An Efficient and Energy Saving DPS Algorithm Based on Adaptive Threshold in Cloud Computing

Guoyuan Yang, Tianyun Shi, Qiuliang Zhang

China Academy of Railway Sciences, Beijing 100081, China

ABSTRACT: With the increasing promotion and application of the cloud computing system, there are more and more servers in the system, which makes the data-handling capacity more and stronger and adds the energy consuming. Meanwhile, if the utilization rate of servers is low, much energy is wasted. To solve the server scheduling in clouding computing system, an efficient and energy saving DPS algorithm based on adaptive threshold is proposed. The proposed algorithm can’t only dynamically adjust the threshold according to the energy cost and number of the leaving task, which can reduce the energy cost. And it can balance the use of reserved servers, which can prolong life of server. Simulation demonstrates that compared with ETSP algorithm and DPECO algorithm, the proposed algorithm can efficiently reduce the energy cost in the condition of ensuring the average response time.

1 INTRODUCTION

There are more and more servers in the cloud computing system with continuous popularization and application of cloud computing system, and the processing capacity becomes more and more strong. Increase of server quantity must lead to increase of system energy consumption, wherein a lot of electricity can be wasted by lower utilization rate of servers [1] because the servers in the cloud computing system are generally in an open state and wait for processing task requests, thereby system tasks have certain randomness. When there are fewer tasks, a part of servers in the system are in the idle state, and power loss is caused. Therefore, server energy consumption optimization belongs to an important direction of cloud computing system optimization research.

Currently, research on server energy consumption optimization can be divided into two categories [2]: one is research and development on more energy-saving hardware equipment; the other is to reach energy-saving purpose through idle server in the predetermined strategy closing system, wherein DPS (Dynamic Powering on/off Service) algorithm is a commonly used algorithm in the second category [3]. DPS algorithm can control turn-on and turn-off of the server in the system on the basis of corresponding control strategy. Idle servers under the open state in the system can be effectively reduced, thereby realizing reduction of system energy consumption. Control strategies in the DSP algorithm can be divided into four categories [4]: randomized strategy, forecasting strategy, timeout strategy and reservation strategy. The former three control strategies cannot easily adapt to dynamic changes of task quantity. In addition, they also have problems of algorithm complexity, poor applicability, etc. DPS algorithm based on the reservation strategy has advantages of simple realization, high efficiency, strong adaptability, etc., and thereby the algorithm can be mainly focused. In DPS algorithm based on reservation strategy, servers in the cloud computing system are divided into two categories of SRM (Service Reserved Module) and SMM (Service Main Module). Servers in SMM are always in the open state and wait for task processing. Servers in SRM can determine turn-on and turn-off of servers according to system state. Static threshold is set by DPECO algorithm proposed in Literature [5] according to vacation queue model, thereby realizing control on server status in SRM. However, static threshold may cause system jitter. Namely, servers in the SRM may be opened and closed frequently, thereby resulting in waste of energy. Servers are scheduled according to ‘hot-spot area’ in the system in Literature [6]. Tasks can be allocated and servers can be scheduled according to CPU computing capacity in Literature [7]. Servers with strong processing capacity can be frequently used by the two algorithms, thereby causing algorithm of server service life. In the paper, an Efficient and Energy saving DPS Algorithm based on Adaptive Threshold (EEDPSAT) is proposed aiming at disadvantages of the above-mentioned. The algorithm not only can dynamically adjust threshold
according to the energy consumption costs and task separation quantity, but also can balance use of servers in SRM through server scheduling, thereby prolonging service life.

2 SYSTEM TASK SCHEDULING MODEL

Task scheduling model of cloud computing system can be seen as M/M/n queue scheduling model [8], namely when task requests are received, the task controller in the system can firstly assign tasks to idle servers in SMM (Service Main Module). If all servers are busy, the request can enter the waiting queue. Meanwhile, once the task request quantity in the waiting queue is higher than the preset threshold, the state manager in the system can start a part of reserved servers in SRM (Service Reserved Module) for handling tasks in the waiting queue in order to control length of the waiting queue. In addition, the waiting time of each task is not infinite. It can leave from the waiting queue once time expires, thereby producing 'task separation'.

Figure 1 shows system task scheduling model based on $M/M/n$. It is assumed that task request quantity service parameter received by cloud computing system within unit time belongs to $\lambda$ poisson distribution, and the task controller can schedule task requests according to the criteria of serving the early arrived requests. The average service rate for the server is $\mu$. In addition the service process of each server and reaching process of each task request are mutually independent. There are a total of $N$ homogeneous servers in the system, wherein there are $n$ ($0 \leq n \leq N$) servers in SMM, then the remaining $N-n$ servers belong to SRM. The task waiting queue $Q_{\text{global}}$ is represented by $(k, \Phi_k, n_{\text{on}})$ wherein, $k$ belongs to task quantity of waiting queue, $\Phi_k$ refers to task separation intensity (its value is related to $k$, $\Phi_k \to \infty$ when $k \to \infty$). $n_{\text{on}}$ refers to quantity of servers ($n \leq n_{\text{on}} \leq N$) which are totally started in the system. State manager can determine whether the servers in the SRM are in the turn-on or turn-off state according to the relationship between $Q_{\text{global}}$ and threshold.

3 ONE EFFICIENT AND ENERGY-SAVING DPS ALGORITHM BASED ON ADAPTIVE THRESHOLD IN CLOUD COMPUTING

3.1 Adjustment of adaptive threshold

The threshold size is directly related to performance and energy consumption of cloud computing system. Smaller threshold can effectively reduce service delay. However, frequent stop and stop of servers in SRM can increase additional power loss. Though larger threshold can effectively reduce the system energy consumption, the task separation quantity can be quickly increased, thereby resulting in reduction of system performance. Energy consumption cost and task separation quantity should be comprehensively considered for setting of threshold, thereby maximizing efficiency.

3.1.1 Average task separation quantity

The calculation formula of average task separation number $L$ is shown as follows:

$$L = \sum_{i=1}^{k} \Phi_i p_i$$  \hspace{1cm} (1)

Wherein, $p_i$ is the probability of task queue length $i$.

Formula (1) shows that $\Phi_i$ and $p_i$ under all states must be firstly known in order to determine L value. It is assumed that $\Phi_i = \delta \times i$ ($0 < \delta < 1$) in order to facilitate analysis, meanwhile $\rho = \lambda / \mu$, $\beta = \delta / \mu$ is set. In addition, $K$ is total task request quantity received by cloud computing system. Then $p_i$ can be solved according to K's equation.

If $K \leq n_{\text{on}}$:

$$\lambda p_0 = \mu p_1$$ under the state of 0, then,
\( p_i = \frac{\lambda}{\mu} p_0 = \rho p_0 \)  

(2)

\( \lambda p_{n-1} = n \mu p_n \) under the state of \( n-1 \), then,

\( p_n = \frac{\lambda}{n \mu} p_{n-1} = \frac{\rho^n}{n!} p_0 \)  

(3)

If \( K > n_m \):

\[ \lambda p_n = (n \mu + \delta) p_{n+1} \] under the state of \( n \), then,

\[ p_{n+1} = \frac{\lambda}{n \mu + \delta} p_n = \frac{\rho^{n+1}}{n!(n+\delta)} p_0 \]  

(4)

According to formula (4), \( p_{n+k} \) is

\[ p_{n+k} = \frac{\rho^{n+k}}{n!(n+\beta)(n+\beta)\cdots(n+k\beta)} p_0 \]  

(5)

Since \( \sum_{i=0}^{n+k} p_i = 1 \), formula (2) to formula (5) are replaced into \( \sum_{i=0}^{n+k} p_i = 1 \), \( p_0 \) is:

\[ p_0 = \left[ \sum_{j=0}^{n+k} \frac{\rho^j}{j!} \left( 1 + \frac{\rho}{n+\beta} + \cdots + \frac{\rho^k}{(n+\beta)\cdots(n+k\beta)} \right) \right]^{-1} \]  

(6)

3.1.2 Energy consumption cost

\( P_{\text{busy}} \) is set to be power consumption during task request processing by single server; \( P_{\text{idle}} \) is set to be power consumption of single server during idle state. \( P_{\text{off-on}} \) and \( t_{\text{switch}} \) are respectively set as energy consumption and time for changing single server from hibernation or closed state to start state. The energy consumption and time for changing from start state into hibernation or closed state are lower. Meanwhile, both aspects can be ignored in order to facilitate analysis. It is assumed that \( T \) is calculation cycle of system energy consumption cost. Meanwhile, there are respectively \( x \) and \( y \) servers in working state and idle state in SRM within \( T \) time. \( z \) servers are changed from hibernation or closed state to start state. Then, system operation energy consumption is \( P_{\text{run}} \) and conservation energy consumption is \( P_{\text{switch}} \) within cycle \( T \).

\[ P_{\text{run}} = \left( P_{\text{busy}} \times (n+x) + P_{\text{idle}} \times y \right) \times T \]  

(7)

\[ P_{\text{switch}} = P_{\text{off-on}} \times t_{\text{switch}} \times z \]  

(8)

Total energy consumption cost of the system within cycle \( T \) is \( P \):

\[ P = P_{\text{run}} + P_{\text{switch}} \]  

(9)

3.1.3 Adaptive threshold

Threshold adjustment principle: if \( L \) value is bigger, it is obvious that there are fewer servers in working state for the cloud computing system, thereby extending the task waiting queue. The threshold should be reduced under the condition, thereby more servers can be in the working state for reducing length of task waiting queue. If \( P \) value is bigger, it is obvious that servers in SRM are frequently used, and the threshold should be increased under the condition in order to reduce the frequency of awakening servers in the SRM. Dynamic adjustment formula of threshold is shown as follows:

\[ \begin{aligned}
  a(t+1) &= \left[ a(t) \times \theta \right] \\
  a(1) &= N
\end{aligned} \]  

(10)

Wherein, \( a(i) \) (\( i=1, 2, 3, \cdots \)) is the threshold of the \( i \)th cycle. \( \theta \) is threshold regulatory factor, and the calculation formula is shown as follows:

\[ \theta = \frac{k - L}{k} \frac{L > L_{\text{fixed}}}{P_{\text{busy}} \cdot (n+x)} \frac{P_{\text{busy}} \cdot n}{P_{\text{busy}} \cdot n} \]  

(11)

Wherein, \( L_{\text{fixed}} \) and \( P_{\text{fixed}} \) are respective task separation threshold and energy consumption threshold.

3.2 Server scheduling

Because service life can be shortened due to frequent turn-on and turn-off of the server, each server in SRM should be used in a balanced mode in order to prolong service life of the servers. It is assumed that \( t_j \cdot Condition_i \) and \( Timer_i \) are respectively priority, state identification, and timer of server \( i \) in SRM. \( \tau_j = \infty \cdot Condition_i = 0 \cdot Timer_i = 0 \) for all servers in SRM at the initial time. After servers in SRM are turned off again, \( Condition_i = 1 \cdot Timer_i \) is set to be zero and starts timing, and \( \tau_i = Timer_i \).

When servers in the SRM are turned on again for processing tasks, \( Condition_i = 2 \cdot \) the timer stops timing, and \( \tau_i \) expression is shown as follows according to the above analysis:

\[ \tau_i = \begin{cases} 
  \infty, & \text{Condition} = 0 \\
  Timer_i, & \text{Condition} = 1 \\
  Timer_i, & \text{Condition} = 2 
\end{cases} \]  

(12)

Server scheduling principle: the state manager can firstly schedule \( \tau_i = \infty \) server for processing tasks in the initial stage. After all servers in SRM are turned on, the state manager can firstly schedule server with \( Condition_i = 1 \) and higher priority for processing task.
3.3 Algorithm realization

![Algorithm Flowchart]

Figure 2. Algorithm flowchart.

Figure 2 shows efficient and energy-saving DPS algorithm based on adaptive threshold. Figure 2 indicates that the algorithms proposed in the paper can be divided into the following four steps:

1. Initialization: the system running parameters can be initialized, such as server quantity in SMM, task separation threshold, energy consumption threshold, etc;
2. Task request processing: since cloud computing system is based on the principle of serving early arrival requests. After the task requests arrive in the system, the system should firstly determine whether the waiting queue is empty or not. If it is not empty, the task enters the waiting queue, otherwise the system can check whether there is idle server or not in SMM. If there is idle server, the task can be assigned to the idle server for processing, otherwise step (3) is started;
3. Adaptive threshold calculation: The system can calculate threshold according to the formula (10), and judge whether the current waiting queue length is larger than the threshold or not. If it is larger than the threshold, step (4) is started, otherwise the tasks can be continuously available in the waiting queue.
4. Scheduling of server: State manager can select the unoccupied server for task processing according to the server scheduling strategy in section 3.2, and related parameters are updated.

4 SIMULATION ANALYSIS

In the paper, a cloud computing system containing 100 servers is established under CloudSim 3.0 environment in order to verify the performance of the algorithm, wherein SMM and SRM contain respectively contain 75 servers and 25 servers. Table 1 shows setup of simulation parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>100</td>
</tr>
<tr>
<td>$n$</td>
<td>75</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>5</td>
</tr>
<tr>
<td>$\mu$</td>
<td>10</td>
</tr>
<tr>
<td>$P_{busy}$</td>
<td>100W</td>
</tr>
<tr>
<td>$P_{idle}$</td>
<td>70W</td>
</tr>
<tr>
<td>$P_{off \rightarrow on}$</td>
<td>110W</td>
</tr>
<tr>
<td>$t_{switch}$</td>
<td>5ms</td>
</tr>
<tr>
<td>$T$</td>
<td>80ms</td>
</tr>
<tr>
<td>$a(1)$</td>
<td>100 pieces</td>
</tr>
<tr>
<td>$L_{fixed}$</td>
<td>0.5k</td>
</tr>
<tr>
<td>$P_{fixed}$</td>
<td>$[n+0.5(N-n)] \times T$</td>
</tr>
</tbody>
</table>

In the section, performance of EEDPSAT algorithm, ETSP algorithm [10] and DPECO algorithm is compared from two aspects of average response time and total system energy consumption based on the above simulation parameters.

Figure 3 shows the changing condition of average response time with task quantity. Figure 3 shows that the average response time difference is not larger under the condition of smaller task quantity because working server quantity in the cloud computing system can meet the needs of task processing. However, the average response time of ETSP algorithm is better than that of EEDPSAT algorithm and DPECO algorithm with continuous increase of task quantity because all servers in cloud computing system are in the start state in ETSP algorithm. Therefore, task requests can be processed as fast as possible. Figure 3 shows that the average response time of DPECO algorithm is larger than that of EEDPSAT algorithm because static threshold strategy is adopted by DPECO algorithm for controlling turn-on of servers in SRM. It is difficult for the static threshold to effectively respond to the change of task quantity. Time of tasks in waiting queue is increased, and average response time is increased. However, EEDPSAT algorithm can dynamically adjust the threshold according to energy consumption cost and task separation quantity, thereby effectively shortening task waiting time and reducing average response time.
Figure 3. Task quantity vs. average response time.

Figure 4 shows the change condition of total system energy consumption with task quantity. It is obvious in Figure 4 that ETSP algorithm has the maximum system energy consumption because all servers are always in the turn-on state and wait for processing task request in ETSP algorithm. Idleness of servers still can consume a lot of electricity. Therefore, ETSP algorithm achieves the worst energy-saving effect. The system energy consumption of DPECO algorithm is lower than that of ETSP algorithm because static threshold is used by DPECO algorithm for controlling server quantity in the turn-on state, namely when the length of task waiting queue is longer than the threshold, servers in the SRM can be turned on, thereby effectively reducing quantity of idle servers and realizing reduction of energy consumption. The system energy consumption of EEDPSAT algorithm is the lowest because EEDPSAT algorithm can adaptively adjust the threshold according to the system operation status. The started server quantity can meet the demand for processing tasks, and server turn-on data in SRM can be better controlled, thereby reducing system energy consumption. In a word, the system energy consumption of EEDPSAT algorithm is the lowest among three algorithms.

5 CONCLUSION

In the paper, an efficient energy-saving DPS algorithm based on adaptive threshold is proposed aiming at problems in DPS algorithm based on static threshold. The algorithm not only can dynamically adjust threshold according to the energy consumption cost and task separation quantity for realizing system energy consumption reduction, but also can realize balanced use of reserved servers, thereby effectively prolonging the service life of servers.

REFERENCES