Solving starvation problem in a 4 core system

Majid Dareini - Amir Mahdi Chiti

January 2022

We know that in this problem we have two priority queues which both are sorted by the priority of each process in. one called the ready queue and the other one called waiting queue.

We could increase the priority of each process in waiting queue by time, depending on the priority of the last process in the ready priority queue. But it may cause starvation for the processes in the ready queue. Beacuse by increasing the priority of processes in the waiting queue, it is possible that not any of processes in the ready queue to run.

To avoid the starvation which stated above, we could set a **threshold** to control the increasing rate of priority of processes. By the formula in the following, we could have a good estimation of the threshold.

$$Threshold = \left\lceil \frac{Priority(last\_process(ready\_queue))}{2} \right\rceil \tag{1}$$

Why we calculated threshold as stated above! the ready queue is a priority queue so the last process' priority is the biggest one and if we have half of this amount we could have a good estimate that the threshold is almost in the middle of processes' priority amount.

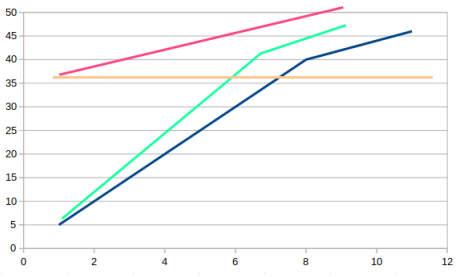
If we could have access to the processes in the queue, we could have an estimation much better than the threshold we calculated in the equation 1. If we could, we could calculate the threshold as follow.[1]

$$Threshold = \left\lceil \frac{Pr(first\_proc) + 4Pr(middle\_proc) + Pr(last\_proc)}{6} \right\rceil \tag{2}$$

In the equation 2, Pr is priority and $proc$ is process, the estimation in the equation 2 is much better than the estimation in the equation 1, because we have included the effect of middle process' priority, first process' priority aswell and it helps us to have a good estimation of threshold because we almost have found an appropriate place for the threshold in which the probability of being in the middle is more.

We said that we increase the priority of each process in every time unit. For any process in the waiting queue which its priority gets equal or more than the threshold, we decrease the rate of priority increasing.

[1] All processes we mentioned in the equation 2 are in the ready queue



For the example above, we have 3 processes which have been marked in blue, green and red and their priorities are as above. Before reaching the threshold which is marked in yellow, their increasing rate is more than after reaching the threshold.[2]

But the another problem we may have is starvation in the waiting queue. The processes in the end of the waiting queue may never could have a free resource and that's why the processes which are in the beginning of the queue may never be popped even with having resource!

To solve the problem which is stated above, we could determine a **quantum** for the time that a process in the end of the queue, this quantum helps to figure out if a process has been waiting for a long time or not, if it has, then we put a **flag** for it and set its priority to one less than the first process' priority, store its real priority and then we push it again to the queue.

So some processes may have a flag and some may not, every time we pop a process from the queue, we check if the process has a flag or not, if yes, we set the process' priority to its real priority and then push it to the ready queue.

[2]The reason that slope has changed a little after the threshold, is that the graph is continious but the priority itself increases descretely