Multi-node Scheduling Algorithm Based on Clustering Analysis and Data Partitioning in Emergency Management Cloud

Qingchen Zhang, Zhikui Chen, and Liang Zhao

School of Software Technology, Dalian University of Technology, Dalian, China, 116620
{qingchen,matthew1988zhao}@mail.dlut.edu.cn, zkchen@dlut.edu.cn,

Abstract. Real-time processing is a key problem for big data analysis and processing, especially in emergency management. Strongly promoted by the leading industrial companies, cloud computing becomes increasingly popular tool for emergency management, that is emergency management cloud. How to make optimal deployment of emergency management cloud applications is a challenging research problem. The paper proposes a multi-node scheduling algorithm based on clustering analysis and data partitioning in emergency management cloud. First, the presented method divides the cloud nodes into clusters according to the communication cost between different nodes, and then selects a cluster for the big data analysis services. Second, the load balancing theory is used to dispatch big data analysis to these computing nodes in a way to enable synchronized completion at best-effort performance. At last, to improve the real-time of big data analysis, the paper presents a multi-node scheduling algorithm based on game theory to find optimal scheduling strategy for each scheduling node. Experimental results show the effectiveness of our scheduling algorithm for big data analytics in emergency management.

Keywords: Big data, multi-node scheduling, emergency management cloud.

1 Introduction

In recent years we have witnessed efforts to open up the Internet of Things and to make its development more inclusive [1]. Recently, more and more embedded devices are joined in IoT to monitor all kinds of objects, including traffic facilities, buildings, and lakes and so on, which makes the size of the data very huge [2]. In other word, we are living in an era where data is being generated from many different sources such as sensors, mobile devices and RFID [3]. To be specific, sensors distributed in different geographic locations collect data continuously from various objects before the amount of long accumulated data is extremely huge [4]. Besides, collected data from numerous mobile devices for multi-target tracking can exceed hundreds of terabytes and be continuously generated. Such big data represents data sets that can no longer be easily analyzed with traditional data management methods and infrastructures and pose a huge challenge on emergency management [5].

Y. Gao et al. (Eds.): WAIM 2013 Workshops, LNCS 7901, pp. 165–174, 2013.

Recent emergency situations in the world show the tendency that the occurrence frequency of natural disasters is expected to increase in future [6]. The effects of natural disasters are very serious and the destruction caused may take a very long time to recover. Real-time analysis and processing for big data is a key to improve emergency management.

The advent of Cloud Computing has been enabling enterprises to deal with such big data in time in emergency management, which is also called emergency management cloud, by leveraging vast amounts of computing resources available on demand with low resource usage cost [7]. With emergency management cloud, any one can share data and information with others over the Internet. What is more important, even if the data centre is ever affected by a natural disaster, data are safe there as well as there are contingency plans to transfer data to other centers if a disaster can be forecast. In addition, massive sensor data collected from various sensors can be processed and responded in real-time with emergency management cloud, especially facing sudden disasters. Therefore, emergency management cloud has become a hotspot in research in recent year. When deploying a cloud application in an emergency management cloud, the application user needs to select a number of cloud nodes including servers and virtual machines to run the cloud applications. How to make optimal deployment of emergency management cloud applications is a challenging and urgent required research problem.

In order to optimize a parallel data analysis and processing in cloud environment, this paper addresses: (a) node selection, i.e., "how many" and "which" computing nodes in cloud should be used, (b) data partition and synchronized completion, i.e., how to optimally apportion big data across parallelized computation environments to ensure synchronization, where synchronization refers to completing all workload portions at the same time even when resources and inter-networks are heterogeneous and situated in multiple Internet-separated clouds, and (c) multi-node scheduling, i.e., how to use multi-node scheduling to reduce the latency of waiting for many tasks.

To address these problems, we develop a novel multi-node scheduling algorithm for big data analysis and processing based on clustering analysis and data partitioning in emergency management cloud. Most of current methods usually rank the available cloud nodes based on their QoS values and select the best performing ones. A drawback of the ranking methods is that these methods cannot reflect the relations between different cloud nodes. So, our method considers not only the QoS ranking of nodes, but also the communication relations between them. In addition, the load balancing theory is used to dispatch big data analysis to different computing nodes in a way to enable synchronized completion at best-effort performance. Besides, current cloud computing scheduling strategy assumes that only one node is responsible for scheduling, which will increase latency of waiting for scheduling. In order to process many tasks in real time, multiply nodes are required to attend to the task scheduling process. The paper views the task scheduling of multiple nodes as a non-cooperative game and constructs a task scheduling model based on complete information static game. At last, we find the optimal scheduling strategies for each node by seeking Nash equilibrium, making the average completion time of each scheduling node minimum.

At last, we design a series of experiments to evaluate the performance of the presented algorithm. The experimental results show that our approach outperforms other existing methods for big data analytics in real time in emergency management cloud.

2 Related Work

QoS can be employed for describing the non-functional performance of cloud nodes [8]. Based on the cloud node QoS performance, a number of selection and schedule strategies have been proposed in the recent literature [9]. These previous methods just consider the order of the node performance, and not consider the relationship between nodes. In this paper, we focus on analyzing the relationship between cloud nodes to achieve optimal deployment of cloud applications. Some approaches have introduced task schedulers with load balancing techniques in cloud computing environments[10]. These methods have mainly focused on keeping the order of tasks in the queue while increasing performance by utilizing an external cloud on demand. However, they do not consider how many and which clouds are required and how much data is allocated to each chosen cloud for parallel processing. Similar with our approach, research efforts for task scheduling have been made to deploy parallel applications over massively distributed computing environments to analyze big data, such as MapReduce, Pregel and Dryad and so on[11-13]. Using only one node for scheduling may get high performance for data analytics if the single node has enough computational power. However, when big data arrive in the same time, this method will increase the latency for data analysis. In this paper, multiple nodes are required to attend to the task scheduling process to improve the real-time for big data processing.

3 Scheduling Algorithm Based on Clustering Analysis and Data Partitioning

There are a number of available distributed nodes in the cloud. Cloud user need to deploy their cloud applications on a number of optimal cloud nodes and use it. We divide the cloud nodes into clusters mainly based on clustering method, making the communication between nodes in the same cluster smallest.

Assume there are n cloud nodes distributed in a cloud, the response times between nodes can be represented as an n by n matrix, where p_{ij} is the response time between node i and node j. Apparently, this matrix is a symmetric matrix.

$$P = \begin{pmatrix} 0 & p_{12} & \cdots & p_{1n} \\ p_{21} & 0 & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & 0 \end{pmatrix} \tag{1}$$

A cluster analysis algorithm is designed to divide the cloud nodes into K clusters.They satisfy the following condition.

$$\begin{cases} C_i \neq \varnothing \quad i = 1, 2, ..., K \\ C_i \cap C_j = \varnothing \quad i, j = 1, 2, ..., K \ and \ i \neq j \\ \bigcup_{i=1}^{K} C_i = D \end{cases} \tag{2}$$

The following formula is used to calculate the distance D between a node to the centroid of the K-th clustering.

$$D = \lambda \times | cal_i - cal_{ck} | + (1 - \lambda) \times \frac{1}{d} \sum_{j: j \in C_k} p_{ij} \tag{3}$$

After the completion of the clustering, the nodes in the cloud platform are divided into several clusters. And then we select one of the clusters to perform computing tasks. There are many nodes in a cluster, so the algorithm intends to parallelize the big analysis task by dividing input data for multiple computing nodes in the same cluster. The paper partitions the data based on the load balancing so that each node has the same processing delay, including communication delay and calculate the delay, to avoid the reduction of the real-time because of the delay of single node.

If the average delay of the task for a unit of data on a node i is denoted as t_i and t_i includes two part, data transmission time and data processing time. And then the overall delay for executing a data size s_i (that is provided to node i for processing task) is $s_i t_i$ .In order to ensure ideal parallelization for n nodes and a set of data, the following formula is satisfied.

$$T = s_1 t_1 = s_2 t_2 = ... = s_n t_n \tag{4}$$

Let s be the total amount of the data s, we can get the following formula.

$$T = s / \sum_{i=1}^{n} 1 / t_i \tag{5}$$

From the above formulas, we can get the following formula.

$$s_i = s / t_i \sum_{i=1}^{n} 1 / t_i \tag{6}$$

Depending on these formulas, the paper can find the optimal data partitioning solution.

4 Multi-node Scheduling Model Based on Complete Information Static Game

After the data arrive, they need to be deployed into compute nodes in the cloud computing platform for parallel processing. Current cloud computing scheduling strategy assumes that only one node is responsible for scheduling, however, in order to process many tasks in real time, multiple nodes are required to attend to the job scheduling process. The paper views the job scheduling of multiple nodes in the cloud computing as a non-cooperative game and constructs a task scheduling model based on complete information static game. And then, we find the optimal scheduling strategies for each node by seeking Nash equilibrium.

We assume there are m nodes responsible for scheduling n nodes for task execution in the cloud computing platform. For each scheduling node *i*, the amount of data that need to be distributed once is *d* and we assume that the size of data the computing node *j* can process every time is *y*. We assume that the probability that the scheduling node *i* sends data to the computing node *j* is *p*.

The average completion time of the task of every scheduling node consists of the data transfer time and the processing time in the computing nodes. The average completion time that the scheduling node i dispatches the task to the computing node j can be expressed as the formula (7).

$$R_{ij}(p) = \omega_j(p) + t_{ij} \tag{7}$$

Where, *w(p)* denotes the processing time of the computing node *j*, determined by the processing capacity of the compute node *j*, and *t* represents the transfer time from the node *i* to the node *j*. We assume that average bandwidth available from the node *i* to the node *j* is *c*, each transfer time can be expressed as formula (8).

$$t_{ij} = p_{ij}\overline{d_i} / c_{ij} \tag{8}$$

Above all, The average completion time that the task scheduling the node *i* schedules the task to n compute nodes can be expressed as the formula (9).

$$R_i(p) = \sum_{j=1}^{n} p_{ij} R_{ij}(p) = \sum_{j=1}^{n} p_{ij}\omega_j(p) + \sum_{j=1}^{n} p_{ij}^2 d_i / c_{ij} \tag{9}$$

Definition 1. The scheduling game model *G* is defined a *G=(I,S,U)*, where *I=(1,2,…,i,…m)* represents the set of game makers, namely *m* scheduling node. The pure strategy space of every scheduling node *i* is $S_i=\{1,2,\ldots,j,\ldots,n\}$. The mixed strategy space of the node *i* is $\sum=\{p_i\}$, where $pi=\{p_{i1},p_{i2},\ldots p_{im}\}$ is one of the mixed strategies of

the node *i*. $U=\{u_1,u_2,\ldots,u_m\}$ is the income of the scheduling nodes. According to the formula (9), The income function of every node is

$$u_i(p)=\sum_{j=1}^{n} p_{ij}R_{ij}(p)=\sum_{j=1}^{n} p_{ij}\omega_j(p)+\sum_{j=1}^{n} p_{ij}^2 d_i / c_{ij} \quad .$$

Definition 2. The Nash equilibrium of *G* is defined as $p^*=\{p_1^*,p_2^*,\ldots,p_i^*,\ldots p_m^*\}$, where p_i^* is one of the mixed strategies of the node *i*, if and only if $u_i(p_i,p_{i-1}^*)<u_i(p_i^*,p_{i-1}^*)$, for every scheduling node *i*.

Definition2 can make sure that every scheduling node gets the largest income, namely the smallest average completion time of task, as the formula (10).

$$\min\{u_i(p)\},\quad \sum_{j=1}^{n} p_{ij}=1\ ;\ p_{ij}\geq 0\ ;\ \sum_{i=1}^{m} p_{ij}d_i \leq y_j \tag{10}$$

The presented game model calculates the best mixed strategies according to the definition 2 and the formula (10), making the average completion time of every node smallest and meeting the real-time request for processing big data.

5 Experiment

5.1 Experiment Setup

Our experimental environment consists of 16 distributed nodes as cloud computing nodes, each of which has a 2.8GHz core, 1GB memory and 250GB hard drive, and 3 nodes as scheduling nodes, each of which has four 3.2GHz cores, 4GB memory and 500GB hard drive. The data in experiments are collected from the digital home lab, including three sets of data: temperature, humidity, and carbon dioxide concentration, and the total size is up to 80GB. In order to evaluate the performance of our presented algorithm, we mainly run the outliers detection algorithm, which is important to emergency management, such as helping detect abnormal data and position sudden event location, in our cloud environment. To compare the performance of our approach against other methods, the metric that we use is makespan, which is defined as the duration between sending out a job and receiving a correct result.

5.2 Impact of Data Transfer Delay on Overall Execution Time

We first show the performance characteristics of computing nodes in the context of data computation and data transfer delay. Fig.1 shows the performance characteristics when we run the mining task with the entire data set.

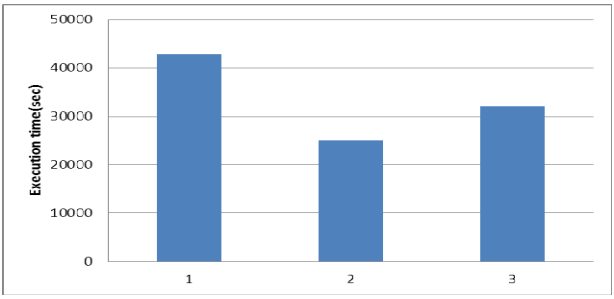

Fig. 1. Performance charateristics of computing nodes

In this figure, the first bar denotes the execution time of running in one virtual machine and the second bar represents that of running in 4 virtual machines, all of which are in the same computer, in parallel. The execution time of running of running in parallel in 4 virtual machines, however, any two of which are distributed in the different computers, is shown as the third bar. All of the machines have the same configuration. From this result, we can get two conclusions.

1) The outlier detection algorithm running in one virtual machine cost the longest time, while the execution time when using four virtual machines to run the algorithm in parallel is lower. This explains the need of parallel execution of big data analytics to improve the performance.

2) The execution time shown as the third bar is higher than that denoted by the second bar because of a significant delay for outlier detection of big data to get transferred over the different computing nodes. Therefore, we have to deal with the data transfer delay carefully.

5.3 Performance Comparison with a Single Scheduling Node

To study the performance of our presented method, we compare our method with the following two approaches.

Random-based: The scheduling algorithm with random-based cloud nodes selection.

QoS-based: The scheduling algorithm with ranking cloud nodes depending on the QoS of the nodes.

In this experiment, we partition cloud nodes in 3 clusters and use the parameter setting: $\lambda = 0.5$ and there is only one single scheduling node. The result shows that the execution time of all the algorithms increases as the size increases. Among all the methods, the scheduling algorithm with random-based nodes selection gets the worst performance and the result is not stable. Because of considering the QoS of cloud nodes during selecting cloud nodes for mining tasks, in most case, the performance of the QoS-based node scheduling algorithm is better than the random-based scheduling algorithm. However, when the size of data is smaller than 1.5GB, the execution time of the QoS-based node scheduling algorithm is lower than that of our method, because computing time occupies most of the overall execution time. With the increasing of

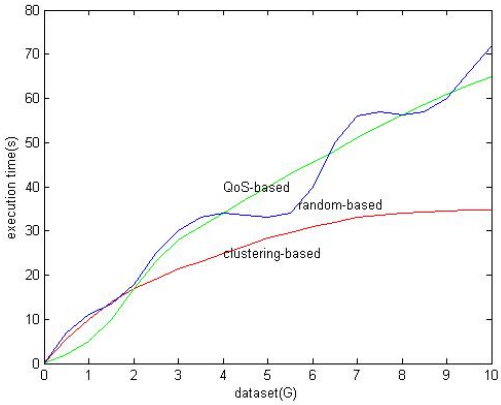

Fig. 2. Performance comparison of three algorithms

the size of data, especially when the size exceeds 2GB, the time of transfer and communication between different cloud nodes increases rapidly, our method obtains better performance than the QoS-based node scheduling algorithm which cannot consider the relations between cloud nodes. In conclusion, in most case, our method gets the best performance among all the scheduling approaches.

5.4 Performance Comparison with Multiple Scheduling Nodes

At last, in order to evaluate the performance of the presented multi-node scheduling algorithm based on game theory, we compare the multi-node scheduling algorithm with the scheduling method with single scheduling node. In this experiment, we use three scheduling nodes in the multi-node scheduling algorithm. The result is as shown in fig.3.

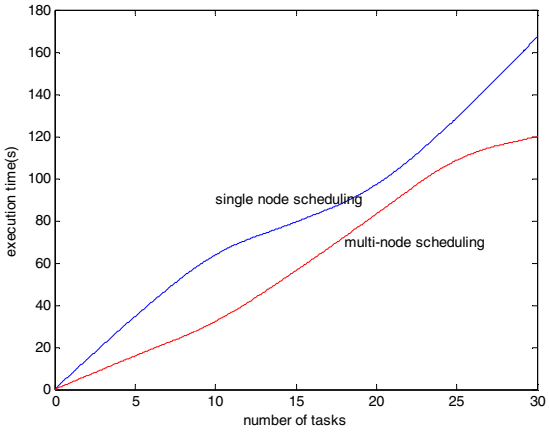

Fig. 3. Performance comparison of two scheduling algorithms

From fig.3, we can see that the execution time increases with the increasing of the number of tasks, but the multi-node scheduling algorithm has better performance than the scheduling algorithm with only one scheduling node. Because our method can schedule multiple tasks at the same time, reducing the waiting time of tasks.

6 Conclusion

Cloud computing is an effective tool for emergency management. With emergency management cloud, data can be processed in time and can be prevented from the sudden disasters. In this paper, we propose a novel multi-node scheduling algorithm to optimize the performance of big data analytics that can be run in distributed computing environment such as cloud computing platform. Generally speaking, our algorithm has supported decision makings on node selection, data partition and multi-node scheduling. We have compared our algorithm with other scheduling methods. Experiment shows that the performance of our algorithm is better than other approaches.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks 54(15), 2787–2805 (2010)
2. Kortuem, G., Kawsar, F., Fitton, D., et al.: Smart objects as building blocks for the internet of things. IEEE Internet Computing 14(1), 44–51 (2010)
3. Ashton, K.: That 'Internet of Things' Thing. RFiD Journal 22, 97–114 (2009)
4. Qin, X.-P., Wang, S.: Big Data Analysis—Competition and Symbiosis of RDBMS and MapReduce. Journal of Software 23(1), 32–45 (2012)
5. Cheng, Y., Qin, C., Rusu, F.: GLADE: big data analytics made easy. In: Proc. of the 28th International Conference on Management of Data, pp. 697–700 (2012)
6. Velev, D., Zlateva, P.: Principles of Cloud Computing Application in Emergency Management. In: Proc. of the International Conference on E-business, Management and Economics, pp. 119–123 (2011)
7. Iosup, A., Ostermann, S., Yigitbasi, M.N., et al.: Performance analysis of cloud com-puting services for many-tasks scientific computing. IEEE Trans. on Parallel and Distributed Systems 22(6), 931–945 (2011)
8. Zhang, Y., Huang, G., Liu, X.: Integrating resource consumption and allocation for infrastructure resources on-demand. In: Proc. of the 3rd IEEE International Conference on Cloud Computing, pp. 75–82 (2010)
9. Budati, K., Sonnek, J., Chandra, A.: Ridge: combining reliability and performance in open grid platforms. In: Proc. of the 16th International Symposium on High Performance Distributed Computing, pp. 55–64 (2007)
10. Frey, J., Tannenbaum, T., Livny, M.: Condor-G: A computation management agent for multi-institutional grids. Cluster Computing 5(3), 237–246 (2002)

11. Kim, H., Parashar, M.: CometCloud: An Autonomic Cloud Engine. Cloud Computing: Principles and Paradigms, 275–297 (2011)
12. Chen, Q., Hsu, M., Zeller, H.: Experience in Continuous analytics as a Service (CaaaS). In: Proc. of the 14th ACM International Conference on Extending Database Technology, pp. 509–514 (2011)
13. Huang, Y.C., Ho, Y.C., Lu, C.H., et al.: A cloud-based accessible architecture for large-scale adl analysis services. In: Proc. of the 4th IEEE International Conference on Cloud Computing, pp. 646–653 (2011)