Summer project: **Server Pools with SRPT and PS Scheduling**
Keywords: modelling, simulations, queueing theory
Supervisor: Prof. Esa Hyytiä (esa@hi.is)

Background: Consider a server with n jobs with (remaining) service times x_i. Suppose $x_1 > x_2 > \ldots > x_n$. SRPT and PS are scheduling disciplines defining the order at which the jobs are processed.

SRPT (shortest-remaining-processing-time) is the optimal scheduling discipline minimizing the mean response time. It processes first the shortest job n, then job $(n-1)$, etc.. SRPT is also *preemptive*, i.e., if a new shorter job arrives, the scheduler will switch to it *immediately*.

PS (processor sharing) processes all jobs *concurrently*. With n jobs presents, each job gets $1/n$ share of the CPU time. Job n will finish first also in this case, but it will take n times longer time! Note that PS is an adequate model multitasking operating systems where several processes are run concurrently.

Project plan: In this project, we study a system of parallel servers where arriving jobs are dispatched immediately upon arrival servers, each processing jobs according to SRPT or PS. The key dimension explored is the dispatching policy that decides on the job assignment.

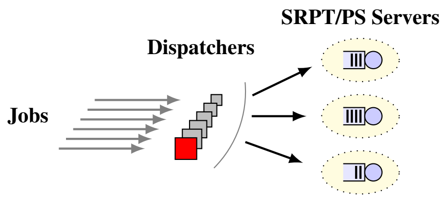
1. The most elementary dispatching policy is random split (RND) choosing server i with probability $p_i = 1/m$, where m denotes the number of servers.
2. Round-robin (RR) rotates between the servers sequentially: $1, 2, \ldots, m, 1, 2, \ldots$
3. Myopic policy minimizes the total increase in the response time given no other jobs will arrive
4. SITA sends short jobs to server 1, and long to server 2 (and similarly with more than two servers).
5. Multi-layer policies first use policy α_1 to split jobs to k secondary dispatchers, which use policy α_2 to route jobs to their final destination. In particular, we consider RR-SITA to *spread* jobs to all servers.

Tasks:

1. Implement a (fast) simulator for this task using Python, C or C++. The first version can be based on RND, but other dispatching policies must be easy to implement. Statistics collected should sufficiently general for deriving interesting performance metrics (starting from the mean response time and its variance, to whole distribution).
2. Implement other policies described above and carry out numerical experiments with them
3. Study your findings and develop a better dispatching rules:
   - A pure heuristic combining the ideas present in the given policies (e.g., a multi-layer design)
   - Machine learning approach can also be considered
4. Optional: implement and study MDP-based policies (together with the supervisor)
5. Report the work: i) description of the simulator software and ii) results of simulation experiments

Expected Skills:

- Basic programming skills (Python, C and C++)
- Not afraid of statistics and probability theory
- Keen to understand how real computing systems can be modelled
- REI503M *Performance Analysis of Computer Systems* (recommended)


SRPT/PS Servers
Dispatchers
Jobs