CHAPTER 4

AN ACO ALGORITHM FOR SCHEDULING DATA INTENSIVE APPLICATION WITH QOS REQUIREMENTS

4.1 INTRODUCTION

Due to the uncertainty and nondeterministic characteristics of resources, many existing algorithms for scheduling data intensive applications can only tackle the problems, with either system centric QoS or application centric QoS. We proposed a new algorithm based on ant colony optimization to adopt the challenges and schedule the data intensive application, which combines both application centric and system centric QoS. The simulation results demonstrates the effectiveness of the proposed scheduling algorithm, by optimizing the time and cost of an application and maximizes the resource utilization.

The scheduling phase aims at finding the best match between a set of jobs and available resources, which is a NP-hard problem. For example, there are $10^{35}$ mappings possible for 25 jobs with 10 resources. The ACO algorithm, which is a population-based approach, has been successfully applied to many NP-hard optimization problems.

Previously, Zh et al (2003) proposed a task scheduling algorithm based on ACO algorithm. The algorithm showed good performance in terms
of response time and average resource utilization. However, the authors emphasized that the algorithm was effective for scheduling computation-intensive applications and not data-intensive applications.

A new instance of QoS-based algorithm for job allocation and scheduling, put forth by Xiangang Zhao et al(2006), calculates the QoS of resources (such as processing power and communication ability) and provides a reallocation mechanism in the event of resource failure. However, the algorithm works well only under stable conditions.

4.2 ACO ALGORITHM FOR JOB SCHEDULING

ACO has been known to be an effective strategy for several problems related to scheduling of jobs in data-intensive applications. The ants build the solution using both information encoded in the pheromone trail and also problem-specific information in the form of a heuristic.

1. Initialization of algorithm

All the pheromone values and parameters are initialized at the beginning of the algorithm.

2. Solution construction

N artificial ants are used in the algorithm, which set out to build N solutions to the problem based on pheromone and heuristic values using the selection rule.
3. **Pheromone updating**

At the end of each iteration, all ants complete their solution and the pheromone values are updated.

### 4.3 DEFINING PHEROMONE AND HEURISTIC INFORMATION

The following notions are defined in the mathematical model:

- $T_i$ - Deadline given by user $i$
- $B_i$ - Budget of grid user $i$
- $P_j$ - Unit price of resource $j$
- $W_j$ - Total workload of resource $j$
- $C_j$ - Current workload of $j$
- $\text{TR}_i$ - Time required for completing the job at resource $j$

When a resource $j$ joins the grid, it should submit the quality factors in set $S$:

$$\tau_j(0) = \sum_{i=1}^{n} s_i f_i, \quad \sum_{i=1}^{n} f_i = 1 \tag{4.1}$$

where

- $\tau(0)$ - Innate performance of resource $j$
- $f_i$ - Intensive weightage factors
- $s_i$ - System-centric parameters
The probability that the task is allocated to resource $j$ ($j = 1, 2 \ldots m$) within a job is computed using the formula (Yan et al 2005),

$$P_j(t) = \frac{[\tau_j(t)]^\alpha [\tau_j(0)]^\beta}{\sum_{j'}[\tau_{j'}(t)]^\alpha [\tau_{j'}(0)]^\beta} \quad \mu = 1, 2 \ldots n$$ (4.2)

where

- $\tau_j(t)$ - Pheromone intensity on the path from scheduler to resource $j$ at time $t$
- $\tau_j(0)$ - Innate performance of resource $j$
- $\alpha$ - Importance of pheromone
- $\beta$ - Resource-innate attribute
- $\mu$ - Resource available for the job

The equation 4.3 is used to update the pheromone intensity on the path from scheduler to the corresponding resource as follows:

$$\tau_j^{new} = \rho \tau_j^{old} + \Delta \tau_j$$ (4.3)

where

1. $\tau_j^{new}$ is the change of pheromone on path from the scheduler to resource $j$.
2. $\rho$ is evaporation of pheromone ($0 \leq \rho \leq 1$).
3. When a task is allocated to resource $j$, $\Delta \tau_j = -K$; $K$ is the quality the task consumed.
4. When a task is canceled and the data resource it used is still in service, $\Delta \tau_j = K$. This will restore the resource quality.

5. When a task successfully returns from resource $j$, $\Delta \tau_j = C_e \cdot K$, $C_e \geq 1$; $C_e$ is the encouraging factor.

6. Otherwise, if the task fails in returning from resource $j$, $\Delta \tau_j = C_p \cdot K$; $C_p$ is the punishing factor and $0 \leq C_p \leq 1$.

The following steps are used to find optimal values for these parameters. These steps are added in the algorithm to adjust automatically based on the resource status. Environment variable, $B_j$, is calculated as follows:

$$B_j = \frac{1}{2} \left( \frac{\text{Resource failure rate (\%)} + \text{Network Stability (\%)}}{100} \right)$$

where

$$\text{Network stability} = \frac{\text{Current bandwidth}}{\text{Required bandwidth}}$$

### 4.4 PROCEDURE FOR ADJUSTING THE INNATE AND DYNAMIC ATTRIBUTES

Procedure ParaAdj

Begin

Compute $B_j$

If $(B_j \geq 0.9)$

Assign $\beta = 0.5$, $\alpha = 1 - \beta$

Else If $(B_j \geq 0.8$ and $B_j < 0.9)$
Assign $\beta = \beta + 0.1$, $\alpha = 1 - \beta$

Else If ($B_j \geq 0.6$ and $B_j < 0.8$)
Assign $\beta = \beta + 0.2$, $\alpha = 1 - \beta$

Else If ($B_j < 0.6$)
Assign $\beta = \beta + 0.3$, $\alpha = 1 - \beta$

End if

End if

End if

End if

End if

End ParaAdj

4.5 RESOURCE ALLOCATION

Once the user submits a job with relevant QoS parameters, such as cost and deadline, it enters into the job pool and lineup. The job is broken down into several simpler tasks. Each task is allocated to the resource according to equation (4.2) subject to the following conditions.

1. Deadline given by the user ($T_i$) is less than total expected time for execution of all tasks.

2. Budget given by the user ($B_i$) is less than the cost associated with all the resources used to execute the tasks.

3. Total workload ($W_j$) is less than current workload ($C_j$).
When a job is submitted to the grid, the scheduling system has the responsibility of selecting the suitable resources to manage the job execution. The decision of which resource to use is the outcome of scheduling algorithm. The simulation parameters used to evaluate the proposed algorithm are given in Table 4.1.

### Table 4.1 Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of grid sites</td>
<td>150</td>
</tr>
<tr>
<td>Number of nodes in each site</td>
<td>30</td>
</tr>
<tr>
<td>Bandwidth capacity</td>
<td>200 Mbps to 2 Gbps</td>
</tr>
<tr>
<td>Processing capacity</td>
<td>512 to 1024 MIPS</td>
</tr>
<tr>
<td>Total number of providers</td>
<td>300</td>
</tr>
<tr>
<td>Total number of users</td>
<td>800</td>
</tr>
<tr>
<td>Budget of users (units)</td>
<td>100 to 1000</td>
</tr>
<tr>
<td>Price of resources (units)</td>
<td>100 to 2000</td>
</tr>
<tr>
<td>Number of files</td>
<td>100 to 15000</td>
</tr>
<tr>
<td>Size of each file</td>
<td>100 MB to 4 GB</td>
</tr>
</tbody>
</table>

Pheromone ($\rho$) is critical for the success of the algorithm. It could be set to a higher value in high-steady environments or a smaller value in other environments. $\beta$ determines the importance of heuristic information. Again, different values could be tested. A high $\beta$ value is necessary to achieve
a good solution in dynamic environment. Heuristic intensive weightage factors, parameters $\alpha$ and $\beta$, and how their variations affect the performance of the algorithm are studied in detail. Their values are decided by the stability of the network and resource condition. Quality and consistency of the network are among the most important factors evaluated. By adjusting the parameters across trials, the performance of the algorithm appears to improve.

Using GridSim simulation toolkit, the performance of the algorithm is evaluated to be compared with two previous algorithms. To obtain realistic results under different conditions of loads, a set of diverse jobs are submitted at different intervals. Intensive weightage factors, $I_1$ and $I_2$, are added to equation (4.1) for making good scheduling decisions and prioritizing the QoS parameters. The heuristic value is driven by these intensive weightage factors. A set of experiments are conducted to evaluate the behavior of the algorithm. The network interface in the simulation toolkit provides dynamic information about the status of the network.

It is decided to compare the performance of the proposed algorithm in terms of two metrics (Tables 4.2 and 4.3), namely economic cost and makespan, which other researchers have also used; for example, in an application-centric algorithm presented by Venugopal et al (2005) and a system-centric algorithm presented by Zhao et al (2006). Economic cost is the minimum cost spent for executing a job. Makespan is the minimum time spent from the beginning of the first task in a job to the end of the last task in the job. Experiments are conducted with different input values. When tested in a simulation environment, the experiments showed that the ant heuristic method could produce a considerable improvement in performance and
increase the reliability of data transfer in a dynamic network as reported in Chapter 3.

A result of the experiments and the improvement in performance of the adaptive QOS algorithm in terms of makespan is shown in Table 4.2. The makespan of the job is mentioned in minutes. When number of jobs increased, the performance in terms of makespan also increased comparatively with existing application and system centric adaptive QoS algorithm. In the result of second experiment shown in Table 4.3 express the performance of proposed algorithm in terms of cost reducing comparatively with existing application and system centric algorithm.

**Table 4.2 Performance of proposed ACO algorithm with makespan**

<table>
<thead>
<tr>
<th>No. of jobs</th>
<th>No. of resources used</th>
<th>Makespan of Application-centric algorithm (Minutes)</th>
<th>Makespan of System-centric algorithm (Minutes)</th>
<th>Makespan of Proposed algorithm (Minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>14</td>
<td>98</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>100</td>
<td>34</td>
<td>145</td>
<td>128</td>
<td>118</td>
</tr>
<tr>
<td>500</td>
<td>84</td>
<td>458</td>
<td>396</td>
<td>352</td>
</tr>
<tr>
<td>1000</td>
<td>104</td>
<td>764</td>
<td>748</td>
<td>651</td>
</tr>
<tr>
<td>2000</td>
<td>134</td>
<td>1167</td>
<td>954</td>
<td>812</td>
</tr>
</tbody>
</table>
Table 4.3  Performance of proposed ACO algorithm with cost

<table>
<thead>
<tr>
<th>No. of jobs</th>
<th>No. of resources used</th>
<th>Cost of application-centric algorithm (Units)</th>
<th>Cost of system-centric algorithm (Units)</th>
<th>Cost of Proposed algorithm (Units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>14</td>
<td>592</td>
<td>528</td>
<td>502</td>
</tr>
<tr>
<td>100</td>
<td>34</td>
<td>881</td>
<td>825</td>
<td>772</td>
</tr>
<tr>
<td>500</td>
<td>84</td>
<td>1528</td>
<td>1328</td>
<td>1196</td>
</tr>
<tr>
<td>1000</td>
<td>104</td>
<td>1826</td>
<td>1718</td>
<td>1486</td>
</tr>
<tr>
<td>2000</td>
<td>134</td>
<td>2146</td>
<td>1974</td>
<td>1679</td>
</tr>
</tbody>
</table>
Figure 4.1 Performance of proposed ACO algorithm with makespan
Figure 4.2 Performance of proposed ACO algorithm with cost.
From the graphs in Figures 4.1 and 4.2, it is evident that the proposed scheduling algorithm is considerably effective in reducing the time and cost as compared to the two other algorithms. The parameters $\alpha$, $\beta$ and intensive weightage factors $I_1$, $I_2$ play a significant role. A major advantage in this adaptive scheduling algorithm is that as the number of jobs increases, it finds only those sites with adequate network bandwidth and least load for job execution. Experimental results showed that the algorithm is workable even under unreliable resource and network conditions.

4.7 CONCLUSION

Adaptive resource scheduling is a prospective approach for data-intensive applications to provide the desired services with appropriate QoS. This chapter proposes a novel scheduling algorithm that combines both application-centric and system-centric benefits. Experimental results suggest that optimization can be achieved when in scheduling data-intensive applications. The proposed algorithm could significantly reduce the total execution time and cost even in unreliable resource and network conditions. Intensive weightage factors, parameters of heuristic value, and pheromone intensity are the key elements optimized in the scheduling algorithm.