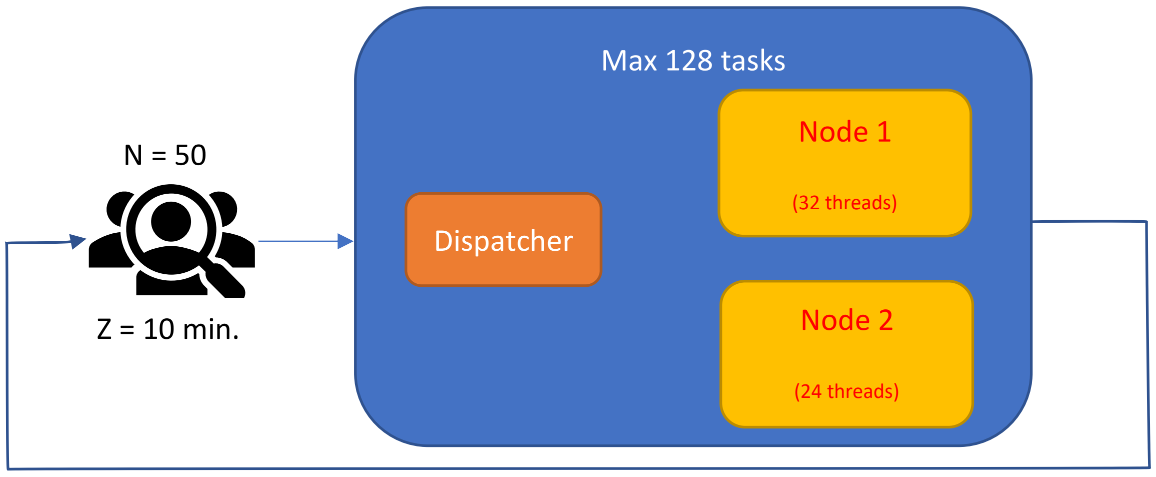
Performance evaluation of a Spark system

A Spark process is divided into three stages, each one parallelizable on a different number of tasks. Tasks are executed on a small system composed of a dispatcher and two computing nodes. The dispatcher requires an average of 2 ms. to assign a task to one of the two nodes, using a join the shortest queue approach. When all the tasks of one stage are completed, the next stage can start. When the three stages are completed, the task is over, and the user can examine the results, and submit a new request after a think time of 10 minutes. There are a total of N = 50 users submitting jobs to the system. To prevent overloading, the system allows at most 128 tasks in execution between the dispatcher and the two nodes. The two nodes are different, each one characterized by a different number of threads that can execute the tasks that can be concurrently, and by a different speed at which they can execute tasks. The table below report the relevant parameters.

	Number of tasks	Node 1	Node 2
Number of threads		32	24
Stage 1	80	10 ms.	12 ms.
Stage 2	64	20 ms.	22 ms.
Stage 3	96	15 ms.	18 ms.

A block representation of the system is the following:



Consider the three stages as three different classes, and the splitting of a job into tasks as a Fork and Join system. Use a class Switch, after the join, to let the stage advance and the job complete after the third one. Use a finite capacity region to limit the number of tasks in execution to 128. Implement the number of threads of a node with its number of servers. Consider all the execution times to be exponentially distributed, and the queuing policy to be Processor Sharing.

Compute:

1. The average response time of a job. Hint: it is simpler to compute the system response time, using the terminal as the reference station, and manually subtract the think time from the result.
2. The throughput of the system
3. The total utilization of the two nodes. Hint: JMT computes the average utilization of the nodes. Manually transform the result into the total utilization

4. The average number of tasks waiting to enter the system (the queue to the finite capacity region)

Solve the problem using the JSimGraph component of JMT. Deliver also a screen capture of the JMT model. Moreover, report only the average value, and do not worry about the confidence interval.