A Study of Cloud Computing Scheduling Algorithm Based on Task Decomposition

Feng Gao & Wangcheng Luo
Department of Software Engineering, Chongqing University Of Arts and Science, Chongqing, China

Changyang Li
Beijing Institute of Technology, Beijing, China

ABSTRACT: The quality of cloud computing scheduling algorithm design can leave great impact on arithmetic speed. The key of restricting scheduling algorithm lies in task decomposition. This thesis conducted a study on task decomposition in cloud computing and applied a heuristic approach to design a task decomposition algorithm based on and-or-tree ideology. Also, it did a simulation test on this algorithm. The results of simulation test showed that: compared with those of traditional task decomposition algorithm, there are significant advantages of average handling capacity, time delay, average execution time, and task loss rate contained in the improved algorithm designed in this thesis.

Keywords: task decomposition; heuristic algorithm; cloud computing; simulation test

1 INTRODUCTION

In this year, cloud computing technology has experienced rapid development and caught great attention from related scholars in each field. Task scheduling is a key problem in cloud computing system. According to the differences contained in the classification of scheduling problems, scheduling policies can be divided into independent task scheduling and dependent task scheduling \[1, 2\]. Min-Min algorithm is a good choice to accomplish independent task scheduling \[3, 4\]. For task scheduling with dependent relations among tasks, directed acyclic graph (DAG) can be used to describe task relations. Previous and following task nodes shall be firstly processed, and then tasks shall be scheduled. Sakellariou, R, etc. (2004) proposed a heuristic scheduling algorithm based on DAG to divide tasks with the same weight as a group. And then, random heuristic algorithms can be used to schedule each task group according to priority order from lower to higher \[5\]. Xiaoli Du, etc. (2006) operated fuzzy clustering theory to process unit classification with similarity based on the dependent relations among tasks. They reduced the searching space of resources and shortened the overall scheduling time \[6\]. Jianning Lin, etc. \[7\] (2008) firstly applied heuristic algorithm to make scheduling graphs of production tasks; and then used task nodes on updated DAG optimized decision path and non-decision path to improve scheduling efficiency according to current scheduling relations, midpoint of DAG, and weights of sides. In order to optimize decomposition algorithms of cloud computing, many people have made great effort. Among which, LI A, etc. \[8\] (2011) improved fineness of decomposition tasks according to the current situation of task decomposition algorithm. However, they didn’t reduce the space complexity during decomposition. Bo Liu, etc. \[9\] (2006) proposed a network task decomposition algorithm to decompose complex tasks into subtasks with different priorities. Subtasks on the same priority layer can be executed at the same time. By this means, task execution time can be reduced and execution efficiency of server can be improved through feature description in combination of task priority. However, during their actual operation, these algorithms can easily fall in locally optimal solution and thus may affect the division strength of task execution. Based on analysis of cloud computing scheduling and task decomposition algorithm, this thesis used heuristic algorithm to design a task decomposition algorithm applicable for cloud computing task...
scheduling, aiming to provide foundation for rapid completion of cloud computing scheduling algorithm.

2 TASK SCHEDULING OF CLOUD COMPUTING

Task scheduling is the same as progress scheduling in tradition computer operating system. It refers to sending progress based on certain execution order to processor for execution and processing under the precondition that system resource restriction can be met, so as to minimize data communication. Cloud computing is development of grid computing, composed of mass commercial machines and high-performance servers. It can provide virtualized resources as required. Hence, the resource dynamic provided by cloud computing can contain high flexibility and great variation. Cloud computing is paid service for users to use. It has large user groups and there are significant differences between the preferences in user task. All these new features of cloud computing can cause many new problems in the scheduling of cloud task. See Figure 1 for the overview of task scheduling in cloud computing. It can be known from Figure 1 that cloud computing system relies on task scheduler to coordinate user service and resource management. Thus, there’re higher requirements for performance in task scheduling strategy. Users’ requirements for resources and services shall be paid more attention to.

In general situation, there are Optimal Makespan, Quality of Service (QoS), Load Balancing and Economic Principles in the optimization goal in cloud computing task scheduling [11]. According to the description given below, we can have the flow chart of task scheduling as described (see Figure 2):

**STEP 1.** Scheduler analyzes the received application program; identifies application program type; and writes analysis results to application configuration file.

**STEP 2.** Scheduler decomposes application program (that is task decomposition).

**STEP 3.** Scheduler reads resource description file; selects appropriate resources according to user demand; and set up resource allocation policy.

**STEP 4.** According to the set resource allocation policy, tasks shall be mapped to corresponding resources. Task scheduling is executed by this means.

3 TASK DECOMPOSITION

3.1 Formalization of task decomposition problem

See Figure 3 for task processing flow chart in which:
task decomposition problem can be defined as the structure given below:

\[ <K, A, E, I, G> \]

Among which: K refers to knowledge collection in the problem (initial condition, target, and intermediate result); A refers to cooperating collection (receive input information and compute result); E refers to execution unit collection (integrated operation to complete operation); I refers to initial condition collection (owned knowledge when task is proposed); and G refers to target collection (knowledge must be obtained when task is completed). In general, I and G define what need to completed while K, A and E define the environment that task completion relies on. The optimal feasibility obtained by task decomposition must accord with the following three conditions:

- **Condition 1.** Input information is very important. This information must be known before any operation is executed.
- **Condition 2.** All knowledge contained in G must be obtained.
- **Condition 3.** All overheads shall be minimized.

The above three conditions can be transformed into the five constraint conditions as given below:

**Constraint 1.** Each server does intelligent execution for once, that is:

\[ \forall i \left( \sum_k E_{ik} \leq 1 \right); \forall i \left( \sum_k E_{ik} = E_{ik} \right) \]

**Constraint 2.** Output collections of all operations must be covered for target collection, that is:

\[ \forall i \left( \sum_k O_{j_k} E_j \geq H \right) \]

**Constraint 3.** Each operation can only be executed when only input information exists, that is:

\[ \forall q \forall i \left( \sum_t E_{t_j} T_{ij} \geq I_{j_k} E_j \right) \]

**Constraint 4.** After task is decomposed, the executed operation sequence must be feasible. If \( R_{ij} \) is set as the transitive closure of \( T_{ij} \), it means:

\[ \forall i \forall j \left( R_{ij} \geq T_{ij} \right); \forall i \forall j \forall k \left( R_{ik} + R_{kj} \leq R_{ij} + 1 \right); \forall i \left( R_{ii} = 0 \right) \]

**Constraint 5.** The value of the overhead function as shown in Formula (1) is the minimum one.

\[ \sum_i \sum_j Z_{ij} \text{ExecFun} \left( A_i, A_j \right) + \sum_j \sum_j W_i \text{Commfun} \left( E_i, E_j \right) \]

In Formula (5), ExecFun(\( A_i, A_j \)) refers to execution overhead for server to execute operation i. Commfun(\( E_i, E_j \)) refers to communication overhead between two servers.

### 3.2 Design of heuristic algorithm

Heuristic algorithm is proposed with respect to optimal algorithm. An optimal algorithm of a problem is the optimal solution of each case. Heuristic algorithm can be defined as an algorithm constructed on basis of perceptual intuition or experience. It is a feasible solution for each case of a combined optimization problem needs to be solved under acceptable division. Generally, the degree of deviation between this feasible solution and the optimal solution cannot be predicted. The basic idea contained in the heuristic algorithm of this thesis is to simplify complex task decomposition problem into problems with feasible operating collections. The essence is to confirm the feasible operating collections of each group under task constraint conditions. As shown in Figure 4, heuristic decomposition solution is designed.

![Figure 4. Heuristic decomposition algorithm.](image-url)
task decomposition problem into two: one is to confirm a group of operations which can satisfy the constraint conditions of the task; the other is to execute the operation scheduling confirmed in the last step and confirm the optimal solution. While dividing task, each subtask shall be stored in a node of “and-or-tree”. The process of user’s request of task decomposition is the constructing process of the tree. Heuristic approach can be applied to find the division points of the task. Task decomposition is a tree-form structure with layers. The root node of the tree is the entire task. There’re collaborative or restricting conditions among nodes on the same layer of the tree.

Heuristic approach is applied to find division points of task. Storage of each subtask can accord with the storing idea of and-or-tree. After division points are selected, overall consumption of each subtask after division shall be predicted. When any division point is found to be improper, each subtask cannot be executed. At this time, it can be assumed that the decomposed subtask execution costs too infinite consumption while the communication consumption between subtasks will also tend to be infinite. When no other division point can be found on any leaf node, task decomposition is completed. See the left part of Figure 5 for task decomposition. Between Task C and Task D is the “or” branch which needs to be clipped. After the task is decomposed into several subtasks, there’re different relations among subtasks. The task arrangement process is to calculate the overhead of each branch according to Formula (5) and keep the branch with the minimum overhead. Remove all the other branches and delete the root node of the branch with the minimum overhead. See the right part of Figure 5 for the result after arrangement.

4 SIMULATION EXAMPLE AND RESULT ANALYSIS

4.1 Experimental design

The experimental environment is as follows:
A) 1 physical machine and 20 virtualized machines with Linux2.16.8 system installed.
B) Hadoop installation package;
C) Java installation package;
D) Eclipse-SDK-3.4.1-Linux-gtk.tar.gz.

Full distribution mode was applied. To analyze by HDFS, the nodes can be divided as Namenode and Datanode, between which there is only one Namenode but several Datanodes. To analyze by MapReduce, the nodes can be divided as JobTracker and TaskTracker, between which there’s only one JobTracker but several TaskTrackers. In this experiment, Namenode and JobTracker were deployed on the same PC. The PC worked as the Master node machine while the rest 20 PCs worked as Slaves (following nodes).

See the key steps as follows:

STEP1. Configure all the 21 PCs according to /etc/hosts file. Hosts file of the Master node machine shall be edited into the IP address of Master node machine and the IP addresses of the other 20 following node machines. Add the IP address of the Master node machine and the addresses of their own into the hosts files of the 20 following node machines.

STEP2. SSH configuration. This configuration mainly works to accomplish seamless connection between machines which can execute orders without any password.

STEP3. Hadoop configuration. Firstly, extract the downloaded Hadoop-0.19.1.tar.gz., and then edit conf/Hadoop-site.xml file. Modify conf/Master and conf/Slave files and copy the installation file of Hadoop to other node machines. Hadoop will be able to provide shell orders realted to control progress execution, such as start-all.sh and startMapReduce.sh orders.


4.2 Result analysis

Under the condition in which data flow load works as impact factor, contrast experiment between assessment handling capacity and time delay was conducted. See Figure 6 for the comparison results of SL (Stream Load), IHA (Improved Heuristic Algorithm), HA (Heuristic Algorithm), SPF (Shortest Path First), and BF (Best First):

It can be known from Figure 6 that the handling capacity of IHA was higher than those of the other three algorithms with different data flow loads. However, the variations occurred in the experimental results of the other three algorithms were insignificant while data flow load changed. When flow load parameter SL was increased from 2.0 to 2.4, the handling capacity of IMP algorithm was increased form 17MB/s to 27MB/s, meaning that the handling capacity of IHA algorithm had significant increase with the increase of application task request. The route time
delay of IHA was significantly lower than those of the other three algorithms as IHA algorithm had found the minimum spanning tree with the shortest execution time from complex DAG graph. Compared with other algorithms with route nature, IHA algorithm can save route search time to a great extent. However, it transforms DAG graph into the minimum spanning tree as the cost. The performance of all the other three algorithms is close. Their delay time increased with data flow assistance. Especially when data flow load was 2.4, the delay time of the three algorithms had great increase as each algorithm has its own fixed critical state and when the critical state value is exceeded, algorithm performance will suffer from great attenuation.

Figure 7 shows the comparison between average execution time and task loss rates of IHA algorithm and HA algorithm. As a whole, the average execution time of IHA algorithm was almost twice of that of HA algorithm. Moreover, with the increase of quantity of batch processing task, the loss rate of IHA algorithm got higher as during the process of IHA transforming DAG into the minimum spanning tree, some dependent relations got lost and caused increase in the loss rate.

Figure 8 shows the task loss rates of different models.
els. The task loss rate situation shown in Figure 8 was with quantity of batch processing task as 100, 200 and 300. As a whole, the task loss rate of IHA algorithm was kept lower than 0.025 and was lower than those of all the other three models. To see from single model, the task loss rate of IHA algorithm got higher with increase in quantity of batch processing task. When quantity of batch processing increased from 100 to 300, the task loss rate became higher from 0.017 to 0.024. However, the loss rate could be kept within an acceptable range.

5 CONCLUSIONS

There are two directions for future development of cloud computing: one is constructing large-scale basis infrastructures in close combination of application programs, so as to expand applications to a larger scale; the other is studying new scheduling algorithms which can better fit detailed applications, so as to satisfy various user demands and enrich user experience. However, the key factors restricting the two development directions of cloud computing are computing speed and precision. According to task decomposition algorithm this thesis conducted related study and designed a task decomposition algorithm based on heuristic algorithm. Through simulation comparison, it can be known that compared with other three algorithms, the improved task decomposition algorithm contains significant advantages. Based on the conclusion, the improved task decomposition algorithm shall be promoted to the scheduling of cloud computing for better application prospect.

REFERENCES