CADOUR Erwann

1st year of master's degree in Computer Science

2019-2020

Université de Bretagne Occidentale

TAL REPORT

-

Implementation of an energy-oriented scheduling algorithm in Cheddar

Lead teacher: Frank Singhoff

Contents:

1- Introduction:

The objective of this project was to implement the ED-H scheduling algorithm in Cheddar. To complete this objective, I had two scientific documents related to this subject at my disposal [1,2].

This project was divided in two different parts. The first part consisted of the reading and the analysis of a scientific paper and a thesis about scheduling in Real-Time Energy Harvesting systems. The objective of this part was to understand the ED-H algorithm to be able to implement it in Cheddar later. The second part consisted of the implementation of the ED-H algorithm in Cheddar using the Ada programing language.

In the first part of this report, I will present the ED-H scheduling algorithm and the RTEH model used by this scheduler. Following will be an ED-H scheduling example will be presented.

Then I will talk about the implementation of the ED-H algorithm in Cheddar. After that I will present the same example as prior, this time using Cheddar.

Finally, I will describe the test files used for scheduling in Cheddar and conclude about this project.

2- ED-H presentation:

The ED-H scheduler uses the RTEH model and was inspired by the EDF scheduler. It is used to schedule sets of jobs in RTEH systems by using the RTEH model and considers the notion of energy. For instance, the EDF scheduler is optimal for most systems but not for RTEH ones since it does not consider the energy required to the execution of a job and the arrival of future ones. Therefore ED-H uses the bases of EDF and considers what EDF does not to schedule RTEH systems.

The Real Time Energy Harvesting model (RTEH) aims at reducing the maintenance needed on batteries and extending their durability by harvesting energy from the environment of a system. This harvested energy can then be stored in batteries and used to power the system. A RTEH system is composed of a processor, a storage unit, an energy harvester which needs an energy source. The processor of this system can process jobs.

A job in the RTEH model:

- A job is a four-tuple T (r, c, e, d):
    - r: is the release time of a job
    - c: is the capacity of a job
    - e: is the maximum energy consumption of a job
    - d: is the deadline of a job
- Every job is discharging, this means that the energy stored at t(i+1) cannot be higher than the energy stored at t(i).

The energy production model:

- The energy produced is not controllable
- There exists an instantaneous charging rate: Pp
- Energy can be produced while a job is executed
- The value of the source power is not constant but can be predicted for the near future

The energy storage model:

- The energy storage unit has a capacity: C
- The energy available a time t is represented by: E(t)
- The capacity can be inferior to some job's energy consumption
- The energy storage unit is fully charged at time t (0)
- The energy stored does not leak over time and can be used at any moment

With ED-H and the RTEH model, there exists two types of starvation:

- Time starvation:
    - a job is not completed and has reached his deadline
    - there is still energy available in store
- Energy starvation:
    - a job is not completed and has reached his deadline
    - there is not enough energy available therefore preventing the job completion

The ED-H scheduler follows 5 rules:

- Rule n°1: The order of priority is the same as EDF.
- Rule n°2: The processor must be idle if there are no ready tasks.
- Rule n°3: The processor must be idle if
    - Condition 1: There is no energy left in the storage unit.
    - Condition 2: There exists a risk of energy starvation.
- Rule n°4: The processor must be busy if
    - Condition 1: The storage unit is full.
    - Condition 2: There exists a risk of time starvation.
- Rule n°5: The processor is either busy or idle, if there are ready tasks, no risk of time starvation and the storage unit is neither empty nor full.

The processor can be in two different states during the scheduling:

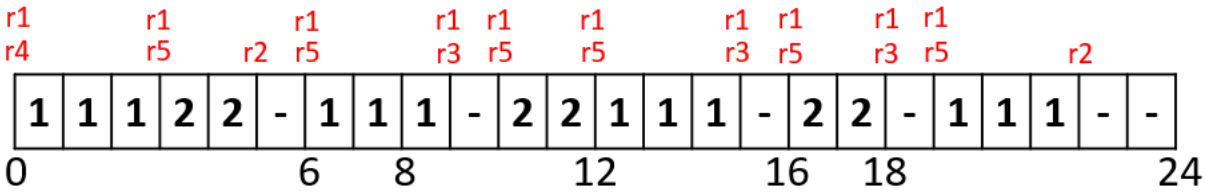
- Idle state: The processor does not compute jobs either because none are ready or there is a risk of energy starvation. The processor may also be idle without any of those requirements met.
- Busy state: The processor is computing jobs either because the storage unit is full or there is a risk of time starvation. The processor may also be busy without any of those requirements met.

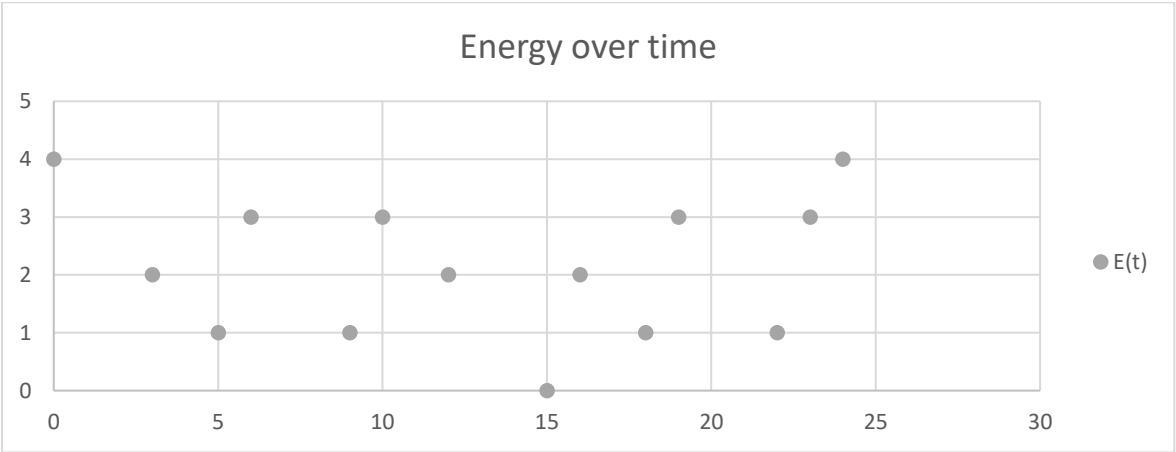
3- ED-H scheduling example:

Here the description of a task set from the thesis [2]:

- Tasks: a task is represented as follows: T(r,c,d,e)
  - T1(0,3,6,8)
  - T2(0,2,8,5)
- Initial energy E (0) = 4
- Capacity C = 4
- Rechargeable power Pp = 2

Scheduling of the task set with ED-H and evolution of the energy over time:





Explanation of the scheduling process:

At time t (0), the battery is full and both tasks are ready. According to rule n°1, T1 is elected. Since the battery is full, rule n°4 is verified, the processor will be busy.

At time t (3), the energy level is 2. According to rule n°1, T2 is elected, it is the only ready task in the system at that time. Since the battery is neither full nor empty and there is enough energy to run T2, rule n°5 is verified, the processor will be busy.

At time t (5), the energy level is 1. There are no tasks ready in the system. Rule n°2 is verified, the processor will be idle.

At time t (6), the energy level is 3. According to rule n°1, T1 is elected. The battery being neither full nor empty and the energy available enough to run T1, rule n°5 is verified. The processor will be busy.

At time t (9), the energy level is 1. According to rule n°1, T2 is elected. T2 needs at least 2 units of energy to run, there is only 1 available. There exists a risk of energy starvation, rule n°3 is verified. The processor will be idle.

The tasks set is scheduled following those 5 rules until time = 24.

4- Implementation of the ED-H algorithm in Cheddar:

To implement the ED-H algorithm in Cheddar, different modifications had to be made.

The battery:

First, the concept of battery had to be defined. In the RTEH model, a battery is composed of:

- A capacity (C)
- A rechargeable power (Pp)
- An initial energy (E (0))
- An Emax

In Cheddar, tasks, processors, and other objects needed are defined in Ada classes. Therefore, a class for the battery was created. This class is composed of all the elements listed above, a battery name and a CPU name. The battery's name can be used to differentiate batteries in case there is more than one in the system i.e. the system possesses a set of batteries. The CPU name is compared to the processor's name to ensure that the right battery is used.

All those attributes are filled in the battery's xml tag in an xmlv3 file used to describe a system and job set to schedule with Cheddar.

The energy:

To schedule systems with the ED-H algorithm, the notion of energy had to be implemented.

Thanks to the battery class defined earlier, useful information is known. However, to properly use the notion of energy within the algorithm one more information is required. It is the energy consumption of a task. The task class was therefore modified to add the energy consumption as a new attribute.

All the required information needed to implement and manage the energy in the algorithm now known, tests and operations on the energy were implemented in the code using the different values of the battery's attributes and the energy consumption of a task.

The state of the processor:

The way the ED-H scheduling algorithm works, its different rules decide of the state of the processing unit. To do so, a Boolean was added to represent the two states of the processor that were described earlier. The two states are represented as:

- True: the processor is idle
- False: the processor is busy

By the end of the process checking all 5 rules, the processor state is tested to determine the evolution of the energy level in the system and if a task will be run.

Implementation of rule 1 and 2:

With the concepts of battery and energy defined the rules were ready to be added in the code. The first two rules are rule 1 and 2:

- Rule n°1: The order of priority is the same as EDF.
- Rule n°2: The processor must be idle if there are no ready tasks.

Both those rules were already present completely or partly in the EDF algorithm code, therefore few modifications had to be made.

For instance, rule n°1 being the exact same as the one in EDF, nothing was changed.

However, rule n°2 brought a few modifications to the code. The variable updated by the test of the rule was changed.

Implementation of rule 3, 4 and 5:

The two first rules completed, rules 3, 4 and 5 were next:

- Rule n°3: The processor must be idle if
    - Condition 1: There is no energy left in the storage unit.
    - Condition 2: There exists a risk of energy starvation.
- Rule n°4: The processor must be busy if
    - Condition 1: The storage unit is full.
    - Condition 2: There exists a risk of time starvation.
- Rule n°5: The processor is either busy or idle, if there are ready tasks, no risk of time starvation and the storage unit is neither empty nor full.

To implement those three rules in the code, all their specific conditions were added, and the processor state updated depending on the verified conditions of each rule.

Rule n°3 and n°4 both test the energy storage level, rule n°3 verifying if it is empty and rule n°4 that if it is full. Those two rules also verify the risk of one type of starvation each.

Rule n°5 completes the energy storage level tests by verifying if it is neither empty nor full. It also verifies that there exist no risks of any kind of starvation.

Time starvation and Energy starvation:

For rule n°3, n°4 and n°5 to work properly, both possible risks of starvation in the system had to be known. To do so, each time the election procedure was called, the risk of time starvation and the risk of energy starvation were calculated.

- Energy starvation, also known as slack energy:
    - The slack energy of each task is calculated
    - The slack energy of the task set is the minimum of the slack energy of all tasks verifying wake up time and deadline conditions
    - The slack energy is then compared to the value of Emax, if Emax is superior to the slack energy then we consider the slack energy to be equal to 0.
    - If the value of the slack energy is equal to 0, then one of the conditions of rule n°3 is verified. There exists a risk of energy starvation, therefore the processor must be idle.
    - If the value is different from 0, then there is no risk of energy starvation. Note that the value cannot be inferior to 0.
- Time starvation, also known as slack time:
    - The slack time of each task is calculated
    - The slack time of the task set is the minimum of the slack time of all tasks verifying wake up time and deadline conditions
    - If the value of the slack time of the task set is equal to 0, then one of the conditions of rule n°4 is verified. There is a risk of time starvation, therefore the processor must be busy.
    - If the value is different from 0, then there is no risk of time starvation. Note that the value cannot be inferior to 0.

Both of starvation risks are tested in:

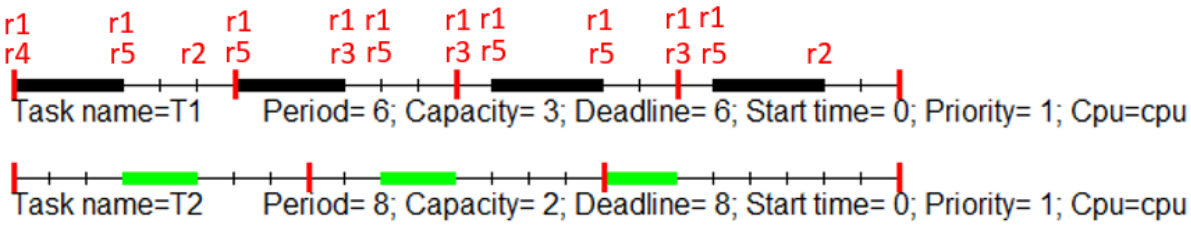
- Time starvation:
    - Rule n°3 and n°5
- Energy starvation:
    - Rule n°4 and n°5

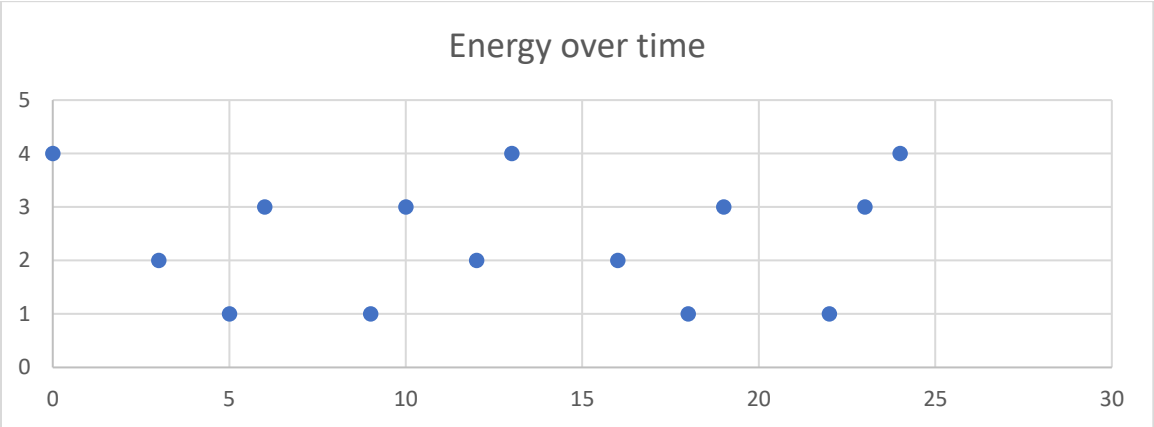
5- ED-H scheduling example with Cheddar:

Here the description of the same task set used for our first example [2]:

- Tasks: a task is represented as follows: T(r,c,d,e)
  - T1(0,3,6,8)
  - T2(0,2,8,5)
- Initial energy E (0) = 4
- Capacity C = 4
- Rechargeable power Pp = 2

Scheduling of the task set with ED-H and Cheddar:





Explanation of the scheduling process:

Since the task set used in Cheddar is the same as in the first example, the scheduling process is the same over the 24-time units except at time t (12).

For both examples, at time t (12), the energy level is 2. But the scheduling of Cheddar differs than the one made on “paper” at this specific time unit.

With Cheddar, at time t (12):

- According to rule n°1, T1 is elected.
- There is not enough energy available to execute the task, according to the implemented rule n°3, the processor will be idle.

On “paper”, at time t (12):

- According to rule n°1, T1 is elected.
- The battery being neither full nor empty, rule n°5 is verified, the processor will be busy.

This difference is due to the way ED-H was implemented in Cheddar; it can be corrected.

6- ED-H Test file presentation:

To schedule task sets with Cheddar, all the information of a set must be described in an xmlv3 file. It is no different with task sets to be scheduled using the ED-H algorithm.

A task is described in an xml tag as follows:

```
<periodic_task id="id_4">
 <object_type>TASK_OBJECT_TYPE</object_type>
 <name>T1</name>
 <task_type>PERIODIC_TYPE</task_type>
 <cpu_name>cpu</cpu_name>
 <address_space_name>ea</address_space_name>
 <capacity>3</capacity>
 <energy_consumption>8</energy_consumption>
 <deadline>6</deadline>
 <start_time>0</start_time>
 <priority>1</priority>
 <blocking_time>0</blocking_time>
 <policy>SCHED_FIFO</policy>
 <text_memory_size>0</text_memory_size>
 <text_memory_start_address>0</text_memory_start_address>
 <stack_memory_size>0</stack_memory_size>
 <criticality>0</criticality>
 <context_switch_overhead>0</context_switch_overhead>
 <cfg_relocatable>FALSE</cfg_relocatable>
 <mils_confidentiality_level>UNCLASSIFIED</mils_confidentiality_level>
 <mils_integrity_level>LOW</mils_integrity_level>
 <mils_component>SLS</mils_component>
 <mils_task>APPLICATION</mils_task>
 <mils_compliant>FALSE</mils_compliant>
 <access_memory_number>0</access_memory_number>
 <maximum_number_of_memory_request_per_job>0</maximum_number_of_memory_request_per_job>
 <period>6</period>
 <jitter>0</jitter>
 <every>0</every>
</periodic_task>
```

A task has a lot of different attributes, the most important ones with ED-H are:

- The CPU name: cpu_name
- The task name: name
- The capacity: capacity
- The energy consumption: energy_consumption, specific to ED-H tasks
- The deadline: deadline
- The start time: start_time

These are the attributes of a task used in the ED-H algorithm to schedule the task set with Cheddar.

All the information of a battery is filled in a specific xml tag in those files as well:

```
<battery id="100">
 <object_type>BATTERY_OBJECT_TYPE</object_type>
 <name>B1</name>
 <capacity>4</capacity>
 <rechargeable_power>2</rechargeable_power>
 <cpu_name>cpu</cpu_name>
 <e_max>1</e_max>
 <initial_energy>4</initial_energy>
</battery>
```

With all those attributes, a battery is created in Cheddar. This battery is then used by the algorithm to manage the energy level over time.

Every xmlv3 file required to schedule a task set with Cheddar also describes the type of scheduler that will be used to schedule the set:

```
<scheduling_parameters>
 <scheduler_type>EARLIEST_DEADLINE_FIRST_ENERGY_HARVESTING_PROTOCOL</scheduler_type>
 <quantum>0</quantum>
 <preemptive_type>NOT_PREEMPTIVE</preemptive_type>
 <capacity>0</capacity>
 <period>0</period>
 <priority>0</priority>
 <start_time>0</start_time>
</scheduling_parameters>
```

Here the scheduler type is the ED-H one. Thanks to this information, Cheddar will be able to schedule the task set using the right scheduling algorithm.

Different tests files were created to verify if the implementation of the ED-H scheduling in Cheddar worked. Not every case of each rule was tested individually.

I created 5 individuals rule case tests and 2 completes tests from examples presented in the paper and the thesis.

Out of the 7 possible cases of the algorithm rules, 5 are working properly, 1 is not working and 1 was not tested due to the absence of a test case.

Out of the 2 complete tests, 1 can successfully be scheduled with Cheddar, the other one cannot.

Even though the first complete set is successfully scheduled, the obtained result does not exactly match the one expected.

7- Conclusion

The objective of this project was to implement the ED-H scheduling algorithm in Cheddar. In order to complete this objective, I had access to two scientific work about ED-H and the Real-Time Energy Harvesting systems.

The results of the project are the following. Five of the seven possible cases were successfully tested either individually or within complete tests. From the two complete tests created, only one is successfully scheduled but is not the exact expected result.

During this project, problems occurred. I had difficulties when implementing certain rules, therefore I was unable to properly implement all five rules of the scheduling algorithm. Another problem I had to face was the fact that the Ada programing language is relatively new to me. This led to several mistakes in the code that I had to correct to be able to move forward.

In conclusion, this project allowed me to acquire more knowledge about Ream-Time systems. I discovered the RTEH systems when reading and analyzing the two documents at my disposal. In fact, the analysis of scientific documents in the way I was requested to was new to me and enriching.

8- References

[1] Chetto, M. Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems. IEEE Transactions on Emerging Topics in Computing. Institute of Electrical and Electronics Engineers. *IEEE Transactions on Emerging Topics in Computing*, *2*(2), 122-133, 2014

[2] El Osta, R. Contributions to Real Time Scheduling for Energy Autonomous Systems. Phd thesis of Université de Nantes. 2017