CHAPTER 5

FAULT AWARE POWER EFFICIENT SCHEDULING

A Cloud computing is now a trending way of computing tasks and more general these days. Cloud computing is adopted by many firms like Google, Amazon, Microsoft and many more for reliable and efficient computing. But as the cloud size increases with expand in number of datacenters and vigorously increasing consumption of power over a data center. Also with increasing request load over a server the computing load on server’s increases, leading high power consumption. So there is a need to balance the request load in such a manner to effectively improve the resource utilization, load with reducing request failure and power consumption.

Cloud computing has made it complicated with variable length request whose proportion may increase or decrease effecting the cloud. Recent surveys show that the power consumption of a datacenter increases linearly with increase in utilization due to request load over a datacenter. This results in high request failure and decreasing power efficiency of system. Resource allocation done without having knowledge of load and power efficiency of a datacenter will increase the power consumption of system and high request failure count. So to overcome these issues a fault and power aware resource allocation and scheduling algorithms are proposed to improve the power efficiency, failure count and average load over a datacenter. A proposed algorithm shows improved performance in term of average load and power efficiency as compared to existing algorithms for cloud infrastructure.
5.1. Approach 1: Power and Fault Aware Reliable Resource Allocation for Cloud Infrastructure

The problem with existing algorithm is that they used for simple task scheduling to improve resource utilization or power efficiency in cloud and manage quality of service of a datacenter. Existing algorithms also assumes cloud as non faulty in nature so do not takes fault occurring in the system for scheduling and only taken load over da datacenter, which is in sufficient to provide better QoS to the user. So to conquer these issues a power and fault aware resource allocation algorithm is proposed. Proposed algorithm utilizes linear power model or evaluation of power efficiency of datacenter. Failures over a datacenter occur randomly that may be due to network or storage failure. Proposed VM allocation algorithm aims to minimize the power consumption of the system and reduce request failure count. Proposed algorithm is based on fitness value which is evaluated using power efficiency and failure probability of datacenter.

Parameters to evaluate fitness value:
- \( PD_i \): \( i \)th Data center.
- \( PE_i \): power efficiency of \( i \)th host in a data center.
- \( U_i \): Current Utilization of \( i \)th host in a data center.
- \( FR_i \): Fault rate that is the number of request failed due to System failure over time \( t \).
- \( FP_i \): Failure probability over a Host \( i \).
- \( F_i \): Fitness value of \( i \)th host.

By Applying liner power utilization of \( PE_i \) can be calculated.

\[
PE_i = LineaPower \left( \frac{(P_{max} - P_{min}) \times U_i}{100} \right) \tag{5.1}
\]

Where \( P_{max} \) & \( P_{min} \) = maximum and minimum power consumed by \( PD_i \) respectively.

Utilization of Data Center can be calculated by

\[
U_i = \left( \frac{(Total\_MIPS - Allocated\_MIPS)}{Total\_MPIS} \right) \tag{5.2}
\]

Since faults over a data center are random in nature and follows Poisson distribution, which over a period of time \( t \) and \( t + \Delta T \) can be defined as:
FPi(t ≤ T ≤ t + ΔT | T > t) = \frac{\exp(-\lambda t) - \exp(\lambda (t+\Delta t))}{\exp(-\lambda t)} \quad (5.3)

FPi(t) = (1 - \exp(-\lambda t)) \quad (5.4)

Fi = PEi + FPi(t) \quad (5.5)

//Fitness value

As in above formula of Ui is calculated by getting total utilization from total MIPS allocated by data center PDi. Once calculate utilization of data centers then calculating power consumed by these data centers and using linear power efficiency formula as above. To get power efficiency of data centers as well to allocation resources for requests is done with below steps. On the other end we need to calculate the fault rate over a data center PDi, which depend on the number of request failed on a data center over a period of time ‘t’. Since fault is random in nature so the probability of failure can be found using poison distribution as shown in equation 5.3. Equation 5.3 defines the probability of failure at data center PDi. Base on the above defined parameters fitness value of each datacenter is calculated, as shown in equation 5.5 which is sum of power efficiency and probability of failure in fraction which range from 0 to 1.

5.1.1. Proposed algorithm

---

**Algorithm : PFARA**

1. PFARA (PD, Q_length)
2. **Input:** Datacenter List PD and Queue length Q_length
3. PD ← Host List
4. i ← No. of Data Centers
5. Q_length ← current queue size
6. PEi ← Power Efficiency of Host PDi
7. Fpi ← Failure Probability of Host PDi
8. If(Q_length = 0) then
9. Allocate_Resources (Req)

**Output:** All request scheduled.

---

Figure 5.1 Proposed PFARA algorithm initialization
Figure 5.1 & 5.2 shows the pseudo code of proposed trust and deadline aware ant colony algorithm. Figures show various phases of algorithm of initialization and evaluation of fitness value and final selection.

5.1.2. Experimental and Results

Proposed power and fault aware VM allocation algorithm is simulated using CloudSim 3.0 and power module package. CloudSim provides a benchmark for simulation of cloud platform and also provides Linear power model for simulation of power model. Proposed algorithm is being tested under various request count with 4 servers S1, S2, S3, S4 & S5. Linear power model directly depend on utilization of servers. Proposed algorithm is compared with basic DVFS (Dynamic voltage and frequency scaling) scheduling [80]. Compression of proposed algorithm is performed for 200, 400, 600, 800, 1200 and 1400 set of requests. These set of request contributes of various type of short, average and large requests sizes. System configuration taken into consideration is as follows:

```plaintext

Algorithm : Find fittest host
Input: Requests list r
1. Allocate_Resource( Req r)
2. Host_list= gethostlist();
3. Sorted_host = Sort_Fitness (Host_list)
4. for (Host h: Sorted_host)
5. Fi=PEi + FPi (t);
6. If( h.isSuitable() && h.fitness_Least() )
7. selected_host= h;
8. end for
9. if(selected_host != NULL) then
10. allocate( request, selected_host)
11. else
12. printf("cannot find suitable server")
Output: The server with minimum fitness value.
```

Figure 5.2 Proposed PFARA algorithm resource allocation
Table 5.1: Experimental parameters used for simulation environment

<table>
<thead>
<tr>
<th>Server</th>
<th>RAM(Mb)</th>
<th>MIPS</th>
<th>Storage (Gb)</th>
<th>core</th>
<th>PE</th>
<th>HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>2000</td>
<td>10000</td>
<td>100000</td>
<td>4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>S2</td>
<td>2000</td>
<td>10000</td>
<td>100000</td>
<td>6</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td>2000</td>
<td>10000</td>
<td>100000</td>
<td>6</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>S4</td>
<td>2000</td>
<td>10000</td>
<td>100000</td>
<td>6</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>S5</td>
<td>2000</td>
<td>10000</td>
<td>100000</td>
<td>6</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 5.3 Power consumption
Figure 5.3 shows the improvement in power consumption by proposed algorithm over DVFS. Figure 5.4 shows the improvement in number of request failed by proposed algorithm over DVFS in various test cases. Figure 5.5 shows the improvement in number of request completed by proposed algorithm over DVFS in various test cases. From the experiment it is shown that proposed algorithm performs better than DVFS in term of failed request, power efficiency and completed request count.
5.2. Approach 2: Trust and Deadline-Aware Scheduling Algorithm for Cloud Infrastructure Using Ant Colony Optimization

This paper proposed a trust and deadline aware algorithm that uses various parameters to evaluate the trust value for a host, on that trust value we have proposed and VM allocation policy to maximize the utilization of resources available in data center. Flow chart of proposed algorithm is below. As can see in flow chart we begin with task pool, here we look for task, if task pool is empty then do nothing but if there is some request in task pool for completion then we proceed to collect the information of each data center. Trust can be defined as an indirect reliability or a firm believe over a host based on its past performance parameters.

Trust is based on:

Start time: Time taken by host to initialize a virtual machine (VM).
Processing Speed: Total number of MIPS of a machine i.e. Number of processor * number of MIPS in each processor
Fault Rate: This can be defined as the total count of request failed over a period of time T
Utilization: this is the current utilization of that host in real time.
Power Efficiency: the ratio of the output power over the input power i.e. the percentage power consumed over a period of time.

For scheduling algorithm, we have proposed an ant colony based VM allocation algorithm which uses a fitness function based on above discussed trust value and deadline to find the fittest host among all.

Steps for proposed algorithm are as follows:
Step 1: Initialize datacenters and host
Step 2: Initialize search ants equal to number of hosts.
Step 3: Assign ants to search randomly and evaluate their fitnesses for a request on each host.
Step 4: Stop when all ants have arrived, otherwise wait for all ants for a fixed time.
Step 5: evaluate the trust value for each host and sort them in descending order.
Step 6: find the fittest host with highest trust value and can fulfill the task with deadline.
Step 7: if found, update pheromone value table with updated trust value that will be used for evaluation of fitness function for other requests.

Step 8: Assign bees to search randomly and evaluate their fitness and find new beast solutions.

Step 9: Stop, when no more requests.

Trust value ($T_i$): Trust value for host $i$.

$$T_i = \alpha_1 \times Initial_i + \alpha_2 \times PS_i + \alpha_3 \times \frac{1}{Fault_i} + \alpha_4 \times \frac{1}{Utilization_i} + \alpha_5 \times \frac{1}{PE_i}$$

(5.6)

$$\alpha_1 < 1, \alpha_2 < 1, \alpha_3 < 1, \alpha_4 < 1, \alpha_5 < 1$$

(5.7)

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1$$

(5.8)

where

Initial$_i$: initialization time of host $i$.

$$PS_i : PE_i \times MIPS_i$$

(5.9)

Fault$_i$: fault rate over host $i$.

Utilization$_i$: utilization on host $i$ at current point of time.

PF$_i$: Power efficiency of host $i$.

Fitness function $F(n)$:

$$F(n) = \text{Min} \ (T_i) \ \& \ D_j < \text{computation time} \ \ i=0\ldots n$$

(5.10)

$j$ is request id and computation time is the time to compute the request over host ‘$i$’. Let take $PE_i$(power efficiency) and $U_i$ (Utilization) of data Centers(i,e; $i=1,2,3,4\ldots\ldots n$). By Applying liner power utilization of $PE_i$ can be calculated.
\[ PE_i = LineaPower \left( \frac{(P_{\text{max}} - P_{\text{min}}) \times U_i}{100} \right) \]  

(5.11)

Where \( P_{\text{max}} \) & \( P_{\text{min}} \) = maximum and minimum power consumed by PD\(_i\) respectively.

\( U_i \) = Utilization of Data Center can be calculated by

\[ U_i = \left( \frac{(\text{Total}_{\text{MIPS}} - \text{Allocated}_{\text{MIPS}})}{\text{Total}_{\text{MIPS}}} \right) \]  

(5.12)

To get power efficiency of each data center first calculating utilization of PD\(_i\) then using Power Liner model to calculate power efficiency of that data center. The proposed algorithms Pseudo is below, this Pseudo code shows that request allocation based on power efficiency of data center minimize power loss and increase utilization of resource that implies, throughput of data center is increasing.

Pseudo code of TDARP (Trust and Deadline Aware Resource Allocation Policy) algorithm takes data centers list, queue length of task in task pool, Power Efficiency of data centers, as shown in pseudo code if task pool is not empty, then calculate the power efficiency on the basis of their utilization.

5.2.1. Proposed Algorithm

<table>
<thead>
<tr>
<th>Algorithm : TDARPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. TDARP (Host List PD and ( Q_{\text{length}} ))</td>
</tr>
<tr>
<td>\hspace{1cm} Input: Host List PD and Queue length ( Q_{\text{length}} )</td>
</tr>
<tr>
<td>2. ( PD \leftarrow ) Host List</td>
</tr>
<tr>
<td>3. ( i \leftarrow ) No. of Data Centers</td>
</tr>
<tr>
<td>4. ( Q_{\text{length}} \leftarrow ) current queue size</td>
</tr>
<tr>
<td>5. ( PE_i \leftarrow ) Power Efficiency of Host ( PD_i )</td>
</tr>
<tr>
<td>6. ( Fpi \leftarrow ) Failure Probability of Host ( PD_i )</td>
</tr>
<tr>
<td>7. ( \text{Initiali} \leftarrow ) Start time of Host ( PD_i )</td>
</tr>
<tr>
<td>8. If(( Q_{\text{length}} \neq 0 ))</td>
</tr>
<tr>
<td>9. \hspace{1cm} \text{Sent}<em>\text{Search}</em>\text{ant}(\text{Reqtype});</td>
</tr>
<tr>
<td>10. \hspace{1cm} \text{find Fittest host}(\text{Reqtype});</td>
</tr>
<tr>
<td>11. \hspace{1cm} \text{Update pheromone}();</td>
</tr>
<tr>
<td>12. \hspace{1cm} ( Q_{\text{length}} \leftarrow \vdash )</td>
</tr>
<tr>
<td>13. If(( Q_{\text{length}} &gt; 0 )) then goto step 1;</td>
</tr>
<tr>
<td>14. End</td>
</tr>
<tr>
<td>\hspace{1cm} Output: The server with minimum fitness value.</td>
</tr>
</tbody>
</table>

Figure 5.6. Proposed TDARPA algorithm (1)
Figure 5.7. Proposed TDARPA algorithm (2)

Figure 5.6 & 5.7 shows the pseudo code of proposed trust and deadline aware ant colony algorithm. Figures show various phases of algorithm of initialization and evaluation of fitness value and final selection.

5.2.2. Experiment and Results

Proposed power based trust and deadline aware allocation algorithm is simulated using CloudSim 3.0 and power module package. Linear power model is used for simulation of power model. Proposed algorithm is being tested under various request count with 4 servers S1, S2, S3 & S4. Linear power model directly depend on utilization of servers. Proposed algorithm is compared with basic DVFS (Dynamic voltage and frequency scaling) scheduling [80]. Compression of proposed algorithm is performed for 1000, 1500, 2000, 2500 and 3000 set of requests. System configuration taken into consideration is as follows:

---

**Algorithm**: Find fittest host

1. Find_Fittest host(Reqtype r)
   - **Input**: Request type r
2. Compute PEI for each host
3. Compute utilization for each host
4. Compute Ti for each host
5. Sort_decending(Ti)
6. Find host with high trust value and fit’s deadline
7. Return selected host

**Output**: The server with minimum fitness value.
Table 5.2: Experimental parameters used for simulation environment

<table>
<thead>
<tr>
<th>Server</th>
<th>RAM (Mb)</th>
<th>MIPS</th>
<th>Storage (Gb)</th>
<th>core</th>
<th>PE</th>
<th>HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>3000</td>
<td>2000</td>
<td>100000</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>S2</td>
<td>3000</td>
<td>2000</td>
<td>100000</td>
<td>6</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>S3</td>
<td>3000</td>
<td>2000</td>
<td>100000</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>S4</td>
<td>3000</td>
<td>2000</td>
<td>100000</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5.8 Comparison of request completed

Figure 5.8 and 5.9 shows the improvement in number of request completed and failed using proposed and ant colony based base algorith [81]. Figure also proves that the proposed algorithm completes more requests than existing algorithm. Figure 5.9 shows the improvement in power consumption when tested over 1000, 1500, 2000, 2500 and 3000 set of requests. figure 5.10 compares the power efficiency of proposed algorithm over existing algorithm over increasing request load. Proposed algorithm proves to consume less power as compared to existing algorithm.
In performance section, it is clear that TDARP is giving high performance as compare to previous proposed algorithm. The main of this algorithm in cloud computing is to complete the request as possible as minimum power and full utilization of resource, proposed algorithm shown that it can maximize throughput and minimize the requests failure count and computation power.

5.3. Conclusion

The main achievement of this study and work was to study the rich literature and solve the issue of resource allocation in fault aware cloud environment. The results obtained with our approach were very competitive with most of the well known algorithms in the literature and justified over the large collection of requests. Proposed resource allocation algorithm proves to provide better fault tolerance as compared to existing algorithm with least request failure, high request completion count, and Power consumption of system.