CHAPTER 4

Optimized PERMA-G based Workflow Scheduling
# CHAPTER 4

Chapter 4: Optimized PERMA-G based Workflow Scheduling

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>INTRODUCTION</td>
<td>87</td>
</tr>
<tr>
<td>4.2</td>
<td>PARAMETERS FOR EVALUATING TIME AND COST</td>
<td>90</td>
</tr>
<tr>
<td>4.3</td>
<td>ESTIMATING THE THRESHOLD VALUE</td>
<td>92</td>
</tr>
<tr>
<td>4.4</td>
<td>OPTIMIZED PERMA-G SCHEDULING ALGORITHM</td>
<td>96</td>
</tr>
<tr>
<td>4.5</td>
<td>RESULTS AND DISCUSSION</td>
<td>97</td>
</tr>
<tr>
<td>4.6</td>
<td>CONCLUSION</td>
<td>105</td>
</tr>
</tbody>
</table>
4. OPTIMIZED PERMA-G BASED WORKFLOW SCHEDULING

4.1 INTRODUCTION:

The optimization techniques based on execution time and power utilized is developed and PERMA is developed with definite planning heuristics in multi-core based Grid systems. QoS and limitations of power both can be addressed only when the significant importance is given to power saving in the scheduling strategy. Both the QoS factors are addressed when the tasks are executed in less time, use less power when both the cost and overall execution time are considered as main factors of workflow scheduling. The task type and environment to execute will lead to the different variations in the workflow of different tasks. The distribution on the grid becomes complex with the dependency on the task and environment to execute. One of the major problems is scheduling is a priority inversion problem. The priority inversion problem occurs when the higher priority task is indirectly pre-empted by the lower priority task. The following method considers three active tasks in the scheduling, each at three different task priorities which is shown in figure 4.1. When the system begins execution. Task T1 is released and executes immediately since there is no other higher priority tasks executing. Shortly, after it starts, it acquires the resource R1. At time t=1.5, task T3 is released and preempts task T1 since it’s of higher priority. At time t=3, task T2, a medium priority task is released, but doesn’t
execute because higher priority task T3 is still executing. Shortly afterward, however, task T3 attempts to acquire the resource R1, but cannot since thread T1 (a lower priority thread) still owns it. This allows task T2 to execute in its place, which effectively violates the priority-order execution of the system, resulting in priority inversion. In this situation, task T3 will continue to block on resource R1, for an unbounded and an unknown amount of time, until task T2 blocks (or terminates).

Figure 4.1: Priority inversion Problem
With priority inheritance, the system that implements resource locking checks to see if a lower priority task already owns a resource when a task attempts to acquire it. If one does, that owning task priority is temporarily increased to match that of the higher priority task attempting to acquire the resource. As a result, the resource holder (once blocked at lower priority) will execute, release the resource, and then be restored to its original priority level.

By the above example, priority inheritance would effectively boost task T1’s priority to equal that of task T3, where task T2 would continue to block, allowing T1 to release its resource sooner. In Figure 4.2, it can be observed how this allows task T3 to resume sooner, without the unbounded latency caused by task T2's unknown

Figure 4.2: Solution for the Priority Inversion
execution time. Once the R1 is released by task T1, its priority is restored to its original value and the system executes according to normal priority-based rules. In this way the priority inversion problem is solved in scheduling workflows.

As an effort in addressing these problems, the tasks are assigned to the VM by workflow scheduling based on the type and the dependency of application which consists of the parameters like time and cost of the grid. The threshold value is determined and then both the cost and time parameters are estimated. Consequently, the time and cost parameters are optimized by the scheduler in analogous to the threshold value. The Minimum Completion Time (MCT) and Minimum Cost (MC) are the two parameters which are used to determine the threshold value.

4.2 PARAMETERS FOR EVALUATING TIME AND COST:

**Minimum Cost (MC):** The grid resources are efficiently used to approximate the least potential time required to complete the given task. Every task is assigned an individual virtual machine in order to reduce the execution time of the task. This way of reducing the execution time will definitely raise the expense because of the investment related to the maintenance of the individual virtual machines for each task. On the contrary, the cost can be reduced with the increase in the execution time
**Minimum Completion Time (MCT):** The least potential cost is approximated. The cost can be reduced by decreasing the number of VMs at any specific level to carry out the task. The tasks that are dependent on each other assigned to a single virtual machine that is already in effect holding the application to realize and the unusable virtual machines are removed to reduce the cost. As more tasks of an application are assigned to a single machine, it takes more time to complete each task which increases the overall execution time. The level of decrease in the threshold of the time and cost is more than the decrease in the threshold of MCT with the rise in the cost. The value of MCT and MC are used to take the verdict. For example, if the cost is increasing, the related tasks are assigned to the already in effect a virtual machine rather than increasing the quantity of virtual machines. Hence, there will be a reduction in the cost, irrespective of the time. The threshold is estimated when the execution time is increasing more for the reason of the inadequate or few VMs in the claim to decrease the execution time. The execution time may be decreased with the addition of more VMs, however the cost is increased.

The harmonizing of such conflicting cases depending on the threshold need to be performed by the scheduler is important in order to trade-off the cost and time by maintaining both the total execution time and overall expense to be low. The threshold can be estimated
using the parameter computed from the least potential execution time compared to the least potential cost.

**Cost Minimization:** The cost of executing an application can be decreased by allocating the task to a virtual machine which is implementing the related or similar tasks at that instance of time. Maintaining a single queue for all the similar tasks on a single virtual machine reduces the cost as the number of virtual machines being used becomes less.

**Make span Minimization:** The total time to complete the execution in the workflow is reduced with the allotment of individual VM to every task upon the demand. The cost and the power utilization increases more and more with the increase in the number of tasks which leads to upsurge in the number of virtual machines.

The workflow can be divided into different intensities depending on the organization and analogous task properties [43]. According to yang Qu et al, the particular task level must be more than the level of its ancestors.

### 4.3 ESTIMATING THE THRESHOLD VALUE:

**Threshold Value:** Approximation of time and cost can calculate in cooperation job and make span reduction. MC and MCT are used by PERMA framework to calculate the Tmax and Tmin, Cmax and Cmin and consequently, threshold of cost and time can be computed using the following equations:
\[ \Delta T = T_{\text{max}} - T_{\text{min}} \]  
\[ \Delta C = C_{\text{max}} - C_{\text{min}} \]

Threshold for Time

\[ \partial_T = \frac{\Delta T}{\Delta C} \]

Where,

\( \Delta T \) is the difference between mid extreme and least total execution time.

\( \Delta C \) is the difference between extreme and least total cost.

Cost Threshold

\[ \partial_C = \frac{1}{\partial_T} \]

Taking into account the counter relationship between them, the execution time or the cost turns out to be extreme here, where the corresponding threshold can be decreased by increasing the other threshold.

The PERMA-G framework is utilized by the OPSA-G algorithm to decrease the power utilization by the workflow scheduling with the approximation of the task completion time. The task \( T \) that is accessible is mapped to the resource \( R \) that is accessible using the above algorithm. Consequently, the execution time of the actual task \( T \)
decides the association of particular sub task \( t \) to resource \( r \) in \( R \). The workflow of the tasks determines the way the scheduling need to be carried out for the complete list of tasks to available resources. The scheduling process is optimized by estimating the threshold based on the difference between the time and cost when the scheduling is not performed. There will a noticeable decrease in the execution time of the task with the least possible number of VMs causing decrease in power utilization according this approach.

With the workflow indicating the dependency of tasks as input, the scheduling of the task to the resources is carried out (line 1-5). The threshold of cost \( \partial C \) and time \( \partial T \) are discovered from line 6 using Eq. 4.1 and Eq. 4.2.

Consequently, the control goes into a loop (line-7) until the scheduling of the available task to the resource is not performed. The ET (Early completion Time) of each distinct job on the respective resource available is calculated for each looping phase in this procedure. The two parameters called Maximum ET and Minimum ET for each individual job are attained in order to schedule on the available source. The task is estimated to complete within the least potential time depending on the threshold.

1. Min ET\( (t, r) \) for each resource

\[
R, \leftarrow \min_{r \in \text{availResources}} \text{ET}(t, r)
\]
2. Min-Min: obtain a task with Min ET \((t, r)\) over tasks and for Max-Min: get a task with Max ET \((t, r)\) over tasks

\[
T \leftarrow \min_{t \in \text{availTask}} \text{ET}(t, r)
\]

If the number of tasks is more than 1 and the total threshold is less than the threshold value of time \((\partial t < T_t)\), (if there is request to lessen the completion time of task then providing an extra VM can be guaranteed) then new tasks are developed in support of the virtual machine that comprise the software such that RC \(\leq \partial C\) (line 16-20). In the other case, the resources that are associated with the tasks are rewarded. Lastly, the task that is scheduled is detached and updated for the assignment of sources (line 21-26).
4.4 Optimized PERMA-G Scheduling Algorithm:

Input: T task

Output:

Begin

1: Schedule T to R

2: for each tЄT do

3: avail t- get ready task whose parent task have been completed, where R is the resource of T

4: schedule avail t

5: end for

6 Calculate ∂T and ∂C

7: while avail TЄR not scheduled

8: for each avail tЄT do avail R for T

9: for each avail rЄR

10: compute ET(t,r)

11: end for

12: if r has no tasks

13: avail r for T with Min ET (t, r)

14: for each rЄR do

15: avail t of Min ET (t,r)
16: \textbf{if} \ Tt > \overline{t}t \text{ and } r > 1
17: \text{avail} t \text{ to new} r \text{ such that} RC \leq \overline{\partial}C
18: \textbf{else}
19: \text{avail} t \text{ of Max ET} (t, r)
20: \text{avail} t \text{ to} r \text{ contained task}
21: \textbf{end for}
22: \text{schedule} t \epsilon TT \text{ to avail} r \epsilon R
23: \text{remove} T \text{ from} availTasks
24: \text{update} ET(R)
25: \textbf{end for}
26: \textbf{end while}
27: \textbf{End}

\textbf{4.5 RESULTS AND DISCUSSION:}

The simulation is carried out using the Cloudsim [44] tool. The cloudsim toolkit is used for scheduling of workflows and hence cloudsim is support grid simulation. cloudsim is an extension of gridsim. The architecture of the cloudsim contains the gridsim as a layer which is shown in figure 4.3.
In order to test the performance of the proposed algorithm, OSAP uses the PERMA-G prototype by means of execution time. The code in the simulator is enhanced to implement OSAP. The evaluation of the some conventional scheduled procedures like Non Power Aware (NPA), Dynamic Voltage and Frequency Scaling (DVFS), First come and First serve (FCFS) and Round Robin (RR) is made based on the
execution time and cost with respect to the number of VMs, MIPS, power consumption and type of hosts and is shown in Figure 4.4. Optimal scheduling is performed in the proposed algorithm to achieve minimum execution time and minimum cost. The relative analysis of algorithms to one another is given in Table 4.1 and Table 4.2. The comparison is made by considering the random work flow of the tasks.

The parameters used in the simulation to evaluate the performance are given in Table 4.3.

PERMA-G operates in a multicore environment and the communication delay is negligible. The task execution in PERMA-G is done by estimating the resource capability and the task requirements. There is no need of task division and assignment to the other resources. The appropriate resource is allocated by estimating the task requirements. The figure 4.4 explains about the performance improvement of the VMs when compared with normal execution. Both tightly coupled and loosely coupled applications can use this technique.

The evaluation of the proposed algorithm and the conventional algorithm in terms of mean power consumption is shown in Figure 4.5 and Figure 4.6 and it can be observed that the proposed algorithm performs better. The way the static power of the host used efficiently to its maximum point so as to run a number of MIPS on four types of hosts with four VMs is shown in Figure 4.6. Different methods are tested with fixed number of VMs and a varying number of hosts in
terms of power consumption and are shown in Figure 4.7, 4.8, and 4.9.

Table 4.1 The Simulation Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time</td>
<td>80100</td>
</tr>
<tr>
<td>Host</td>
<td>10-50</td>
</tr>
<tr>
<td>VMs</td>
<td>50-300</td>
</tr>
<tr>
<td>RAM (MB)</td>
<td>512-24576</td>
</tr>
<tr>
<td>Bandwidth (b/sec)</td>
<td>1000-1000000</td>
</tr>
<tr>
<td>Scheduling</td>
<td>FCFS, DVFS, RR</td>
</tr>
</tbody>
</table>

Table 4.2 Virtual Machines Setup

<table>
<thead>
<tr>
<th>Resource/VM</th>
<th>VM Type1</th>
<th>VM Type2</th>
<th>VM Type3</th>
<th>VM Type4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiPs</td>
<td>750</td>
<td>1000</td>
<td>1500</td>
<td>2000</td>
</tr>
<tr>
<td>Cores</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAM(MB)</td>
<td>512</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Bandwidth(b/sec)</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Storage(MB)</td>
<td>25000</td>
<td>25000</td>
<td>25000</td>
<td>25000</td>
</tr>
</tbody>
</table>
Table 4.3 Host Setup

<table>
<thead>
<tr>
<th>Resource/Host</th>
<th>Host type0</th>
<th>Host type1</th>
<th>Host type2</th>
<th>Host type3</th>
<th>Host type4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiPs</td>
<td>1500</td>
<td>2000</td>
<td>2500</td>
<td>4200</td>
<td>6000</td>
</tr>
<tr>
<td>Cores</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RAM(GB)</td>
<td>24576</td>
<td>24576</td>
<td>24576</td>
<td>24576</td>
<td>24576</td>
</tr>
<tr>
<td>Bandwidth(bps)</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
<td>100000</td>
</tr>
<tr>
<td>Storage(MB)</td>
<td>100000000</td>
<td>100000000</td>
<td>1000000000</td>
<td>100000000</td>
<td>100000000</td>
</tr>
</tbody>
</table>

Table 4.4 VC allocation based on Time and Cost for workflow without workflow scheduling

<table>
<thead>
<tr>
<th>Task(Length)</th>
<th>VCs</th>
<th>Time</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1(10), T3(20), T4(30),</td>
<td>M1</td>
<td>721s</td>
<td>721s</td>
</tr>
<tr>
<td>T2(40), T3(50), T4(60), T5(70)</td>
<td>M1={T2}, M2={T3}, M3={T4,T5}</td>
<td>M1=694s, M2=721s, M3=3959s+5643s</td>
<td>9602s</td>
</tr>
<tr>
<td>T6(80), T7(90), T8(100)</td>
<td>M1={T6}, M2={T7,T8}</td>
<td>S1=12356s, S2=25197s+31217s</td>
<td>56414s</td>
</tr>
</tbody>
</table>

Table 4.5 VC allocation based on Time and Cost for workflow with workflow scheduling

<table>
<thead>
<tr>
<th>Task(Length)</th>
<th>VCs</th>
<th>Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1(10), T3(20), T4(30),</td>
<td>M1</td>
<td>721s</td>
<td>721s</td>
</tr>
<tr>
<td>T2(40), T3(50), T4(60), T5(70)</td>
<td>M1={T2,T5}, M2={T3}, M3={T4}</td>
<td>M1=694s+5643s, M2=721s, M3=3959s</td>
<td>6337s</td>
</tr>
<tr>
<td>T6(80), T7(90), T8(100)</td>
<td>M1={T6}, M2={T7}</td>
<td>M1=12356+25197s, M2=18499s</td>
<td>37553s</td>
</tr>
</tbody>
</table>
Figure 4.4 Overall execution time based on number of tasks

Figure 4.5 Power utilization based on the number of VMs
Figure 4.6 Power utilization based on Host with VMs

Figure 4.7 Power utilization performance
Figure 4.8 Power utilization performance for the 350 Hosts with 800 VMs

Figure 4.9 Power utilization performance for the 150 Hosts with 800 VMs
4.6 CONCLUSION

The major motivation behind this study is to propose a powerful straightforward prototype that can calculate power and formulates an equation for the decrease of power utilization figured in contrast to the execution time over the Grid. This chapter proposes PERMA-G framework alongside work process planning so as to reduce utilization of energy without affecting the execution time. PERMA-G in combination with scheduling of work process has abundantly exhibited the probability of reducing power utilization without compromising on the loss of execution time in the Grid. Subsequently, PERMA-G framework empowers an improvement of the energy by calculating the execution time of 66.57% over the Grid. In this manner, this chapter endeavors to solidly make the adequacy of the system of PERMA-G for different Grid-based situations including multi-core processors.