A n-core non-stochastic resource sharing system

Erik Leonards

February 26, 2023

1 The general setup

A n core resource sharing system consists of a queue, n-cores and shared resources. Arriving tasks are placed in a spot in the queue according to a queueing policy. The queueing policy will be First In First Out (FIFO) unless listed otherwise. The task at the front of the queue is moved to the first available core. The core will then execute the entire task while sometimes requesting for the shared resources. We will give more details in the next subsections.

1.1 The tasks

The inter-arrival times between the tasks are $E(\lambda)$ distributed (exponential distribution with parameter λ). The execution requirement R of the tasks is $E(\alpha)$ distributed. Let $p \in [0, 1]$ be fixed for all tasks and that indicates which part of the execution requirement requires shared resources. The core doesn't require the shared resources when $p = 0$ and the core needs shared resources for the entire execution requirement if $p = 1$. To be more specific, for a task having service requirement $r = R$, we have that a execution requirement of pr needs shared resources and $(1-p)r$ doesn't need shared resources. We will refer to the execution part with and without shared resources as respectively the core-part and the shared-part.

1.2 The task execution

When a core receives a task with execution requirement $r = R$ we have noted earlier that the core-part is equal to $(1-p)r$. The core can execute this part with full capacity so the execution rate is 1, which means it takes $(1-p)r$ time

to execute this part. It is also known that the shared-part is pr. The shared resources are equally divided between the active cores requesting them. To be more precise, when there are $c \leq n$ number of cores executing tasks, and if there are $s \leq c$ active cores executing the shared-part, then each of the s cores has an execution rate of $\frac{1}{s}$. Notice how this creates an interesting dynamic whenever there are more active cores at the same time:

more active cores
$\implies$ **more cores in shared-part at the same time**
$\implies$ execution rate of shared-part decreases
$\implies$ execution time of shared-part increases
$\implies$ all cores spend larger proportion of execution time in shared-part
$\implies$ **more cores in shared-part at the same time**.

It is very important to note that the execution time of the core-part is fixed at $(1-p)r$ given an execution requirement of $R = r$, while the execution time of the shared-part is dependent on the other cores. This introduces a stochasticity in the execution time. This stochasticity is dependent on the way the cores handle the tasks. A task-handling approach could for example be that the core-part is done first and then the shared-part, or the other way around. It could also be the case that the core switches many times between the core- and shared-part. In next section we will formulate and analyse two different approaches and motivate which one we will choose for the model.

2 Task-handling approaches for two servers

In this section are formulated and analysed two different approaches for the order in which every core handles the core- and shared-part. An important distinction is that the core switches only once between the core-part and the shared-part in the first approach, while in the second approach the core switches infinitely many times between the core- and shared-part.

2.1 The Core Shared approach

One way to approach this is by the Core Shared (CS) approach, which involves first doing the core-part and then the shared-part. The analysis of this approach is easy when there is only one active processor, because the execution rate will be equal to 1 during both the core- and shared-part (Figure 1).

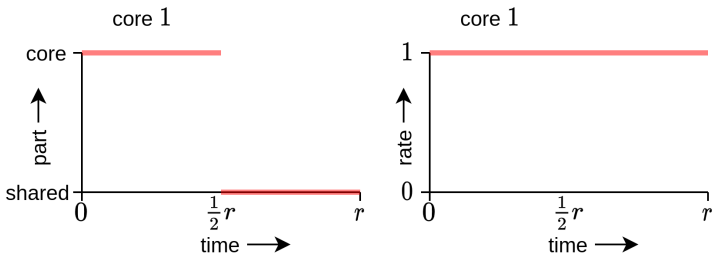


Figure 1: The execution rate during the core-part and the shared-part when $p = \frac{1}{2}$ and the execution requirement is r.

The complexity of the situation is greatly increased with two active cores. A possible scenario with $p = \frac{1}{2}$ is shown in Figure 2.

At time 0 a task arrives in core 1 and at time $\frac{1}{2}r$ the execution of the core-part is finished and the shared-part begins. A task is already running in core 2 at time 0 and at time t_1 a new task arrives in core 2. For simplicity we have set the execution requirement of this task in core 2 to be equal to $\frac{1}{2}r$. This means that the core-part has a execution requirement of $\frac{1}{4}r$ and is run in $\frac{1}{4}r$ time. The shared-part of this task overlaps completely with the shared-part of the task in core 1. This means that the execution rate is halved and it takes twice as long to finish the shared-part of the task in core 2.

In core 1 the shared-part starts at time $\frac{1}{2}r$ and from t_2 to t_3, the rate of the shared-part is halved. We know that $t_3 - t_2 = \frac{1}{2}r$ and during this time the execution rate in the shared-part is halved. This means that an execution requirement of $\frac{1}{4}r$ is done and there is still a requirement of $\frac{1}{4}r$ left to be done in the shared-part. This means core 1 finishes the task after $\frac{5}{4}r$ time.

The execution time of the task in core 1 can be generalised. For a task arriving in core 1 with execution requirement r, let t_c and t_s be respectively the time it takes for the core- and shared-part of the task in core 1 to be executed. Now let $t = t_c + t_s$ be the total execution time of the this task in core 1. Also define

$$X \coloneqq \frac{t_{1,2}}{t_s}$$

where $t_{1,2}$ indicates the time that both core 1 and 2 use the shared resources. Notice that X is equal to the proportion of the shared-time of core 1 that core 2 also uses the shared resources. Note that X is stochastic and between 0 and 1. The shared-part time of the task in core 1 is then equal to

$$t_s = t_{1,2} + (pr - \frac{1}{2}t_{1,2}) = pr + \frac{1}{2}t_{1,2} = pr + \frac{1}{2}Xt_s,$$

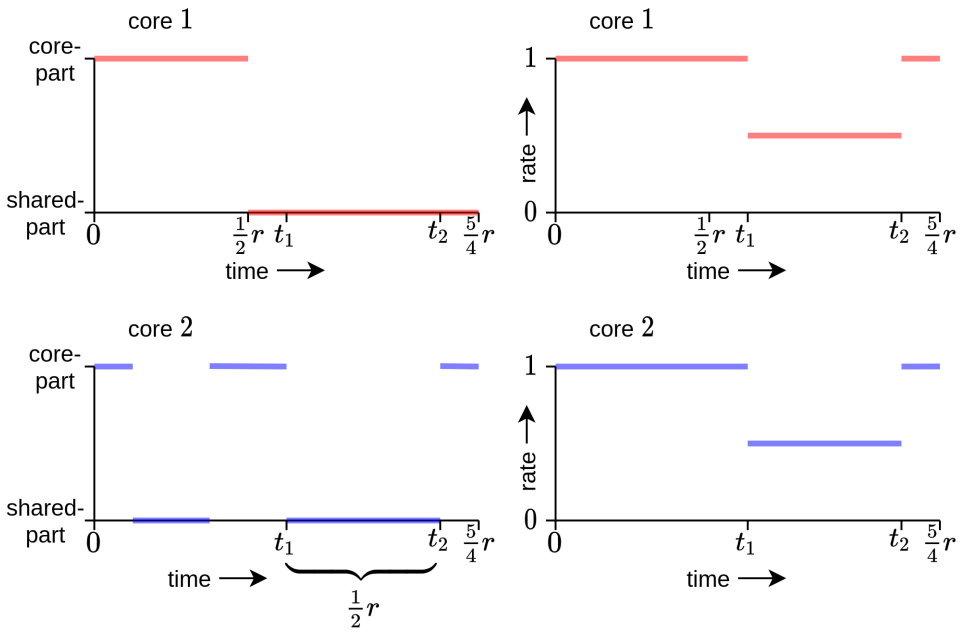


Figure 2: A possible scenario of the execution rate in two cores when $p = \frac{1}{2}$ and the execution requirement is r.

because $\frac{1}{2}t_{1,2}$ is the requirement done during the time that both 1 and 2 are using the shared resources. This means a service requirement of $pr - \frac{1}{2}t_{1,2}$ is done when all shared resources are available for core 1. From the formula we can deduce that

$$t_s(1 - \frac{1}{2}X) = pr \implies t_s = \frac{pr}{1 - \frac{1}{2}X}.$$

Moreover, we have that

$$t = t_c + t_s = (1 - p)r + \frac{pr}{1 - \frac{1}{2}X},$$

and observe that t is stochastic even when the execution requirement is known.

2.2 The Infinite Mix approach

The previous approach considered the case in which the core only switches between the core-part and the shared-part once. The disadvantage of this approach is that the total time t is stochactic for a given execution requirement r. The Infinite Mix (IM) approach aims at eliminating this stochasticity. The

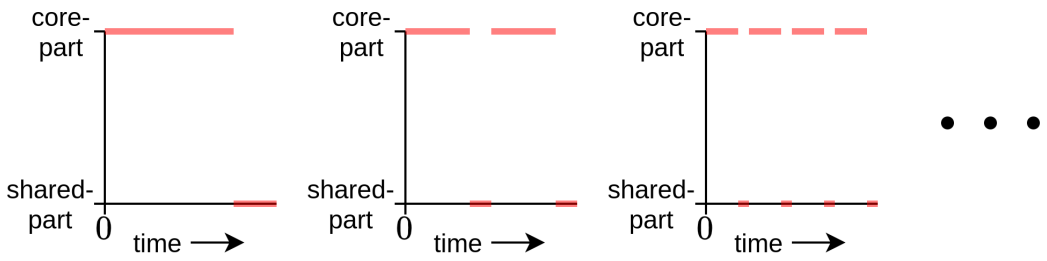


Figure 3: The approaches $A^{(0)}, A^{(1)}$ and $A^{(2)}$ for $p = \frac{3}{4}$. The IM approach is the limit of these approaches.

IM approach is the limit of the approaches $A^{(n)}$ in which the core switches after having executed 2^{-n} part of the core- and shared-part (Figure 3).

This has the advantage that the execution rate in the shared resources is only dependent on the number of active cores, because all active cores will use the shared resources equally in each time interval. This is different from the CS approach, for which the execution time was stochastic. Next sections are devoted to determining the exectution time of a task arriving at a free whenever there are other cores active.

2.2.1 one active core

In this section we are determining the execution time t of a task arriving at a core whenever there is the only task in the system.

It holds that the $t = t_c + t_s$ where t_c and t_s are respectively the time for the core- and shared-part. Now define

$$t^{(1)} := \frac{t_s}{t},$$

which is the relative portion that this task spends for the shared-part. We can use this to write

$$\begin{aligned} t = t_c + t_s &= (1-p)r + t^{(1)}t \\ \implies (1 - t^{(1)})t &= (1-p)r \\ \implies t &= \frac{(1-p)r}{1-t^{(1)}}. \end{aligned}$$

Observe that the value of t only depends on $t^{(1)}$ and therefore we will devote the next part of this section to determining this value.

In Figure 4 is shown a diagram of one active core, where the grey area indicates time spend in the shared-part and the other area indicates time

spend in the core part. Since we use the IM approach, the length of the time interval doesn't matter. It only matters that the core executes the same proportion of the core- and shared-part. To be more precise, it must hold that

$$\begin{aligned}
&\text{executed proportion in core-part} = \text{executed proportion in shared-part} \\
\implies &\frac{1-t^{(1)}}{(1-p)r} = \frac{t^{(1)}}{pr} \\
\implies &\frac{1-t^{(1)}}{1-p} = \frac{t^{(1)}}{p} \\
\implies &t^{(1)} = p.
\end{aligned}$$

Observe that the value of the service requirement r is not relevant. This makes intuitive sense because we are looking at proportion instead of duration. From this we can deduce that

$$t = \frac{(1-p)r}{1-p} = r,$$

which is exactly what you would expect with one active core.

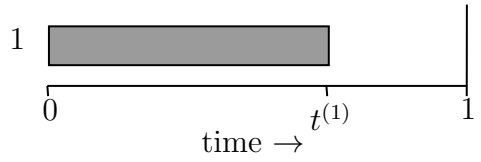


Figure 4: The time diagram of 1 active core for an interval from 0 to 1 in which both cores are active the entire time. The grey area indicates the shared-part and the other area is the core-part.

2.2.2 Two active cores

This can be extended to two active cores. A time diagram of this is shown in Figure 5a. Notice how core 1 divides the time interval into two parts. Due to the IM approach, core 2 needs to execute in each time interval an equal proportion of the shared- and core-part. This gives arrives to two intervals (which are $[0, t_1^{(2)}]$ and $t_2^{(2)}$) in which core 2 executes the shared-part. Since we are only interested in $t^{(1)}$ value of $t_2^{(2)}$ doesn't influence the value of $t^{(1)}$,

we can ignore this valu and arrive at a simplified time diagram in Figure 5b. For this diagram it must hold that

$$\frac{1-t^{(1)}}{1-p} = \frac{(t^{(1)}-t^{(2)}) + t^{(2)} \cdot \frac{1}{2}}{p},$$

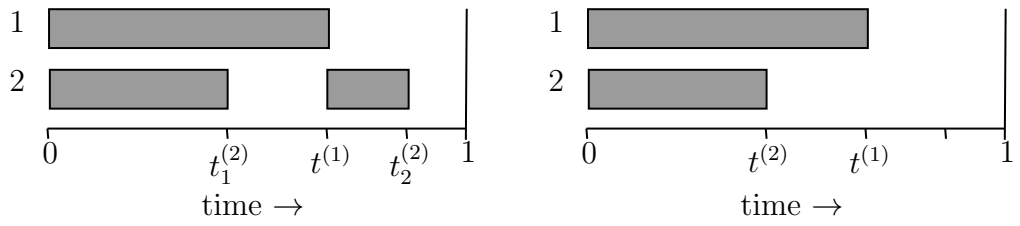
because during time interval $[t^{(1)}, t^{(2)}]$ core 1 achieves a service rate of 1 while during the time interval $[0, t^{(2)}]$ core 1 achieves a service rate of $\frac{1}{2}$.

Furthermore, we must have that

$$\frac{t^{(1)}-t^{(2)}}{1-p} = \frac{t^{(2)} \cdot \frac{1}{2}}{p},$$

because core 2 is doing the core-part in time interval $[t^{(2)}, t^{(1)}]$ and the shared-part in time interval $[0, t^{(2)}]$ with a service rate of $\frac{1}{2}$. This system of equations can easily be solved to obtain $t^{(1)} = \frac{p(p+1)}{p^2+1}$. This means that

$$\begin{aligned}
t &= \frac{(1-p)r}{1-t^{(1)}} \\
&= \frac{(1-p)r}{1-\frac{p(p+1)}{p^2+1}} \\
&= \frac{(1-p)(p^2+1)}{p^2+1-p(p+1)} \\
&= \frac{(1-p)(p^2+1)r}{1-p} \\
&= (p^2+1)r.
\end{aligned}$$



(a) The full time diagram.

(b) The simplified time diagram.

Figure 5: The time diagrams of 2 active cores for an interval from 0 to 1 in which both cores are active the entire time. The grey areas indicate the shared-part and the other areas are the core-part.