Assignment 3A – Jorrit Kroes – CPS (Cache and Scheduling)

Cache analysis

I filled in the analysis with the provided excel structure:

Name: Jorrit Kroes
Student id: 11690399

Must analysis		
Node	**Before**	**After**
1	[{}, {}, {}, {}]	[{a}, {}, {}, {}]
2	[{a}, {}, {}, {}]	[{b}, {a}, {}, {}]
3	[{a}, {}, {}, {}]	[{c}, {a}, {}, {}]
4	[{}, {a}, {}, {}]	[{d}, {}, {a}, {}]
5	[{c}, {a}, {}, {}]	[{a}, {c}, {}, {}]
6	[{d}, {}, {a}, {}]	[{b}, {d}, {}, {a}]

May analysis		
Node	**Before**	**After**
1	[{}, {}, {}, {}]	[{a}, {}, {}, {}]
2	[{a}, {}, {}, {}]	[{b}, {a}, {}, {}]
3	[{a}, {c}, {}, {}]	[{c}, {a}, {}, {}]
4	[{b, c}, {a}, {}, {}]	[{d}, {b, c}, {a}, {}]
5	[{c}, {a}, {}, {}]	[{a}, {c}, {}, {}]
6	[{d}, {b, c}, {a}, {}]	[{b}, {d}, {a, c}, {}]

Scheduling analysis

T = Period (time between consecutive intervals

D=Deadline (Deadline relative to arrival times of instances)

C=Computation time

Utilization analysis

Task-set A set has a utilization of 2/8+2/16+7/20= 29/40 (=72.5 %)

Task-set B has a utilization of 2/3+1/5+1/8=119/120 (= close to 99 %)

For **EDF** we can assess schedulability by first looking at the relation between period and deadline. In this case deadlines are all the same as the periods, which means there is no restriction on the upperbound for utilization (meaning its capable of reaching 100%).

We know that task combinations that are below this upperbound are always schedulable under EDF. Therefore, we can conclude that both A and B are schedulable under EDF, since as shown in the utilization calculations above, they are both lower than 100%.

When assessing **RM** (Rate Monotonic) scheduling this can less easily be guaranteed, as the U_bound(RM) is about 0.69, which both task-set A and B exceed.

Therefore, we need to look at the critical instance for each task in the set, meaning the situation in which all other (higher prio) tasks are ready to execute. That involves waiting the absolute worst-case amount of time and only then executing. We do this on A and B using fixed-point iteration. Since for RM the only thing that matters for prioritization is the 'release rate' (= 1/T) so for both task sets they are already given in order from highest priority to lowest (1-3).

For **A** this means:

looking at task 1 there is no task with higher prio so response time R_1 = C1 = 2.

For R_2 we start with its max computation time 2, but we have to add the possible impact of task 1 since it could pre-empt task 2. So R_2 = 2 + (2/8) * 2 = 2.5. Rounding this up we get 3 so then R_2 becomes round(2+(3/8) * 2) = 3 < 16 = D2. Therefore we move on to R_3 = 3 where we iterate over R_3(l)=1 + (R_3(l-1)/3) * 2 + (R_3(l-1)/5) * 2

- [l=1] => R_3 = 3 + (3/3)*2 + (3/5)*2=6,2
- [l=2] => R_3 = 3 + (6/3)*2 + (6/5)*2=9,2, which is rounded equal to 9
- [l=3] => R_3 = 3 + (9/3)*2 + (9/5)*2=12,6
- [l=4] => R_3=3 + (13/3)*2 + (13/5)*2=16.9
- [l=5] => R_3=3 + (17/3)*2 + (17/5)*2=21.1 > D3 = 20

So, we can conclude that task-set A is **not schedulable** under RM, as we cannot schedule the third task.

For task-set **B** we do the same (even if it seems difficult based on the high utilization, shorter task computation times means it might be easier to schedule).

Starting the process, we have R_1 = 2 + 0 = 2 < D1 = 3 so task 1 is schedulable.

Then for task 2, [l=1] => R_2 = 1 + (1/3) * 2 = 1,67 -> 2, following iterations will then be:

- [l=2] => R_2=1 + (2/3) * 2= 1,33 -> 2 < D2 = 5 ,so task 2 is also fine

For R_3 we start with C3 = 1 as well.

Then we iterate R_3(l)=1 + (R_3(l-1)/3) * 2 + (R_3(l-1)/5) * 1

- [l=1] => 1 + (1/3) * 2 + (1/5) * 1 = 1,87 -> 2
- [l=2] => 1 + (2/3) * 2 + (2/5) * 1 = 2,73 -> 3
- [l=3] => 1 + (3/3) * 2 + (3/5) * 1 = 3,60 -> 4
- [l=4] => 1 + (4/3) * 2 + (4/5) * 1 = 4,47 -> 4 < D3 = 8

So, we can conclude that all tasks of task-set **B** are indeed schedulable under RM