

STORM

Simulation TOol for Real-time Multiprocessor scheduling

User Guide V3.2 – May 2009

This work is part of the project PHERMA supported by ANR grants ANR-06-ARFU-003.

1. Introduction

The increasing complexity of the hardware multiprocessor architectures as well as of the real-time applications they support makes very difficult even impossible to apply the theoretical real-time multiprocessor scheduling results currently available. Thus, so as to be able to evaluate and compare real-time multiprocessor scheduling strategies on their schedulability performance as well as energy efficiency, a simulation approach has been preferred and led us to develop an open and flexible multiprocessor scheduling simulation and evaluation platform called STORM.

STORM stands for “Simulation TOol for Real time Multiprocessor scheduling” and Figure 1 gives an overview of the STORM platform architecture. For the time being, engineering efforts on STORM have concerned its simulator component since it is the retaining element of the platform. For a given “problem i.e. a software application that has to run on a (multiprocessor) hardware architecture, this simulator is able to “play its execution” over a specified time interval while taking into account the requirements of tasks, the characteristics and functioning conditions of hardware components and the scheduling rules.

As shown in Figure 2, the specification of the architectures and scheduling policy to be simulated is done via an XML file. The result of simulation is a set of execution tracks that are either made directly observable through diagrams, or recorded into files for a subsequent computer-aided analysis. All these results allow the user to analyze the behaviour of the system (tasks, processors, timing, performances, etc.).

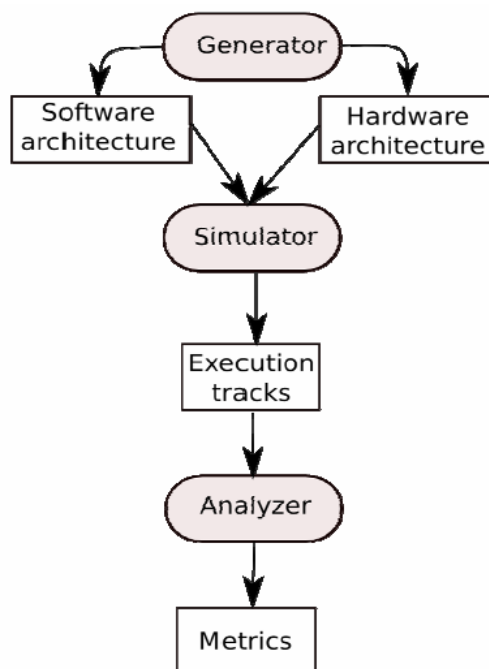


Figure 1. The STORM platform.

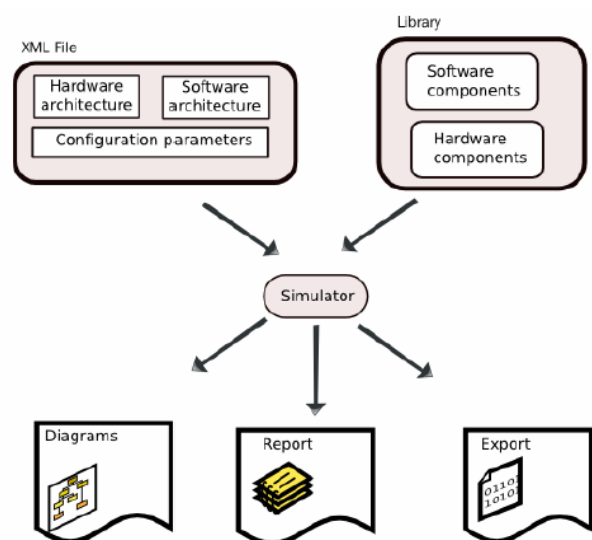


Figure 2. The STORM simulator.

For the installation procedure of the STORM software, see the “Getting Started” document on the web site: <http://storm.rts-software.org/>.

2. The XML file as the specification of a simulation

This section describes the XML file used as input with the "exec" command (see §3). This file is organized hierarchically around some tags as shown in Figure 3. Attributes are associated to some tags. Moreover it is possible for the developer of new scheduler components to introduce and inform new dynamic attributes.

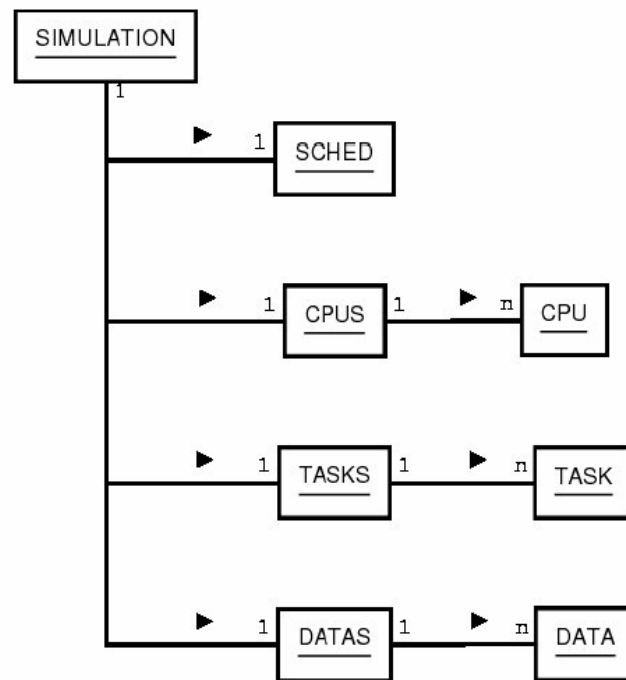


Figure 3. The present hierarchy of tags in the XML input file.

• SIMULATION

The SIMULATION root tag of the XML file is associated with two attributes.

The "duration" attribute gives the upper bound of the simulation interval in number of slots.

The “precision” attribute is a floating point value which gives the absolute duration of one slot in milliseconds. By default, the precision is equal to 1.0, and thus the duration of a slot is one millisecond.

Example:

```

<SIMULATION duration="75" precision="1.0">
. . .
</SIMULATION>

```

- **SCHED**

The SCHED tag enables to specify the scheduler used in the simulation. Its unique attribute "className" must indicate the name of the scheduler Java class. A lot of classes are provided in the library of schedulers associated with STORM. See appendix 3 for the list of presently available scheduler components.

Example:

```
<SCHED className="storm.Schedulers.FP_P_Scheduler">
</SCHED>
```

- **CPUS**

The CPUS tag (for "CPU Set") begins the section relative to the specification of the set of processors that compose the hardware architecture.

- **CPU**

The CPU tag enables to define an instance of processor. The attributes that must accompany this tag are:

- **className**: the name of the Java class that models the type of the considered processor. It is a reference to a processor type in the library. See appendix 4 for the list of presently available processor types;
- **name**: the name given to this instance of processor. It is used as a reference in some outputs (for example, the title given to the windows representing temporal diagrams of this processor instance);
- **id**: the numeric identifier that the simulation kernel will use to identify this processor instance. This identifier must be unique in the XML file.

Example:

```
<CPU className="storm.Processors.CT11MPCore" name="CPU B" id="1">
</CPU>
```

- **TASKS**

The TASKS tag (for "TASK Set") begins the section relative to the specification of the set of tasks that compose the software architecture.

- **TASK**

The TASK tag enables to define an instance of task. The attributes that must accompany this tag are:

- **className**: the name of the Java class that models the type of the considered task. See appendix 2 for the list of presently available task types;
- **name**: the name given to this instance of task. It is used as a reference in some outputs (for example, the title given to the windows representing temporal diagrams of this task instance);
- **id**: the numeric identifier that the simulation kernel will use to identify this task instance. This identifier must be unique in the XML file.

The following other attributes are optional. They have default values and some of them are conditional to the kind of used scheduler:

- **BCET**: the "Best Case Execution Time" of this task instance in milliseconds as a floating point value (default value = 0.0);
- **WCET**: the "Worst Case Execution Time" of this task instance in milliseconds as a floating point value (default value = BCET);
- **activationDate**: the date of its first activation in slots relatively to the beginning of the simulation interval (default value = 0);
- **period**: its activation period in milliseconds as a floating point value (default value = 0.0);
- **deadline**: its deadline (critical delay) (default value = period);
- **priority**: its integer fixed priority, mandatory for fixed-priority schedulers (default value = 0);
- **powerDeviation**: percentage of processor electrical power consumed by this task instance execution (default value = 100);

Example:

```
<TASK className="storm.Tasks.PTask_NAM" name="PTASK T1" id="2"
period="10" activationDate="0" WCET="5" priority="1">
</TASK>
```

• DATAS

The DATAS tag (for "DATA Set") begins the section relative to the specification of the set of data that are exchanged between the tasks of the software architecture.

• DATA

The DATA tag allows to define an instance of a data which is sent from a task instance towards another task instance. The attributes that must accompany this tag are:

- **source**: the id number of the task instance which produces the data;
- **destination**: the id number of the task instance which consumes the data.

Two optional attributes can be added:

- **rate**: the production rate of this data, i.e. the number of times it must be produced before to be consumed (default value = 1);
- **size**: the size of the data in bytes (default value = 0).

Example:

```
<DATA source="3" destination="2" rate="1" size="10">
</DATA>
```

3. Graphic User Interface

The user interface includes a permanent window named "console". It is in this dialog window that the user gives his commands after the prompt "##>". The main window is used for displaying the results that are produced by certain commands. Hereafter all the commands are organized by theme.

Remarks:

- A command has always two forms: the complete form (example: “exec”) and an associated short form (example: “ex”). The short form is given hereafter in brackets;
- Several commands on the same line must be separated with “;”. They will give rise to a sequential execution.

3.1. Console commands

3.1.1. Simulation execution

- **exec** (ex) <FileName.xml>

This command launches a simulation based on the specifications given in the XML file < FileName.xml>.

Example:

```
##> exec test-edf1.xml
```

3.1.2. Configuration information display

- **context** (ctx)

The command gives the list of the identifiers and names of the entities that take part to the simulation. These identifiers (“entity_Id” are used in some other commands as explained in next sections) and names are those specified in the XML file.

- **windowsinfo** (wi)

The command gives the list of the identifiers and the names of the various windows that are displayed. These identifiers (“window_Id”) are required in some other commands, for example the “print” one.

- **taskdata** (td)

The command displays the graph of data dependencies between task instances (see §3.3 for more information).

- **explorer** (exp)

The command opens a window that gives the tree of the entities that are involved in the current simulation. As shown in Figure 4, by selecting an entity, it is possible to view its detailed description (static and dynamic attributes).

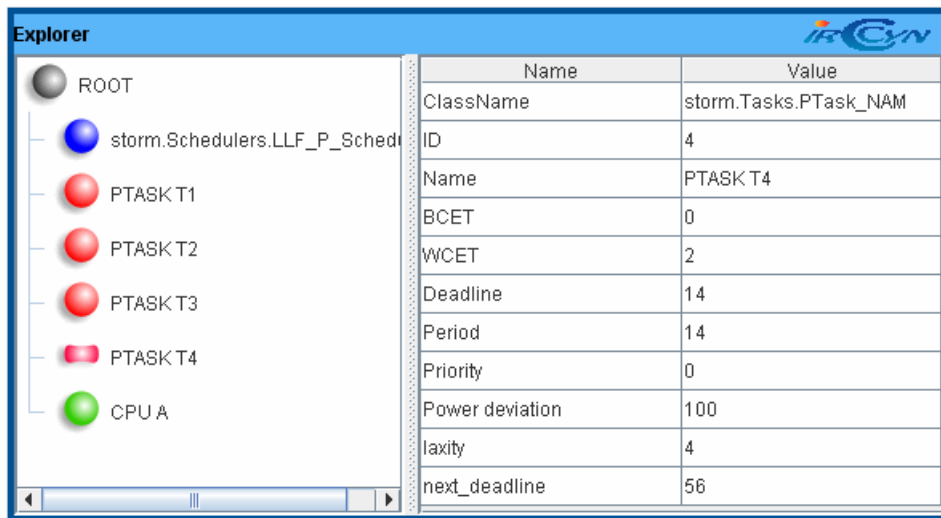


Figure 4. An example of the “explorer” command.

3.1.3. Simulation graph display

- **plotall** (pa)

The command displays the execution sequences built by the simulation. For each task and each processor, a window shows its Gantt diagram (see §3.2 for more information).

- **plotpower** (pp) <cpu_Id>

The command plots a graph showing the power consumption for the processor <cpu_Id>. The value of <cpu_Id> is given in the XML file or is known with the “context” or “explorer” commands (see §3.4 for more information).

- **plotdvfs** (pd) <cpu_Id>

The command plots a graph showing the behaviour of the DVFS system for the processor <cpu_Id>. The value of <cpu_Id> is given in the XML file or is known with the “context” or “explorer” commands.

- **plotload** (pl) <cpu_Id>

The command plots a graph showing the load of the processor <cpu_Id>. The value of <cpu_Id> is given in the XML file or is known with the “context” or “explorer” commands.

- **plotret** (pt) <task_Id>

The command plots a graph showing the progress of the RET (“Remaining Execution Time”) for the task <task_Id>. The value of <task_Id> is given in the XML file or is known with the “context” or “explorer” commands.

- **gantt** (gtt) <item_Id>

The command displays the Gantt diagram for the task or processor <item_Id>. The value of <item_Id> is given in the XML file or is known with the “context” or “explorer” commands.

- **close** (cl) <window_Id>

The command closes the window <window_Id>. The value of <window_Id> is known with the “windowsinfo” command.

- **closeall** (cla)

The command closes all the windows except, of course, the console window.

- **setpos** (sp) <window_Id> <time_slot>

- **setpos** (sp) <window_Id> <time_slot> <zoom_value>

The command changes the origin of the graph <window_Id> to the slot <time_slot>. Optionally the display is zoomed (in or out) depending on the value <zoom_value> (values: 0|1|2|3). The value of <window_Id> is known with the “windowsinfo” command.

- **setzoom** (sz) <window_Id> <slot_1> <slot_2>

- **setzoom** (sz) <window_Id>

The command displays the graph <window_Id> between <slot_1> and <slot_2>. If these two values are missing, the display returns to its original form. The value to use for <window_Id> is known with the “windowsinfo” command.

Shortcuts for the management of the graphs

- For the active window:

<ctrl> <q>	:closes the window
<shift> <end>	:resets the time axis to the zero date
<shift> <→>	:moves right along the time axis
<shift> <←>	:moves left along the time axis
<shift> <↑>	:zooms in the time axis
<shift> <↓>	:zooms out the time axis

- For all the windows:

<shift> <home>	:resets all the time axis to the zero date
<shift> <tab>	:goes to the next window and makes it active

3.1.4. Specific computations

All the following commands give their results in the console window.

- **clear** (cls)

The command erases the content of the console window.

- **showmess** (sm) <slot>

The command shows the kernel messages that have been processed at date <slot>.

- **showdvfs** (sd) <slot_1> <slot_2>

The command shows the DVFS operations that have been achieved between the dates <slot_1> and <slot_2>.

- **showmemory** (mem)

The command shows information about the (available and used) memory of the JVM on which STORM is running.

- **calcpower** (cp) <cpu_Id>

The command computes the electrical consumption of the processor <cpu_Id> over the entire simulation interval. The value of <cpu_Id> is given in the XML file or is known with the “context” or “explorer” commands.

Example:

```
##> calcpower 0
##> CPU Power consumption for CPU B
0,06 J
duration 75 ms
0,85 W
##>
```

Shortcuts for the management of the console window

<↑> :returns the previous command in the history buffer
<↓> :returns the next command in the history buffer

3.1.5. File management commands

- **print** (pr) < window_Id > < FileName.jpg >

The command saves the graphical view of the window < window_Id > in the file < FileName.jpg > in jpeg format. The value of <window_Id> is known with the “windowsinfo” command.

Example:

```
##> print 0 /tmp/cpuA.jpg
```

- **report** (pr) < spec_file> < FileName.ext >

The command gives rise to the automatic creation of the report file <FileName.ext> according to the information contained in the file < spec_file>. To be documented.

3.1.6. Miscellaneous

- **help** (?)
The command displays all the available commands and shortcuts in the console window.
- **exit** (xt)
The command exits the STORM simulator.
- **version** (vr)
The command gives the version number of the currently used STORM software.

Shortcuts for the management of the GUI

- <F12> : saves the position of the displayed windows
- <F1> : restores the windows in the lastly saved position
- <F2> : cascades the windows
- <F3> : tiles the windows
- <F4> : cascades the window by group
- <F5> : shows / hides the explorer window

3.2. Gantt diagrams for tasks and processors

The “**plotall**” command displays in several windows the execution sequences that result from the last simulation.

- For each task, a window shows its Gantt diagram over an interval beginning at date 0 and ending at date 50 (default values). Figure 5 illustrates such a task diagram. The title of the window refers to the name given to the task in the XML file (cf. attribute “name”).

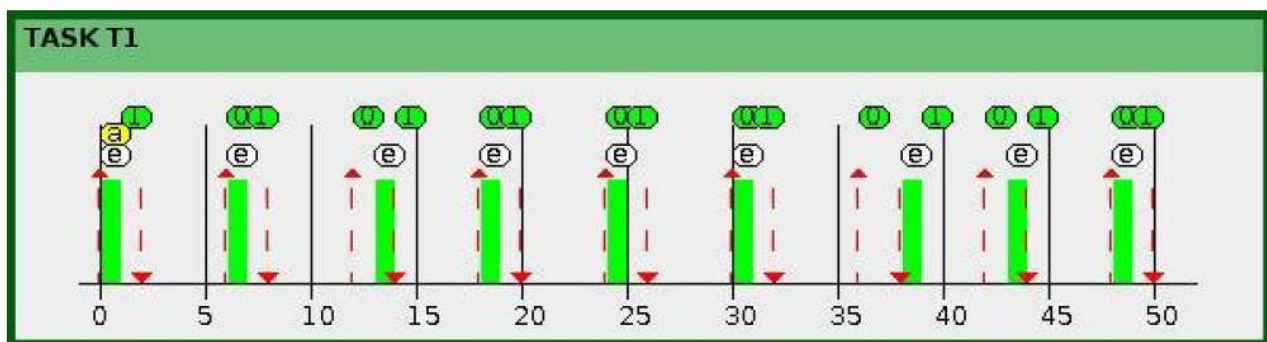


Figure 5. Gantt diagram for TASK T1.

- For each processor, a window shows its allocation to the tasks over an interval beginning at date 0 and ending at date 50 (default values). Figure 6 illustrates such a processor diagram. The title of the window refers to the name given to the processor in the XML file (cf. attribute “name”).

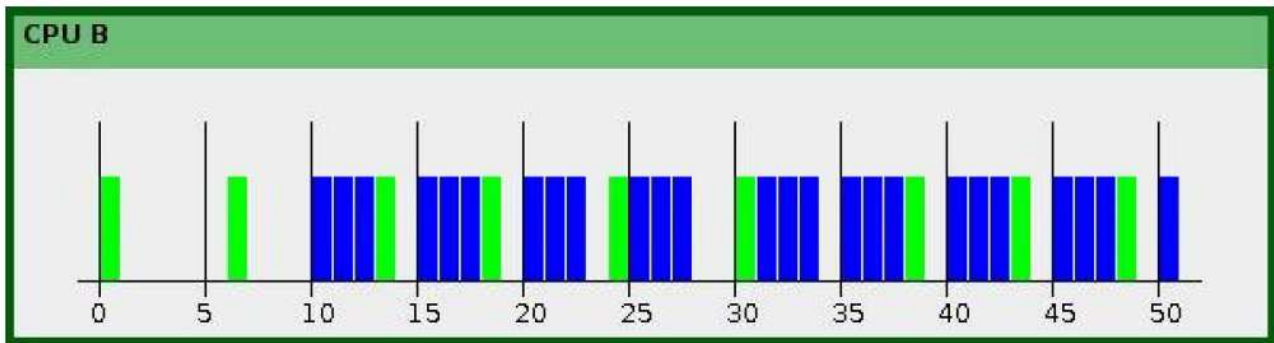


Figure 5. Gantt diagram for CPU B.

- These windows can be moved and re-arranged in the main window (by clicking on the upper bar of the window and then moving it).
- The main information relative to a task appears in a contextual window by clicking on its area. Figure 7 shows the result of such operation carried out on the diagram of a processor.

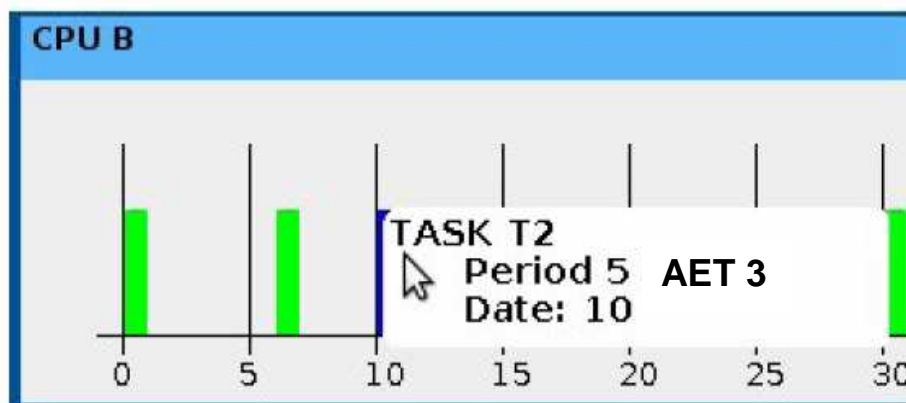

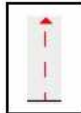


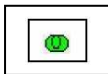


Figure 7. Contextual information.

- On the Gantt diagram of a task, icons are used to identify special instants:
 - The first activation of a task: 
 - The release of a job (for a periodic task): 
 - The completion of a job: 
 - The deadline of a job: 
 - The occurrence of a task blocking or unblocking event: 

The number in the circle indicates the status of the block counter. This information is linked to the internal task states managed by the simulation kernel. This information is useful for debug purposes.

3.3. Diagrams for task dependencies

The “**taskdata**” command displays the graph of data dependencies between task instances. Figure 8 shows an example. The tasks linked by a red bar are data dependent. The task on top is the data producer while the one under is the data consumer. The consumer task must wait for the end of the associated producer task. When there is more than one link, a “AND” behaviour is assumed: the consumer task must wait for the execution of all the producer tasks. Each link can also be characterized by the size of the exchanged data (attribute “size” in the XML file) and the number of producer executions required in order to unblock the consuming task (attribute “rate” in the XML file).

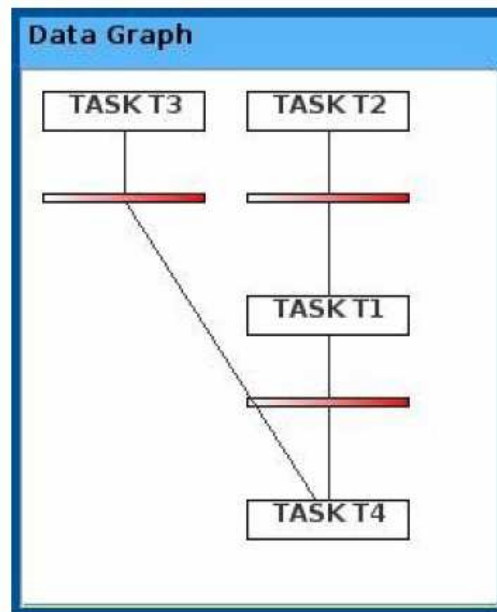


Figure 8. A software architecture with task dependencies.

3.4. Diagrams for CPU power

The “**plotpower**” command plots the electrical power consumption for a processor. Figure 9 shows it for the processor named “CPU B” (the characteristics of this processor type are: 1 W in running mode, 0.2 W in idle mode).

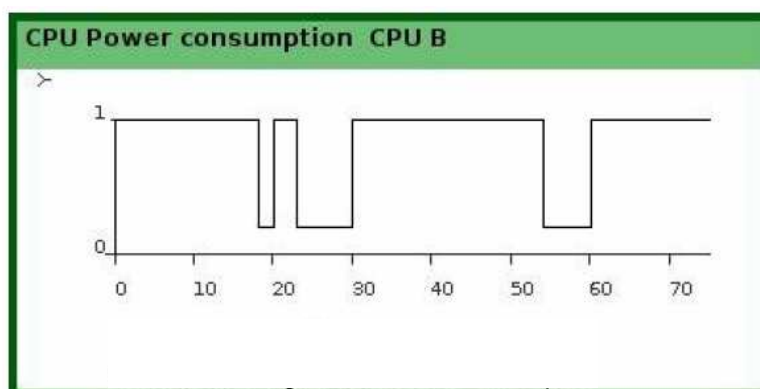


Figure 9. Power consumption for CPU B.

4. File example

This is a test file with a hardware architecture composed of 2 processors and a software architecture composed of 3 periodic independent tasks. Scheduler implements a fixed priority full preemptive scheduling algorithm.

Task	Initial phasing	Period	WCET	Priority
T1	0	10	5	1
T2	2	9	3	5
T3	4	6	2	10

<Test-FPP5.xml>

<!-- FP_P Fixed Priority Preemptive-->

<!-- comments -->

<SIMULATION duration="50">

<SCHED className="storm.Schedulers.FP_P_Scheduler"></SCHED>

<CPUS>

<CPU className="storm.Processors.CT11MPCore" name ="CPU A" id="11" ></CPU>

<CPU className="storm.Processors.CT11MPCore" name ="CPU B" id="12" ></CPU>

</CPUS>

<TASKS>

<TASK className="storm.Tasks.PTask_NAM" name ="PTASK T1" id="1" period="10" activationDate="0" WCET="5" priority="1"></TASK>

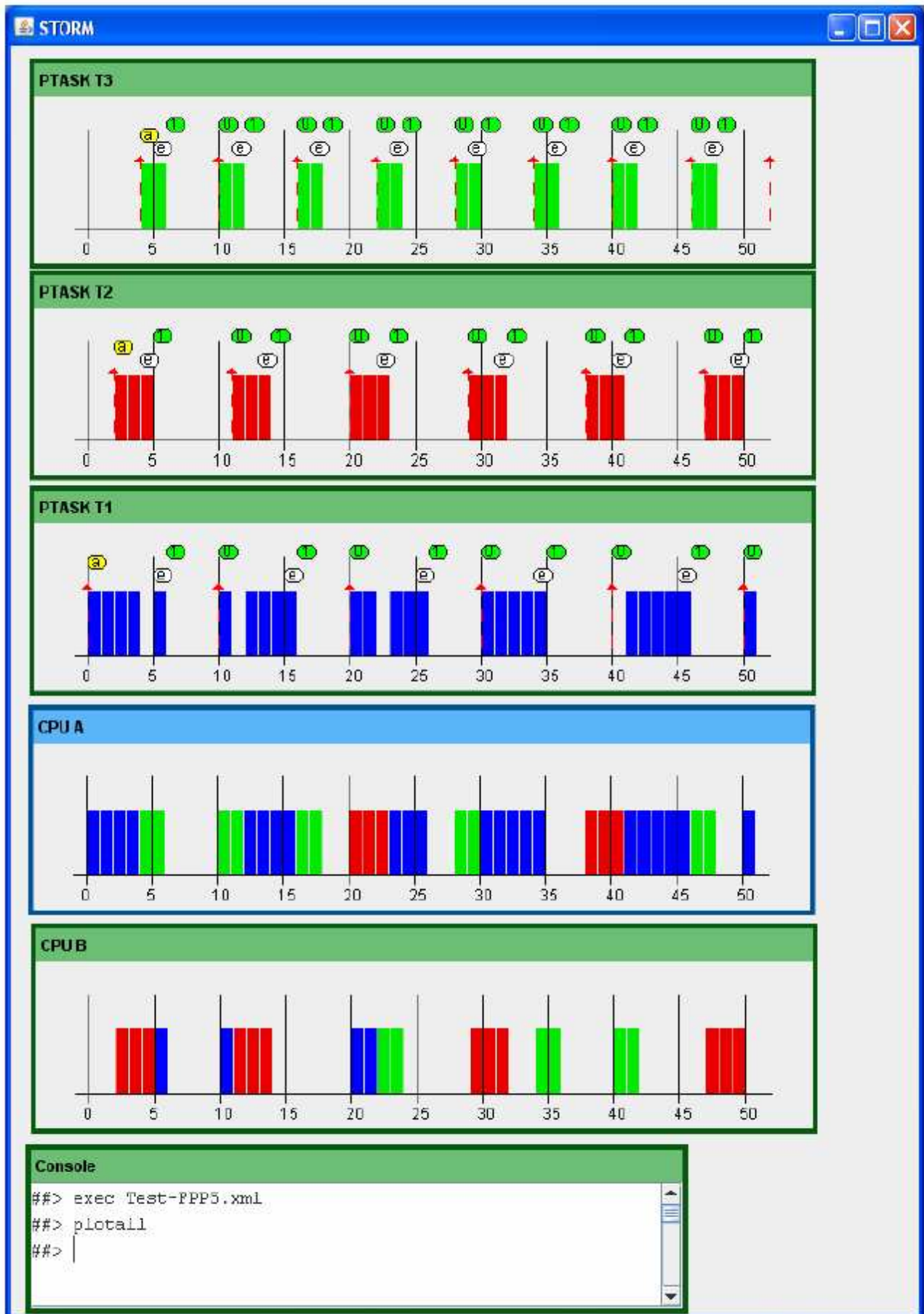
<TASK className="storm.Tasks.PTask_NAM" name ="PTASK T2" id="2" period="9" activationDate="2" WCET="3" priority="5"></TASK>

<TASK className="storm.Tasks.PTask_NAM" name ="PTASK T3" id="3" period="6" activationDate="4" WCET="2" priority="10"></TASK>

</TASKS>

</SIMULATION>

Hereafter the screenshot resulting from the two commands “exec Test-FPP5.xml” and “plotall” (followed by a manual rearrangement of the windows) is given.



Appendix 1 : STORM commands

Simulation execution

- **exec** (ex) <FileName.xml>

Configuration information display

- **context** (ctx)
- **windowsinfo** (wi)
- **taskdata** (td)
- **explorer** (exp)

Result window commands

- **plotall** (pa)
- **plotpower** (pp) <cpu_Id>
- **plotdvfs** (pd) <cpu_Id>
- **plotload** (pl) <cpu_Id>
- **plotret** (pt) <task_Id>
- **gantt** (gtt) <item_Id>
- **close** (cl) <window_Id>
- **closeall** (cla)
- **setpos** (sp) <window_Id> <time_slot>
- **setzoom** (sz) <window_Id> <slot_1> <slot_2>
- **setzoom** (sz) <window_Id>

Shortcuts for the management of the graphs

- For the active window:

- | | | |
|---------|-------|--|
| <ctrl> | <q> | :closes the window |
| <shift> | <end> | :resets the time axis to the zero date |
| <shift> | <→> | :moves right along the time axis |
| <shift> | <←> | :moves left along the time axis |
| <shift> | <↑> | :zooms in the time axis |
| <shift> | <↓> | :zooms out the time axis |

- For all the windows:

- | | | |
|---------|--------|--|
| <shift> | <home> | :resets all the time axis to the zero date |
| <shift> | <tab> | :goes to the next window and makes it active |

Specific computations

- **clear** (cls)
- **showmess** (sm) <slot>
- **showdvfs** (sd) <slot_1> <slot_2>
- **showmemory** (mem)
- **calcpower** (cp) <cpu_Id>

Shortcuts for the management of the console window

- <↑> :returns the previous command in the history buffer
<↓> :returns the next command in the history buffer

File management commands

- **print** (pr) < diagram_Id > < FileName.jpg >
- **report** (pr) < spec_file> < FileName.ext > To be documented.

Miscellaneous

- **help** (?)
- **exit** (xt)
- **version** (vr)

Shortcuts for the management of the GUI

- <F12> : saves the position of the displayed windows
<F1> : restores the windows in the lastly saved position
<F2> : cascades the windows
<F3> : tiles the windows
<F4> : cascades the window by group
<F5> : shows / hides the explorer window

Appendix 2 : List of task types

The list of task types that can be referenced in the attribute "className" is:

- **storm.Tasks.PTask_NAM**

Periodic Task, No Activation Memory, no abort on missed deadline

- **storm.Tasks.PTask_NAM_A**

Periodic Task, No Activation Memory, with Abort on missed deadline

- **storm.Tasks.PTask_WAM**

Periodic Task, With Activation Memory, no abort on missed deadline

- **storm.Tasks.PTask_WAM_A**

Periodic , With Activation Memory, with Abort on missed deadline

- **storm.Tasks.ATask**

Aperiodic Task

Appendix 3 : List of schedulers

The list of schedulers that can be referenced in the attribute "className" is:

- **storm.Schedulers.FP_P_Scheduler.java**
Fixed Priority full Preemptive
- **storm.Schedulers.FP_NP_Scheduler.java**
Fixed Priority Non Preemptive
- **storm.Schedulers.FP_M_Scheduler.java**
Fixed Priority Mixed preemptive
- **storm.Schedulers.FP_P_FIFO_Scheduler.java**
Fixed Priority full Preemptive with FIFO management for the same priority tasks
- **storm.Schedulers.FP_M_FIFO_Scheduler.java**
Fixed Priority Mixed preemptive with FIFO management for the same priority tasks
- **storm.Schedulers.EDF_P_Scheduler.java**
Earliest Deadline First full Preemptive
- **storm.Schedulers.EDF_NP_Scheduler.java**
Earliest Deadline First Non Preemptive
- **storm.Schedulers.RR_Scheduler.java**
Round Robin

Appendix 4: List of processor types

The list of processor type that can be referenced in the attribute "className" is:

- **storm.Processors.CT11MPCore**

- simple processor without DVFS and two power states "running" at 1 W and "idle" at 0.2 W