Ptolemy Project Description

In this assignment, you are asked to develop a DVFS algorithm using StateCharts (the Modal Model tool) in Ptolemy. The time space you will work on is divided into discrete intervals. For each interval, your module will get CPU and memory utilization values based on a default CPU frequency. You should adjust the frequency based on the given utilization values. As a result of the new frequency, both the power consumption of the CPU and the interval length might change. Your algorithm will be evaluated based on two criteria:

1. The total execution time (the sum of each interval length)
2. The total energy consumed

Implementation Details

You are given two sample files, “CPU.txt” and “Memory.txt”. In each file, there is a single number between 0-100, representing CPU/Memory utilization. You do not need to perform error-checking, and you can assume that the number of lines in the CPU and Memory files is the same. Additionally, the last entries of both these files are 0. When you see this value, you need to make sure that your model terminates.

There are 10 designated frequency values, from 300MHz to 3GHz and the frequencies are equally separated, i.e. assuming that the frequency 1 is the slowest, we can represent the frequencies as $300MHz \times i$, where $1 \leq i \leq 10$.

The default frequency is always the fastest (3GHz) and the default interval length is 1ms.

The power consumption profile for all the frequencies is given by Table 1.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Idle Power Consumption (mW)</th>
<th>Active Power Consumption (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300MHz</td>
<td>84</td>
<td>219</td>
</tr>
<tr>
<td>600MHz</td>
<td>91</td>
<td>246</td>
</tr>
<tr>
<td>900MHz</td>
<td>98</td>
<td>278</td>
</tr>
<tr>
<td>1.2GHz</td>
<td>107</td>
<td>316</td>
</tr>
<tr>
<td>1.5GHz</td>
<td>117</td>
<td>359</td>
</tr>
<tr>
<td>1.8GHz</td>
<td>128</td>
<td>405</td>
</tr>
<tr>
<td>2.1GHz</td>
<td>140</td>
<td>455</td>
</tr>
<tr>
<td>2.4GHz</td>
<td>153</td>
<td>508</td>
</tr>
<tr>
<td>2.7GHz</td>
<td>166</td>
<td>565</td>
</tr>
<tr>
<td>3GHz</td>
<td>180</td>
<td>625</td>
</tr>
</tbody>
</table>

Table 1. Frequency vs. Power Consumption
A. Performance Categorization

We classify a given interval into 3 different categories based on memory utilization:

1. Memory utilization ≤ 50
2. 50 < Memory utilization ≤ 75
3. Memory utilization > 75

If you slow down the frequency of the CPU by 1 level, your CPU and memory utilization values change. How much these utilization values will change depends on the category your current utilization values belong to:

1. If you are in the first category, slowing down the frequency by 1 level results in 4% increase in CPU utilization and 1% increase in memory utilization.
2. If you are in the second category, slowing down the frequency by 1 level results in 3% increase in CPU utilization and 1% increase in memory utilization.
3. If you are in the third category, slowing down the frequency by 1 level results in 2% increase in CPU utilization and 1% increase in memory utilization.

Also note that you need to make sure that the utilization values should not exceed 100%. In other words, you can slow down the frequency further after hitting 100% but the utilization value should not increase further.

With this categorization, we are classifying an interval as either CPU-based or memory-based. The main logic is that the performance hit should be greater if you apply DVFS in a CPU-based interval. Here, what you need to do is “to decide when to slow down the frequency”. You should consider the two goals described previously when designing your DVFS module. Also, note that lower power consumption does not always mean lower energy consumption.

B. Interval Calculations

Interval Length

After, computing the new CPU and memory utilization values and the new frequency, you should compute the new interval length. The equation you should use for this computation is:

\[
\text{interval}_{\text{new}} = \text{interval}_{\text{def}} \times \left(\frac{\text{frequency}_{\text{def}}}{\text{frequency}_{\text{new}}}\right)^{a_{\text{CPU}}} \times (a_{\text{CPU}} \times \left(\frac{\text{CPU utilization}_{\text{def}}}{\text{CPU utilization}_{\text{new}}}\right)^{a_{\text{CPU}/2}} + a_{\text{memory}} \times \left(\frac{\text{memory utilization}_{\text{def}}}{\text{memory utilization}_{\text{new}}}\right)^{a_{\text{memory}/2}})
\]

where:

- \(\text{interval}_{\text{new}}\) and \(\text{interval}_{\text{def}}\) are the new and default interval length values respectively.
- \(\text{frequency}_{\text{new}}\) and \(\text{frequency}_{\text{def}}\) are the new and default frequencies respectively.
- \(\text{CPU utilization}_{\text{new}}\) and \(\text{CPU utilization}_{\text{def}}\) are the new and default CPU utilization values respectively.
– $memory\ utilization_{new}$ and $memory\ utilization_{def}$ are the new and default memory utilization values respectively.
– $\alpha_{CPU}$ and $\alpha_{memory}$ are CPU and memory based coefficients. For these variables, you should use:
  o $\alpha_{CPU} = 0.8$ for category 1.
  o $\alpha_{CPU} = 0.5$ for category 2.
  o $\alpha_{CPU} = 0.2$ for category 3.
  o $\alpha_{memory} = 1 - \alpha_{CPU}$ for all categories.

Note: $\alpha_{memory}$ equation has been changed.

These coefficient values represent how sensitive the execution time is. A smaller coefficient means that the execution time depends less on the corresponding variable.

Energy
You should also compute the energy consumption of every interval:

$$Energy = interval_{new} \times power$$

and

$$power = Idle\ Power_i + Active\ Power_i \times CPU\ utilization_{new}$$

where $i$ is the index of $frequency_{new}$ and you can obtain the power numbers from Table 1.

C. Output
You should plot the variables below with a $SequencePlotter$, i.e. you need to collect this data for each interval and feed them into a $SequencePlotter$ actor:

1. The new interval length
2. The energy consumption
3. The new frequency

Also, when your model finishes execution, you should display:

1. Total execution duration
2. Total energy consumption

Submission
Your submission should include 2 files:

1. A brief report (at most 2 pages). It should include how your algorithm works, three sequence plots and the total energy consumption and total execution duration. This report should be in pdf format.
2. The XML file of your Ptolemy implementation.
The deadline is Feb 8th at 9:30 AM (prior to the start of class).

**Grading**

The grading depends on the total execution duration and total energy consumption values your model obtains. This project consists of 10% of the overall class project and should be done individually.

**Appendix A: Ptolemy Implementation**

When you are implementing your DVFS module with Ptolemy, you should choose the option “New Modal Model” from the “New” menu. This tool enables you to create statechart-like models where you can add other computation models *inside the states*. These computation models placed inside the states are called “Refinements”. Refinements allow you to execute the models of other domains, such as SDF, PN, SR, etc. This ability is necessary when reading from files and outputting values to the screen. For more information about the “Refinements” please read the Ptolemy documentation.