\mathrm{Task}[j])}{n} \tag{9}$$

where Task[j] denotes task which is executing, n denotes the numbers of nodes in the system, m denotes the number of tasks executed by each node (total number of completed and being executed).

3.3. Operating procedure and algorithm design of ATAS

The section takes into consideration the cluster heterogeneity and carries out improvements on the MapReduce speculative execution method. We propose a brand-new operation scheduling algorithm, known as Adaptive Task Allocation Scheduling (ATAS). The ATAS can automatically schedule the load balancing of a job queue size in the dynamic scheduling system according to the loadings of the existing system, using a more accurate method to speculate straggler task and allocating it to a calculating node with better performance in order to enhance the system response time, in addition to controlling the quantity of backup tasks, thereby preventing task jitter. The operation process of ATAS is shown in Fig. 2 and each step is expressed as follows.

(1) All TaskTracker are executed from task queues accordingly.
(2) Each node is calculated with the speed and remaining time for the task under execution, in addition to determining which nodes are SlowNodes.
(3) When a QuickNode requests a task and the system has a backup task that requires execution, the backup task is prioritized for allocation to the QuickNode for execution, whereas the SlowNode is not allocated with the backup task when requesting the task.

Based on Fig. 2, the pseudo code for ATAS algorithm can be designed as follows:

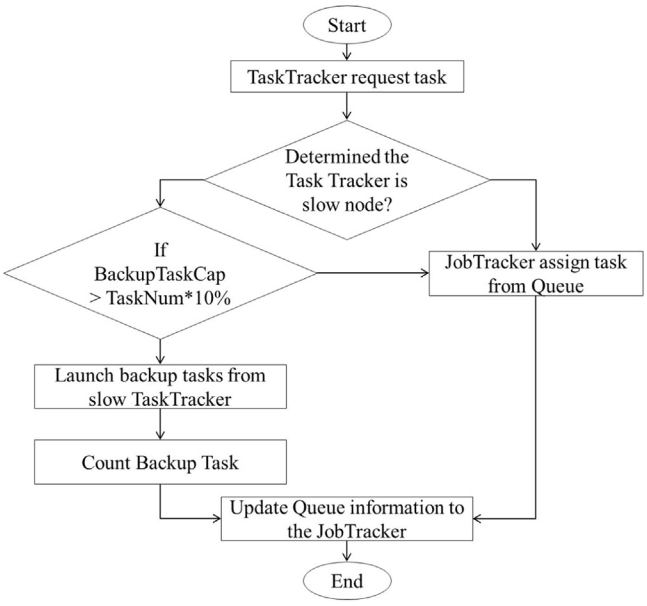

Fig. 2. The operational flow of ATAS.

Table 1
Detail environment of experiments.

Cloud groups	No. of VM for each PC	No. of PC	No. of nodes	Write rate (MB/S)
Cloud 1	1	10	10	33.2
Cloud 2	2	15	30	49.3
Cloud 3	4	15	60	63.1

Algorithm ATAS()
{
Input:
#define TaskTracker[1…i] i=1…n;
#define AssignNewTask(): Assign new task from queue;
#define AssignBackupTask(): Launch backup task from slow TaskTracker;
#define AET_Weight: The task approximate end time;
#define AvgAET_Weight: The average of tasks approximate end time;
#define SlowNodeThreshold: The threshold of slow node;
#define ProgressWeight$_i$: The progress of i-th task;
#define T_i: The execution time of i-th task;
#define BackTaskNum: The number of backup tasks;
#define TaskNum: The number of executing tasks;
Output:
To design ATAS for improving the MapReduce performance;
Method:
BEGIN{
For each running TaskTracker do
TaskTracker[i]=
FindSlowTaskTracker(TaskTracker[i]);
For each idle TaskTracker do
If (TaskTracker[i].equal("SlowNode"))
Assign BackupTask(TaskTracker[i]);
Else
AssignNewTask(TaskTracker[i]);
End For

Procedure **FindSlowTaskTracker**(TaskTracker[i]){
For each running TaskTracker do
AvgAET_Weight=($\sum_{i=1}^{m}$ AET_Weight$_i$)/m;
SlowNodeThreshold=AvgAET_Weight*0.25;
AvgProgress$_i$=ProgressWeight$_i$ /T_i;
AET_Weight$_i$=(1−ProgressWeight$_i$)/AvgProgress$_i$;
If (AET_Weight$_i$–AvgAET_Weight > SlowNodeThreshold)
Return SlowNode;
End For
}End

Procedure **AssignNewTask**(TaskTracker[i]){
JobTracker assign new task;
TaskNum++;
Collecting results and updating JobTracker information;
}End

Procedure **AssignBackupTask**(TaskTracker[i]){
If (BackupTaskNum >= askNum*10%)
AssignNewTask(TaskTracker[i]);
Else{
Launching backup tasks from SlowNode;
BackupTaskNum++;
Collecting results and updating JobTracker information; }
}End

}
END ATAS.

4. Simulations setup and results analysis

4.1. Simulations setup

In a heterogeneous computing environment, a set of different machines is interconnected to provide a variety of computing capabilities to meet the processing demands of large, diverse groups of tasks. An important research problem for heterogeneous computing is how to assign computing and communication resources to tasks and to tune the schedule order of their executions to meet some performance criteria.

Our experiments in this paper, we install the heterogeneous cloud environment through physical machines and virtual machines. All virtual machines (or called nodes) in the simulation environment are to be installed on the Ubuntu system, Also, we use VMware 6.5 to manage all nodes, JDK version 1.6.0 and Hadoop version 0.20.2. Our proposed ATAS can perform simulations under this environment as shown in Table 1.

The experiments were expected to install 100 nodes with different hardware configurations. Three types of cloud groups will be divided in total, with each group to introduce a computer system with a different level of hardware configuration, to conduct simulation experiments in a heterogeneous cloud environment constructed on three groups of computers, in which the system architecture of the simulation experiment and hardware specifications of three types of groups are shown in Fig. 3 and Table 2.

The benchmarks used in these experiments are examples in Hadoop: MinuteSort, WordCount and Grep, because Hadoop also uses these programs as benchmarks. Three benchmarks clearly show key characteristics of MapReduce. The characteristics of the benchmarks are shown in Table 3.

Capacity of input data will be set to 500 GB, and there are two replications of the system setting, with one TaskTracker operating on each node. Two Map tasks and two Reduce tasks are executed simultaneously on each TaskTracker. The comparisons on simulation experiments upon three schedules of ATAS, LATE and Hadoop Speculative are conducted so as to confirm whether ATAS is more applicable to MapReduce or Hadoop in a

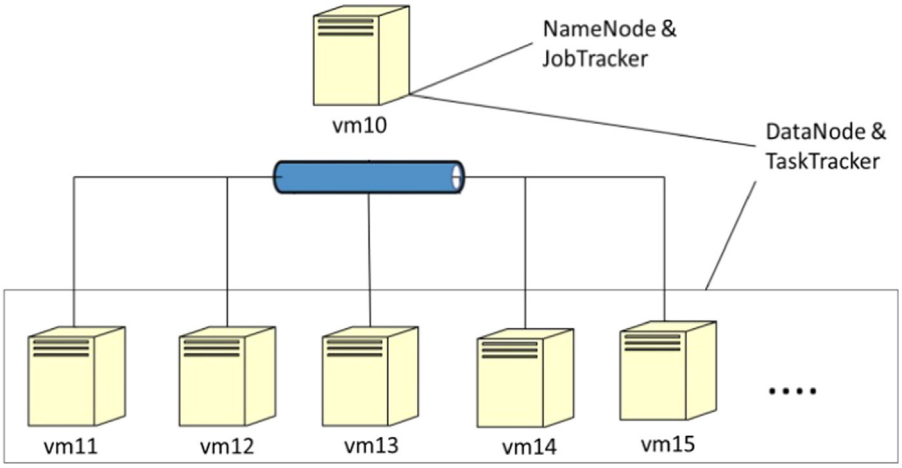

Fig. 3. The system architecture of simulation environment.

Table 2
Hardware specifications.

Hardware configurations	Cloud groups		
	Cloud 1	Cloud 2	Cloud 3
Operation system	Ubuntu 10.10	Ubuntu 10.10	Ubuntu 10.10
CPU	2 CPU	4 CPU	4 CPU
Memory	4 GB	8 GB	16 GB
Disk	500 GB	1 TB	2 TB

heterogeneous cloud environment compared to the other two types of scheduling.

In this paper, we will use 3 KPIs to observe and evaluate throughput performance and simulation outcomes of ATAS. The 3 KPIs are execution time, average task throughput and average task latency. The purpose of analyzing these 3 KPIs are shown in Table 4.

First of all, execute MinuteSort, WordCount and GrepCount programs by 3 schedules of ATAS, LATE and Hadoop Speculative, collect the average of 10 executions and analyze accordingly, then the worst, average, and best execution times will be derived. This is because of the execution time of each simulation experiment there are slightly different, so we can perform ten times averaged to obtain more objective results.

Besides, these 3 schedules is set under different nodes with distribution of nodes as shown in Table 5. We will execute MinuteSort, WordCount and GrepCount programs for 10 times and analyze the worst, average and best for averaging task throughput and latency. It analyzes average task latency of tasks throughput under different nodes in order to verify that the ATAS method can reduce average task latency.

4.2. Results analysis

First, Fig. 4 shows the execution time for all scheduling methods executed with MinuteSort. Under the heterogeneous environment of the testing cluster, relative to the original scheduling of Hadoop Speculative. In the Average Case, the execution time of ATAS and LATE Scheduler can be reduced by 30% and 12%, respectively, compared to the execution time of Hadoop Speculative. This is because the tasks in the Reduce phase spend a longer time in MinuteSort. Launching backup tasks on the QuickNode can finish in a shorter time than the SlowNode, and hence save a lot of time. The JobTracker first finds the Straggler and then launches backup tasks on the QuickNode, and ATAS can achieve better performance than another MapReduce schedulers.

Then, Fig. 5 shows that relative to MinuteSort, WordCount has only minor improvements. In the Average Case, the execution time of ATAS and LATE Scheduler can be reduced by 18% and 9%, respectively, compared to the Hadoop Speculative. This is because tasks in the Reduce phase spend a shorter time in WordCount than the MinuteSort benchmark. Hence, launching backup tasks on the QuickNode for Stragglers cannot save much time.

Figure 6 shows that under the heterogeneous environment there are better improvements on the execution time relative to WordCount. In the Average Case, the execution time of ATAS and LATE Scheduler can be reduced by 21% and 12%, respectively, compared to the Hadoop Speculative. This is because the tasks in the Reduce phase spend more time in the GrepCount than the WordCount benchmark. Hence, launching backup tasks on QuickNode for Stragglers can save much more time.

To summarize Figs. 4–6 as depicted in Table 6, the ATAS Scheduler shows more applicability to the heterogeneous cloud computing environment and obtains better improvements than Hadoop Speculative and LATE Scheduler. After using the ATAS, the performances on execution time are enhanced for three different Benchmarks. Under the heterogeneous cloud environment, ATAS divides the nodes into QuickNode and SlowNode, which accurately evaluates the completion time for tasks and can effectively improve the MapReduce performance.

Figure 7 indicates the average task throughput of the MinuteSort program running by different nodes 10 times, in which the improvement performance of ATAS will become eminent along with an increase in node number. When the node number is 100, LATE will enhance average task throughput by around 11% compared to Hadoop Speculative, while ATAS will enhance the average task throughput by around 33% compared to Hadoop Speculative. This is because of an increase in node number and system resource. Reasonable execution of task backup on the fast node has further enhanced computing performance and caused an increase in average task throughput, showing that ATAS could improve the throughput performance of MapReduce in the heterogeneous cloud environment.

The average task throughput of the WordCount program run 10 times by different nodes is shown in Fig. 8. When the node number is 100, LATE will enhance average task throughput by around 8% compared to Hadoop Speculative inferential execution, while ATAS will enhance average task throughput by around 17% compared to Hadoop Speculative. This is because of the short Reduce task execution

Table 3
The details of three benchmarks.

Benchmarks	Detailed description
MinuteSort	Use a large amount of data in 60 s or shorter to compare the size of data amount sorted in one minute.
WordCount	Dividing sentences and calculating the number of times the words appear. Map requires large computing and it tests the stability and reliability of MapReduce in processing large-scale data.
GrepCount	Retrieve the strings with designated words from the document in addition to conducting counting statistics.

Table 4
Three key performance indicator.

KPIs	Purpose of analysis
Execution time	Compare execution time of 3 schedules of ATAS, LATE and Hadoop Speculative on MinuteSort, WordCount, GrepCount, analyze execution time of these 3 schedules, confirm that ATAS could effectively enhance the throughput performance of MapReduce in a heterogeneous cloud environment.
Average task throughput	Compare average task throughput of 3 schedules of ATAS, LATE and Hadoop Speculative under different number of nodes, analyze changes in average task throughput of different nodes so as to verify that the ATAS method will foster average task throughput.
Average task latency	Compare average task latency of 3 schedules of ATAS, LATE and Hadoop Speculative under different number of nodes, analyze average task latency of task throughput under different nodes in order to verify that the ATAS method will reduce average task latency.

Table 5
Node configuration.

No. of nodes	Cloud groups		
	Cloud 1	Cloud 2	Cloud 3
10	3	3	4
20	6	7	7
30	10	10	10
40	10	15	15
60	10	25	25
80	10	30	40
100	10	30	60

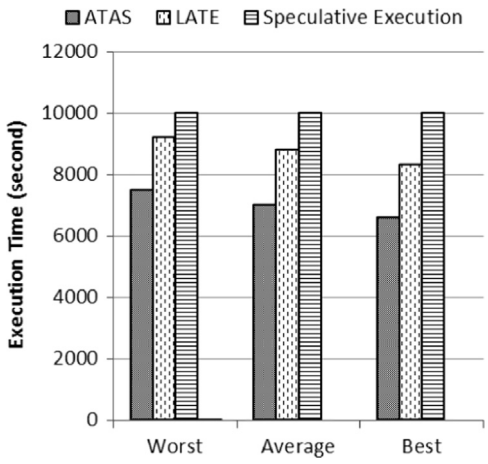

Fig. 4. The execution time result of MinuteSort.

by WordCount, a backup task is not needed, therefore, reporting small improvements over MinuteShort when executing WordCount.

Figure 9 indicates average task throughput of GrepCount by running different nodes 10 times, in which performance of ATAS will become eminent along with an increase in node number. When the node number is 100, the LATE will enhance average task throughput by around 13%, while ATAS will enhance average task throughput by around 23% compared to the Hadoop Speculative inferential execution. This is because GrepCount reports longer execution times than WordCount, which could effectively increase

Fig. 5. The execution time result of WordCount.

Fig. 6. The execution time result of GrepCount.

task throughput and further enhance the throughput performance of MapReduce when there are more successful backup tasks.

To summarize Figs. 7–9, an increase in node number increases system computing resources, which can enhance task throughput.

Table 6
The execution time of benchmark and improvements.

Benchmark	Scheduler	Worst case (s)	Average case (s)	Best case (s)	Average improve ratio (%)
MinuteSort	**ATAS**	**7492**	**7001**	**6596**	**30**
	LATE	9196	8788	8402	12
	Hadoop Speculative	10,001	9998	9997	–
WordCount	**ATAS**	**8497**	**8188**	**8003**	**18**
	LATE	9395	9102	9001	9
	Hadoop Speculative	9992	9987	9985	–
Grep	**ATAS**	**8133**	**7883**	**7665**	**21**
	LATE	9001	8813	8495	12
	Hadoop Speculative	9998	9994	9991	–

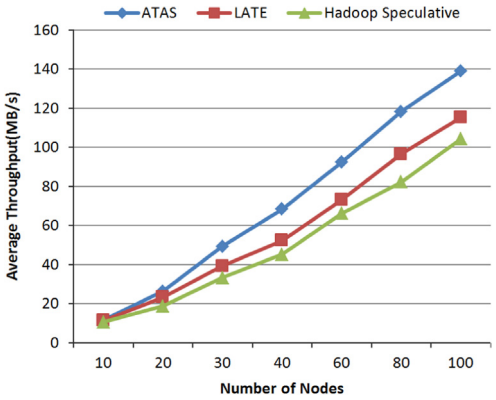

Fig. 7. The average task throughput on different nodes of MinuteSort.

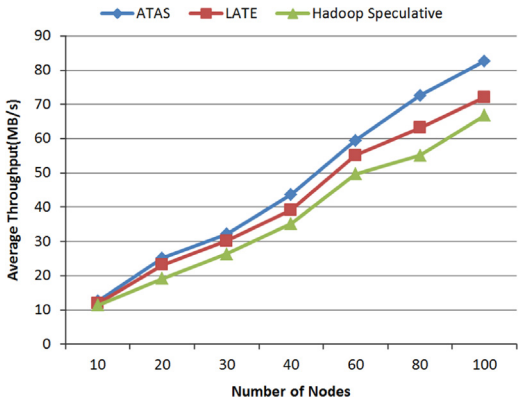

Fig. 8. The average task throughput on different nodes of WordCount.

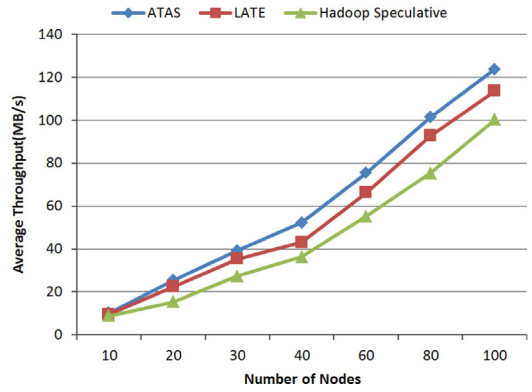

Fig. 9. The average task throughput on different nodes of GrepCount.

Table 7
The average task throughput of benchmarks and improvements.

Benchmarks	Schedulers	Worst case (MB/s)	Average case (MB/s)	Best case (MB/s)	Average improving ratio (%)
MinuteSort	**ATAS**	**134.23**	**138.95**	**140.17**	**33**
	LATE	112.99	115.28	117.42	11
	Hadoop Speculative	101.21	104.32	105.12	–
WordCount	**ATAS**	**120.16**	**122.95**	**124.22**	**17**
	LATE	112.12	115.32	117.73	9
	Hadoop Speculative	103.99	105.32	106.21	–
Grep	**ATAS**	**121.88**	**123.66**	**125.78**	**23**
	LATE	111.12	113.56	115.24	13
	Hadoop Speculative	99.12	100.26	101.26	–

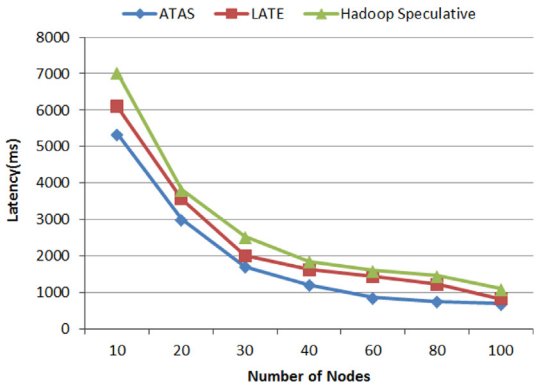

Fig. 10. The average task latency on different nodes of MinuteSort.

The use of ATAS is more applicable to the heterogeneous cloud environment, in which task throughput of 3 different types of benchmark test programs will be enhanced after use of ATAS. Table 7 indicates average task throughput of 3 benchmark test programs by 3 scheduling methods when the node number is 100, as well as under circumstances of the Average Case and the improvement ratio of the other 3 schedules compared to inferential execution of Hadoop.

Figure 10 indicates the average task latency of the MinuteSort program by running different nodes 10 times. An increase in node number will help improve average task latency. When node number is 100, LATE will reduce average task latency by 25% compared to Hadoop Speculative, while ATAS will reduce average task latency by 36% compared to Hadoop Speculative. This is because of the fast node and the slow node that are divided by LATE and ATAS, in which the fast node allows execution of the backup task while there are no any task to be allocated the slow node, which helps reduce waiting time among nodes and further reduce extra redundant time effectively. Also, we did find that ATAS can get more accurate estimation time of the remaining tasks to be compared with LATE, thus it can effectively reduce the task latency in the heterogeneous cloud environment.

Figure 11 indicates average task latency of the WordCount program by running different nodes 10 times. When the node number is 100,

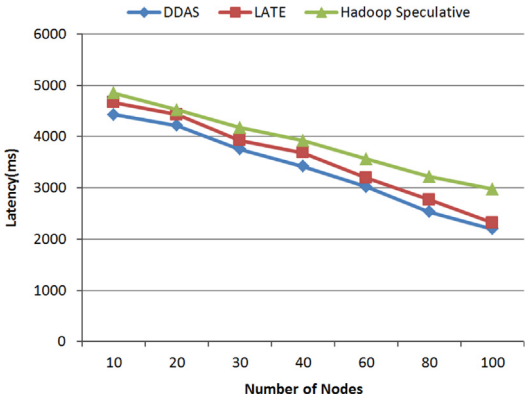

Fig. 11. The average task latency on different nodes of WordCount.

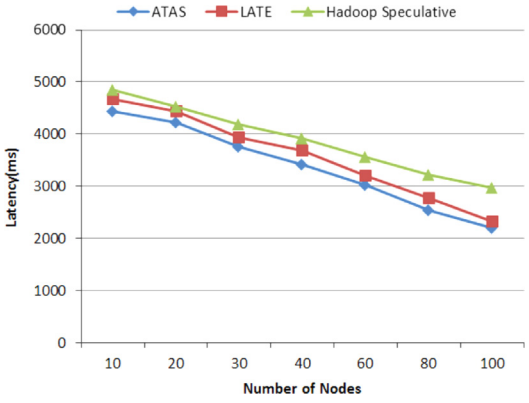

Fig. 12. The average task latency on different nodes of GrepCount.

Table 8
The average task latency of benchmarks and improvements.

Benchmarks	Schedulers	Worst case (ms)	Average case (ms)	Best case (ms)	Average improving ratio (%)
MinuteSort	**ATAS**	**706**	**699**	**685**	**36**
	LATE	831	820	798	25
	Hadoop Speculative	1123	1099	1069	–
WordCount	**ATAS**	**1621**	**1566**	**1478**	**34**
	LATE	2112	2012	1952	5
	Hadoop Speculative	2456	2365	2278	–
Grep	**ATAS**	**2288**	**2198**	**2107**	**26**
	LATE	2412	2322	2246	22
	Hadoop Speculative	3109	2974	2789	–

the LATE will reduce average task latency by 15% compared to Hadoop Speculative, while ATAS will reduce average task latency by 34% compared to Hadoop Speculative. This is because the massive computing required by the WordCount program execution at the Map stage, Reduces tasks on the fast node shall be conducted after completion of the small node. Hence, it causes additional waiting time, while ATAS could effectively reduce additional time.

Figure 12 indicates average task latency of GrepCount program by running different nodes 10 times, in which GrepCount shows small improvements compared to WordCount. This is because GrepCount requires no massive computing at the map stage compared to WordCount. The LATE will reduce average task latency by 22% whiling ATAS will reduce average task latency by 26% compared to Hadoop Speculative, when the node number is approaching to 100.

Figures 10–12 are summarized as Table 8, showing that ATAS can effectively reduce average task latency compared to the other 3 schedules, which become eminent especially when massive and complex computing is needed. It is verified that ATAS shows a more accurate forecast on expected task completion time than LATE Scheduler and Hadoop Speculative, in which average task latency could be effectively reduced while performance of average task throughput will be enhanced via reasonable distribution of tasks and execution of backup tasks on a straggler.

5. Conclusion

In this paper, we surveyed in-depth research on MapReduce, which is the core technology of Cloud Computing application development platform. Then, we proposed Adaptive Task Allocation Scheduling (ATAS) to find out the straggler through more reasonably calculated the TimeToEnd Time of each task. We divided the nodes into QuickNode and SlowNode. The QuickNode is prioritized with allocation of backup tasks. The comparisons of three different benchmarks show that LATE Scheduler and ATAS could be reduced by 12% and 30% maximum, average tasks throughput increased by 13% and 33% maximum, and average tasks latency reduced by 25% and 36% maximum comparing to Hadoop Speculative, indicating a more accurate speculative executing and reasonable allocation of backup tasks. Consequently, the proposed ATAS scheme can effectively enhance the processing performance of MapReduce to be compared with LATE Scheduler in a heterogeneous cloud environment.

Acknowledgments

The partial content of this paper is funded and supervised by Minister of Science and Technology in Taiwan, R.O.C. under Grant NSC 102-2410-H-031-061.

References

Almeer M. Cloud Hadoop MapReduce for remote sensing image analysis. J Emerg Trends Comput Inf Sci 2012;3(4).
Amin A, bockelman B, Letts J, Levshina T, et al., High throughput WAN data transfer with Hadoop-based storage. In: Proceedings of the international conference on computing in high energy and nuclear physics, vol. 331(5); 2011.
Armbrust M, Fox A, Griffith R, Joseph AD, et al. A view of cloud computing. Commun ACM 2010;53(4):50–8.
Chen Q, Zhang D, Guo M, Deng Q, Guo S. Samr: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In: Proceedings of the 10th IEEE international conference on computer and information technology; 2010. p. 2736–43.
Chen R, Chen H, Zang B. Tiled-MapReduce: optimizing resource usages of data-parallel applications on multicore with tiling. In: Proceedings of the 19th international conference on parallel architectures and compilation techniques; 2010. p. 523–34.
Dean J, Ghemawat S. MapReduce: simplfied data processing on large clusters. In: Proceedings of 5th symposium on operating systems design and implementation; 2008. p. 137–50.
Gu Y, Grossman R. Toward efficient and simplified distributed data intensive computing. IEEE Trans Parallel Distrib Syst 2011;22(6):974–84.
Hu W, Tian C, Liu X, Qi H, Zha L, Liao H, et al. Multiple-job optimization in MapReduce for heterogeneous workloads. In: Proceedings of 6th international conference on semantics knowledge and grids; 2010. p. 135–40.
Jiang W, Ravi T, Agrawal G. Comparing MapReduce and free ride for data-intensive applications. In: Proceedings of cluster computing and workshops; 2009. p. 1–10.
Kambatla K, Pathak A, Pucha H. Towards optimizing HADOOP provisioning in the cloud. In: Proceedings of the first workshop on hot topics in cloud computing; June 2009.
Kim S, Han H, Jung H, Eom H, Yeom H. Improving MapReduce performance by exploiting input redundancy. J Inf Sci Eng 2011;27(3):1137–52.
Kruijf M, Sankaralingam K. MapReduce for the cell broadband engine architecture. IBM J Res Dev 2009;53:10:1–12.

Pallis G. Cloud computing: the new frontier of internet computing. Intern Comput J 2010;14(5):70–3.
Rao BT, Reddy LSS. Survey on improved scheduling in Hadoop MapReduce in cloud environments. Int J Comput Appl 2011;34(9).
Rao BT, Sridevi NV, Reddy VK, Reddy LSS. Performance issues of heterogeneous Hadoop clusters in cloud computing. Glob J Comput Sci Technol 2011;6(8).
Seo S, Jang I, Woo K, Kim I, Kim JS, Maeng S. HPMR: prefetching and pre-shuffling in shared MapReduce computation environment. In: Proceedings of the 11th IEEE international conference on cluster computing; 2009.
Tian C, Zhou H, He Y, Zha L. A Dynamic MapReduce scheduler for heterogeneous workloads. In: Proceedings of the 2009 eighth international conference on grid and cooperative computing; 2009. p. 218–24.
Visalakshi P, Karthik TU. MapReduce scheduler using classifiers for heterogeneous workloads. J Comput Sci Netw Secur 2011;11(4):68–73.
Xu Z, Yan B, Zou Y. Beyond Hadoop: recent directions in data computing for internet services. Int J Cloud Appl Comput 2011;1(1):45–61.
Zaharia M, Konwinski A, Joseph AD, Katz R, Stoica I. Improving MapReduce performance in heterogeneous environments. In Proceedings of 8th USENIX symposium on operating systems design and implementation; 2008.
Zhou M, Zhang R, Zeng D, Qian W, Zhou A. Join optimization in the MapReduce environment for column-wise data store. In: Proceedings of the international conference on semantics knowledge and grid; 2010. p. 97–104.