Università
Ca'Foscari
Venezia

Software Performance and Scalability [CM0481]

# Assignment 2: Conditioned expected response time


**Students**

Forcellato Francesco 875290

Pilotto Gabriele 902388

Visconti Pietro 885448


**Academic Year**

2023 / 2024

# Contents

**1 Introduction** **3**

**2 FCFS analysis** **4**

**3 PS analysis** **7**

**4 SRPT analysis** **8**

**5 Final considerations** **10**

5.1 All can win theory definition . . . . . 10

5.2 Plots Comparison . . . . . 10

5.3 Conclusions . . . . . 11

# 1 Introduction

The goal is to study and compare the mean slowdown of the following queuing disciplines:

- FCFS
- PS
- SRPT

The slowdown is the amount of delay imposed by a discipline normalized to the job size. Let $\bar{R}^d(x)$ be the expected response time of a job of size $x$ under discipline $d$. Then, it's slowdown is:

$$\bar{S}(x) = \frac{\bar{R}^d(x)}{x}$$

A discipline is fair if it imposes the same slowdown to all jobs.

The default values chosen for shaping the Pareto distribution used to compare the policies are the following:

$$\alpha = 1.4 \tag{1}$$

$$\rho = 0.7 \tag{2}$$

$$\text{max\_v} = 100000 \tag{3}$$

# 2 FCFS analysis

FCFS is a scheduling policy which consists of always choosing **the job at the head of the queue** to be served and running that job to completion. Intuitively, we can deduce that little jobs may potentially encounter bigger jobs ahead on the queue, and consequently would deal with huge slowdowns compared to their job sizes.

For instance, let $A$ and $B$ be jobs with expected service times **1s** and **100s**, respectively. Assume that $A$ arrives immediately before $B$. The FCFS policy will, of course, serve the $A$ job first and $B$ after. Given this background, let's compute the response times (waiting time + service time):

$$R_A = 0s + 1s = 1s$$

$$R_B = R_A + 100s = 101s$$

Now, we can compute the mean response time as

$$E[R] = \frac{R_A + R_B}{2} = \frac{1s + 101s}{2} = 51s$$

This sounds like a pretty fair policy: given those job sizes, the mean response time is reasonable.

Nevertheless, let's swap the positions in the queue: assume that $B$ arrives immediately before $A$ and apply the same calculations:

$$R_B = 0s + 100s = 100s$$

$$R_A = R_B + 1s = 101s$$

And the mean response time turns out to be:

$$E[R] = \frac{R_A + R_B}{2} = \frac{101s + 100s}{2} = 100.5s$$

Compared to the first example, **the mean response time is almost doubled**. We can deduce that FCFS policy is not fair in all cases and it requires some assumptions to work properly.

One of those assumptions is the value of the **variance of the jobs' sizes** (or expected service times). In the previous examples, we saw a particular case in which the variance was extremely big and we came up with disappointing conclusions

about this policy. Let's have one last example with $A$ and $B$ having sizes **5s** and **5s**, respectively, and the order by which they arrive in the queue is insignificant since the sizes are the same. The expected response times are:

$$R_A = 0s + 5s = 5s$$

$$R_B = R_A + 5s = 10s$$

And the mean response time is:

$$E[R] = \frac{R_A + R_B}{2} = \frac{5s + 10s}{2} = 7.5s$$

Notice that, in this case, **swapping the positions doesn't interfere at all on** $E[R]$. We can deduce that:

FCFS is a scheduling policy that works properly given these assumptions:

1. **The job sizes are known in advance** (not always possible);
2. **The variance related to the job sizes is equal or near 1**.

Mathematically speaking, let's prove this statement by relying on the **P-K formula**. In a **M/G/1 queue**, the expected response time is equal to:

$$\overline{R} = E[R] = \frac{\rho + \lambda\mu\sigma^2}{2(\mu - \lambda)} + \frac{1}{\mu} = \frac{\rho(1 + CV^2)}{2(\mu - \lambda)} + \frac{1}{\mu}$$

In which the first element indicates the waiting time (the P-K formula, indeed) and $\frac{1}{\mu}$ is the service time. We can notice that $CV$ is exactly the variance that we mentioned before. Since it is on the numerator, huge values of variance would greatly increase the actual value of the waiting time and, by contrast, a variance near 1 would result as acceptable waiting time values.

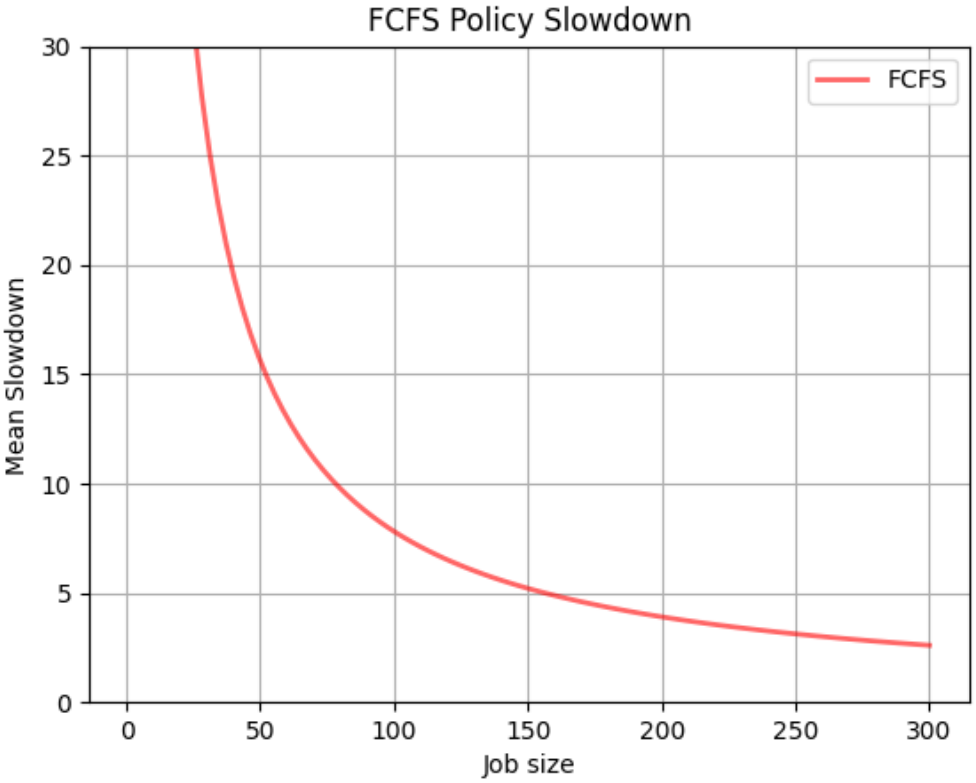



Figure 1: FCFS plot

# 3 PS analysis

Processor sharing is a discipline where **resources are equally split between users**. Every time a job enters the system, a fraction of processor is assigned to the user, so less customers are using resources, more resources are assigned to a single customer. By this definition we can assume that this **Processor Sharing is intrinsically fair**. To demonstrate it, we created a plot where we first propose the slowdown in relation to the job size. Slowdown is computed as $\overline{S}(x) = \frac{1}{1-\rho}$ where $\rho$ is our load factor calculated as $\rho = \frac{\lambda}{\mu}$. As in the previous analysis, we assigned to $x$ a sequence of 100 equally distant values from 1 to 300 (included).

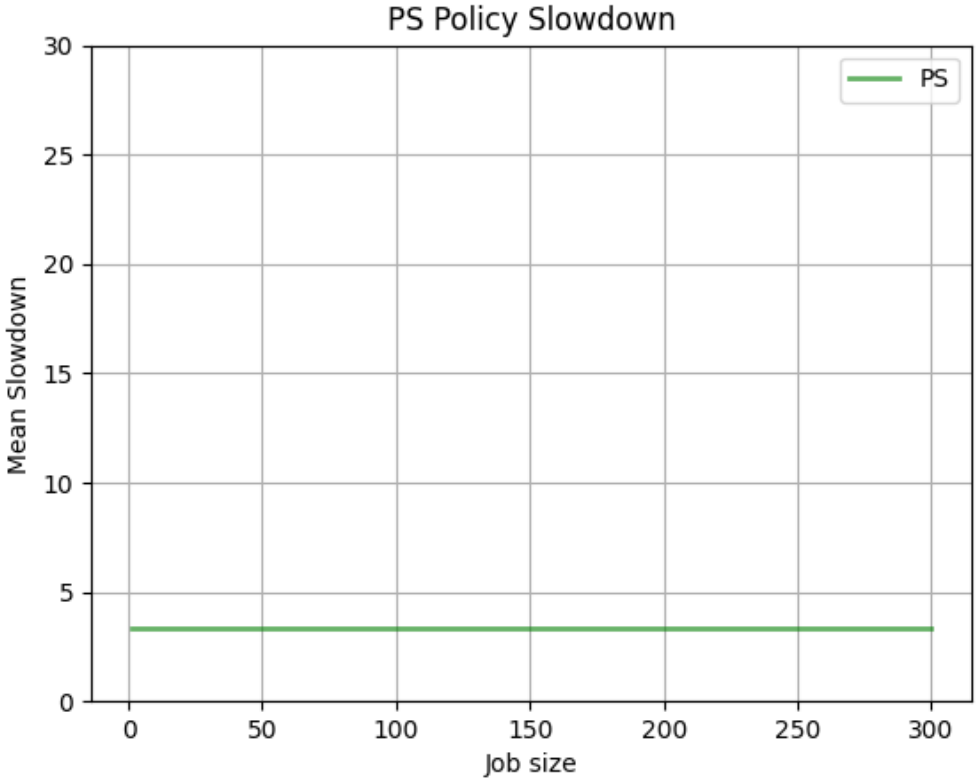



Figure 2: Processor Sharing plot

As expected, the slowdown remain unchanged since in processor sharing discipline changing job size does not affect slowdown confirming that this discipline is fair. This makes processor sharing a great solution in scenarios where workload is variable and unpredictable since it **allows every user to get access at least to a portion of processing resources** without increment slowdown.

# 4 SRPT analysis

The *Shortest Remaining Processing Time* is a preemptive discipline with resume. It schedules the jobs based on the shortest remaining work to be done. It minimize the expected response time with respect to any other discipline. It is difficult to implement in real world scenarios because it is not always clear how much work remain to a particular job. It is also worth noting that large jobs may be starved since they might wait forever before being computed.

In order to calculate the slowdown of *Shortest Remaining Processing Time* we can use the following formulas.

Let $f(t)$ be the pdf of the service time distribution and $F(t)$ its corresponding cdf. This means that:

$$\mu^{-1} = \int_0^\infty t f(t) dt$$

$\mu^{-1}$ is the average service time (in seconds) and $\mu$ is the service rate measured in jobs per second. $\lambda$ is the arrival rate measured in jobs per second. Suppose that we consider the jobs up to a certain size $x$, then the load factor due to these jobs is:

$$\rho(x) = \lambda \int_0^x t f(d) dt$$

and the second moment is:

$$m_2(x) = \int_0^x t^2 f(t) dt$$

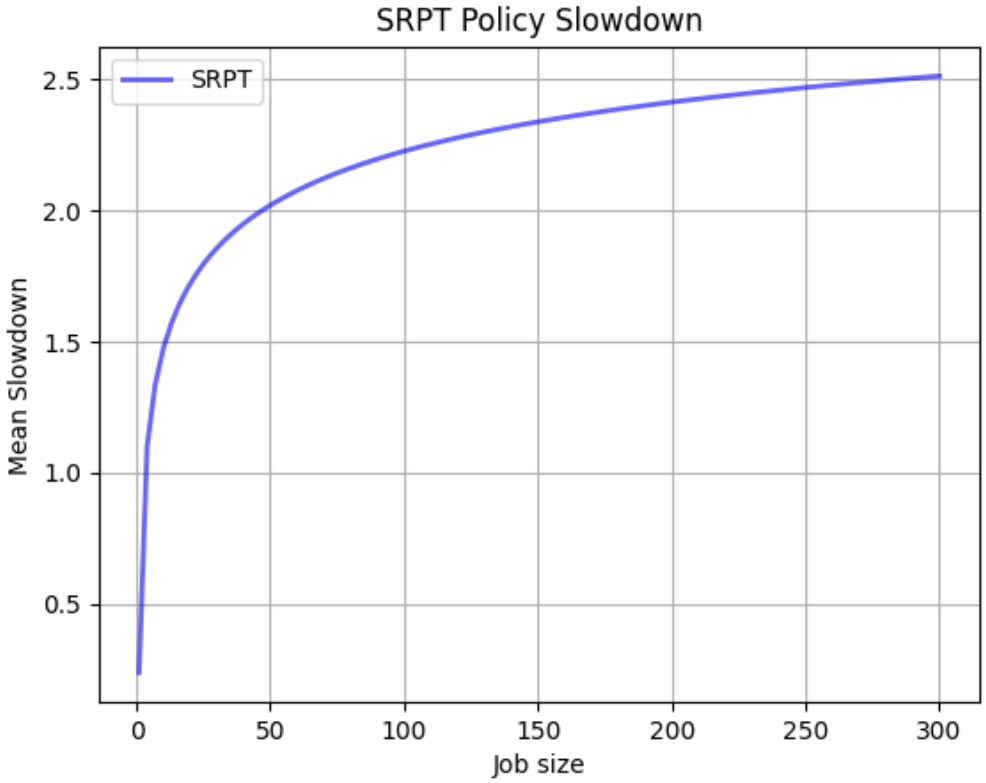
The conditional expected response time is:

$$\bar{R}^{SRPT}(x) = \frac{\lambda(m_2(x) + x^2(1 - F(x)))}{2(1 - \rho(x))^2} + \int_0^x \frac{1}{1 - \rho(t) dt}$$

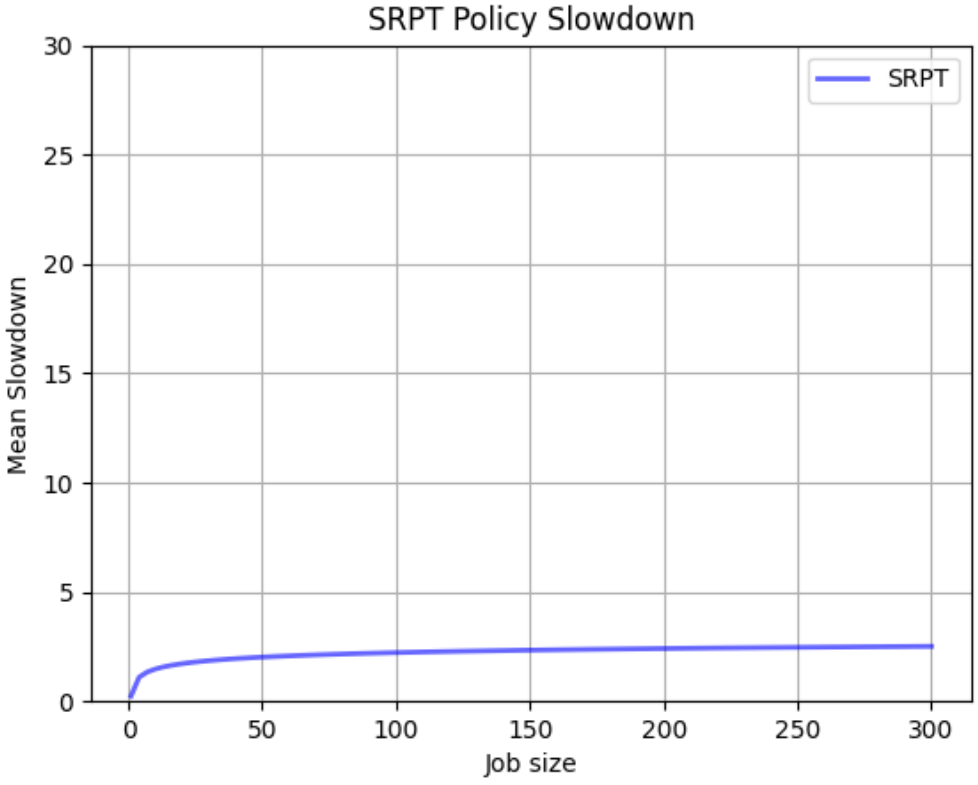
So the slowdown can be computed as follows:

$$\bar{S}^{SRPT}(x) = \frac{\bar{R}^{SRPT}(x)}{x}$$

In the plot of figure 3a it is represented the SRPT graph in the whole spectrum, in this way it is possible to study the shape of the plot. In the figure 3b it is represented the same graph, using the same values, but with the $y$ axis resized so that we can focus and compare it with values interesting also for the other scheduling disciplines.




(a) SRPT plot: all the values




(b) SRPT plot: $y$ axis is resized to make the plot comparable to the other ones

# 5 Final considerations

## 5.1 All can win theory definition

Given an M/G/1 if $\rho < \frac{1}{2}$ then, $\forall x$,

$$E\left[T(x)\right]^{SRPT} < E\left[T(x)\right]^{PS}$$

Assuming $\rho < \frac{1}{2}$, the All-Can-Win theorem says that every single job (every x) prefers SRPT to PS in expectation, and this property holds until $\rho < 0.96$.

## 5.2 Plots Comparison

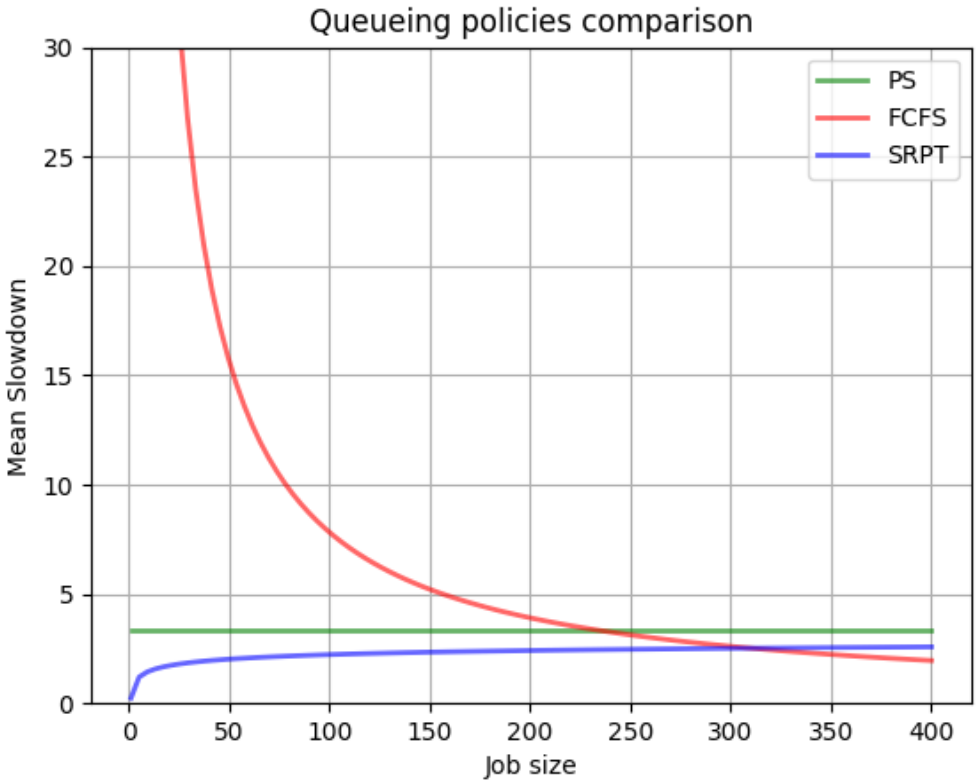



Figure 4: All combined plots for making comparisons.

## 5.3 Conclusions

Taking a look at the plots, we can deduce that also SRPT is a fair discipline for all job sizes until 400. We can see how the SRPT discipline is always below the PS plot, and consequently, the All-Can-Win theory holds having $\rho = 0.7$; therefore this property is also valid for $\rho = 0.5$. As seen previously, a discipline satisfies the All-Can-Win theory when $\rho$ it's at least $< 0.5$, so the results obtained respect the requirements.