The shared memory model as closed queueing network

Erik Leonards

March 21, 2023

1 The processor

Consider a processor with n cores that are executing tasks. An active core sometimes want to access the memory. The core does this by sending memory requests to the memory. It might be the case that the return value of the request is only needed in the far future, which means that the core can continue execution of the task while the memory request is being sent and the core doesn't need to wait. It could also be the case that the memory request is necessary in the nearby future, which would imply that the core can only do a little bit of execution of the task before it has to wait for the memory request to return. The measure for waiting that is normally used for processors is utilisation.

1.1 The utilisation

The utilisation for a certain time interval $[t_1, t_2]$ is defined as

$$\text{utilisation} = \frac{\text{active clock cycles}}{\text{total clock cycles}},$$

where the active clock cycles indicate the cycles where the core is actively doing computations. This means that the total clock cycles are made up of the waiting clock cycles and the active clock cycles.

The utilisation changes during the execution of a task, but for now we assume that the utilisation is uniform during the entire execution of the task.

1.2 The bandwidth

It is important to know that the utilisation changes when more cores are active. This is because the memory only has a certain capacity (called band-

width and denoted with B) with which it can execute the memory requests. The bandwidth is then equal to the number of memory requests it can simultaneously execute. When the rate at which the memory requests arrive is larger then the bandwidth, then a queue in the memory is formed.

Example 1.

- *Let $B = 100$ and suppose there are 2 cores sending memory requests with respectively a rate of 60 and 30. Since the combined rate of the memory requests is $90 < 100 = B$ we have that there is no queue forming in the memory. This means that the isolated execution of the two tasks is equal to the execution of the two tasks when run simultaneously.*

- *Let $B = 100$ and suppose there are 3 cores sending memory requests with respectively rates of $70, 40$ and 10. The combined rate is $120 > 100 = B$, so a queue will form in the memory. Core 1 gets for example only requests back with a rate of 60, core 2 with a rate of 32 and core 3 with a rate of 8. This means that return time of a request is delayed. Since the return values are needed for the execution of the task in the future, the core is slowing down in comparison with the isolated execution. This will have the affect that it takes longer to execute a task and the rate at which memory requests are sent decreases. The rate at which the memory requests are sent will decrease until the arrival rate and the departure rate of requests is the same and an equilibrium is reached.*

1.3 The scenario

We will now consider the case of n tasks running simultaneously without knowing whether the bandwidth is reached.

2 The mathematical model

There are three possible ways of modelling the memory. You can either model it as a First In First Out (FIFO) queue, where B number of memory requests can be handled at the same time, each with a service rate of μ. A service rate of μ means that the service time is $E(\mu)$ (exponentially with parameter μ) distributed. This will be called the FIFO Memory Queue (FIFO-MQ) model. This model can be slightly modified to obtain the Random Order of Service Memory Queue (ROS-MQ) where the next one to be served by the memory is randomly chosen from all requests currently waiting in the memory that want to be served.

An alternative model is when the memory has B bandwidth, where each memory requests receives a service rate of

$$\min\left\{\mu, \frac{B}{x}\mu\right\},$$

when there are x memory requests in the system. This means that the service time of each of the x requests is $E\left(\min\left\{\mu, \frac{B}{x}\mu\right\}\right)$ distributed. This case will be called the Bandwidth Sharing (BS) model, because the bandwidth is equally shared between the requests and each request can receive a maximum service rate of μ.

2.1 The requests

There are requests moving between the cores and the memory. The total number of these requests is fixed. For n active cores, let $m_1, m_2, \ldots, m_n$ be the number of requests in the system where m_i is the number of requests belonging to core i. Now define

$$m \coloneqq \sum_{i=1}^{n} m_i.$$

If there are k requests in core i, then core i is sending requests with a rate of $\lambda_i(k)$ to the memory, where $\lambda_i(0) = 0$ and λ_i is increasing for every i. This means that the inter-arrival times of requests are $E(\lambda_i(x))$ distributed. This means that the rate is dependent on the number of requests currently in the core. This is in line with the theory, because many requests in the core imply that the the core has just received many requests, so it can probably keep executing the task and sending requests to the memory, without having to wait.

Depending on the model, the memory is sending the requests back with a certain rate. Now define

$$X(t) = (X_0(t), X_1(t), \ldots, X_n(t))$$

as the state of the system at time t, where $X_0(t)$ is the total number of requests in the memory and $X_i(t)$ $(1 \leq i \leq n)$ is the number of core-i-requests in core i. Now let $x = (x_0, x_1, \ldots, x_n)$ be a possible state of the system, where x_0 indicates the total number of requests in the memory and $x_1, x_2, \ldots, x_n$ indicate respectively the number of requests in cores $1, 2, \ldots, n$. Now define

$$\psi \coloneqq \left\{x = (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} | 0 \leq x_i \leq m_i \forall i, \sum_{i=0}^{n} x_i = m\right\}$$

as the state space and observe $X(t) \in \psi$. The begin state is given by $X(0) = (0, m_1, m_2, \ldots, m_n)$, which means that all requests start in the core at time 0. Now define

$$X_i \coloneqq \lim_{t\to\infty} \mathbb{E}[X_i(t)] \quad i = 1, 2, \ldots, n$$

as the expected number of core-i-requests in core i when time goes to infinity. Observe that this limit exists under the assumption that the distribution of $X(t)$ will converge to an invariant distribution (equilibrium distribution). Now also define

$$X_0 \coloneqq \lim_{t\to\infty} \mathbb{E}[X_0(t)]$$

as the expected number of requests in the memory and keep in mind that

$$X_0 = \sum_{i=1}^{n}(m_i - X_i) = m - \sum_{i=1}^{n} X_i.$$

Now also define

$$X = (X_0, X_1, \ldots, X_n).$$

2.2 The Bandwidth Sharing model

The Bandwidth Sharing (BS) model is illustrated in Figure 1. The leaving and arriving memory requests are modeled as red and green dots moving between the memory and the cores. In Figure 1, we have that red dots belong to core 1 and green dots belong to core 2. This means that $m_1 = 13$, because 8 requests (dots) are in the memory being processed and 5 requests (dots) have returned from the memory and are currently in the core. Similarly for core 2, we have $m_2 = 7$.

2.2.1 The departure of requests from the memory

The requests in the memory have been send by the core and the memory must fulfill the requests. In the bandwidth sharing model the bandwidth of the memory is equally divided between all requests in the memory. This means that a memory running at full bandwidth is serving each of the m requests with a rate of $\frac{B}{m}\mu$, where μ is the rate at which isolated requests are served. In Figure 1 each red dot in the memory is served with a rate of $\frac{B}{12}\mu = \frac{4}{12}\mu = \frac{1}{3}\mu$, because there are 12 total requests in the memory. Since there are currently 8 core1-requests in the memory, the rate at which core1-requests are returning to core 1 is equal to $\frac{8}{3}\mu$. Similarly, the core2-requests are returning to core 2 with a rate equal to $\frac{4}{3}\mu$.

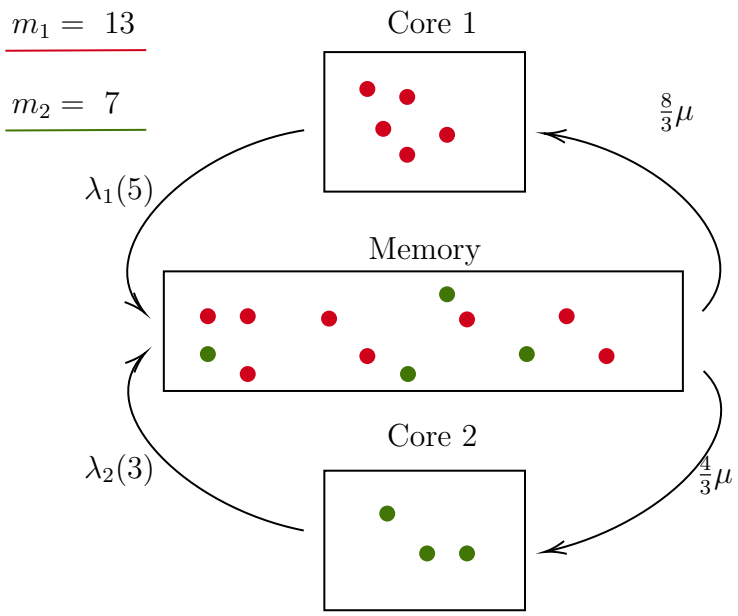

Figure 1: The overview of the bandwidth sharing model where the red and green dots are the requests. We have $B = 4$.

2.2.2 The equilibrium

If at time t the core-i-memory rate $\lambda_i(X_i(t))$ is greater than the memory-core-i rate, then it is expected that $X_i(t)$ will decrease. This means that $\lambda_i(X_i(t))$ will decrease. This means that $X_0(t)$ (the number of requests in memory) will increase and the memory-core-i rate will increase. This will happen until the memory-core-i rate is equal to the core-i-memory rate and an equilibrium is reached.

To determine the equilibrium probabilities $X = (X_0, X_1, \ldots, X_n)$ we need to specify the function $\lambda_i(x)$. We must have that $\lambda_i(x)$ is increasing, because more requests in core i means that the core has more information and can faster execute the task and send requests to the memory. A possible approach would be to let $\lambda_i(x)$ be linear, which means that $\lambda_i(x) = \lambda x$ for a certain $\lambda > 0$. Notice how we assume that this value of λ is equal for all $i \leq n$.

To determine the equilibrium we need to distinguish between two cases ($X_0 \leq B$ and $X_0 > B$).

2.2.3 The case $X_0 \leq B$

In the equilibrium it must be the case for all $i \leq n$ that the arrival and departure rate at the core are equal:

$$\lambda_i(X_i) = \lambda X_i = (m_i - X_i)\mu \quad \Longrightarrow \quad X_i = \frac{\mu m_i}{\mu + \lambda}.$$

This means that

$$X_0 = \sum_{i=1}^{n} \left(m_i - \frac{\mu m_i}{\mu + \lambda} \right) = \frac{\lambda m}{\mu + \lambda}.$$

Since $X_0 \leq B$ we must have

$$X_0 = \frac{\lambda m}{\mu + \lambda} \leq B.$$

2.2.4 The case $X_0 > B$

Memory interference between requests in the memory occurs if $M > B$. We must have an equal arrival and departure rate of requests in the core and memory:

$$\lambda X_i = \frac{(m_i - X_i)B\mu}{X_0}.$$

Since

$$X_0 = \sum_{i=1}^{n}(m_i - X_i) = m - \sum_{i=1}^{n} - \sum_{i=1}^{n} X_i,$$

we can solve this set of equations to obtain the solution of

$$X_i = \frac{m_i B\mu}{\lambda m}, \quad i = 1, 2, \ldots, n$$
$$X_0 = m - \frac{\mu B}{\lambda}.$$

Since $X_0 > B$, we must have

$$\begin{aligned} & m - \frac{\mu B}{\lambda} > B \\ \Longrightarrow \; & m > (1 + \frac{\mu}{\lambda})B = \frac{\mu + \lambda}{\lambda} B \\ \Longrightarrow \; & \frac{m\lambda}{\mu + \lambda} > B. \end{aligned}$$

This means the two cases can be summarised to obtain

$$X_i = \begin{cases} \frac{\mu m_i}{\mu+\lambda} & \text{if } \frac{\lambda}{\mu+\lambda} \leq B \\ \frac{m_i B \mu}{\lambda m} & \text{otherwise} \end{cases} \quad i = 1, 2, \ldots, n,$$
$$X_0 = \begin{cases} \frac{\lambda m}{\mu+\lambda} & \text{if } \frac{\lambda m}{\mu+\lambda} \leq B \\ m - \frac{\mu B}{\lambda} & \text{otherwise} \end{cases}. \tag{1}$$

2.3 The equilibrium distribution

Define for $k \in \mathbb{N}$

$$\phi_i(k) = \begin{cases} \lambda_i(k) & : 1 \leq i \leq n \\ \min\{k, B\}\,\mu & : i = 0 \end{cases}$$

as the rates at which requests are leaving the memory ($i = 0$) and the cores ($i = 1, 2, \ldots, n$). We define for $x \in \psi$

$$\tilde{\pi}(x) \coloneqq \prod_{j=0}^{n} \prod_{k=1}^{x_j} \frac{m_j - X_j}{\phi_j(k) X_0},$$

and we will now proof that for the normalised version

$$\pi(x) \coloneqq \frac{\tilde{\pi(x)}}{\sum_{x \in \psi} \tilde{\pi}(x)},$$

it holds that it is equal to the invariant distribution which is approached when time goes to infinity:

$$\pi(x) = \lim_{t \to \infty} \mathbb{P}[X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n].$$

To prove this, we need to define for $i = 0, 1, \ldots, n$

$$\delta_i : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1} : (x_0, x_1, \ldots, x_n) \mapsto (x_0, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_n)$$

the function that adds 1 to component i of the system. Also for $x, \tilde{x} \in \psi$ with $x \neq \tilde{x}$ let $q(x, \tilde{x})$ be the rate at which state x is changed to state $\tilde{x}$. Observe that only requests can move from i to 0 and from 0 to i for $i = 1, 2, \ldots, n$. This means that

$$q(\delta_i x, \delta_0 x), q(\delta_0, \delta_i x) \quad i = 1, 2, \ldots, n,$$

are the only non-zero entries. We have

$$q(\delta_i x, \delta_0 x) = \lambda_i(x_i + 1) \quad i = 1, 2, \ldots, n,$$

because it is the rate at which core i requests are moving to the memory. Furthermore, we also have that

$$q(\delta_0 x, \delta_i x) = \frac{m_i - X_i}{X_0} \min\{x_0 + 1, B\}\, \mu,$$

because it is expected that the proportion of core-i-requests in the memory is equal to

$$\frac{m_i - X_i}{X_0}.$$

If can now be easily checked that

$$\tilde{pi}(\delta_i x) \sum_{j \neq i} q(\delta_i x, \delta_j x) = \sum_{j \neq i} \tilde{pi}(\delta_j x) q(\delta_j x, \delta i x).$$

This directly shows that $\tilde{\pi}(x)$ is an invariant measure and since ψ is finite, we must have that the normalised version $\pi(x)$ is an invariant distribution.

2.4 The ROS Memory Queue model

The Random Order of Service Memory Queue (ROS-MQ) model is illustrated in Figure 3. This model is very similar to the BS model, but the difference is that at most B number of requests are served with a rate of μ. Each request in the memory is now either in the waiting area, or it is being served with service rate μ. In Figure 2 there are 3 red dots being served by the memory, so the core1-arrival rate is 3μ. Similarly, there is 1 green dot that is being served, so the core2-arrival rate is μ.

2.4.1 The equilibrium

Just like the BS model we are interested in finding the equilibrium of the model when the system has n active cores. Let C_i, W_i and E_i be the number of core-i-requests in respectively core i, the waiting area and the execution area. Define

$$C \coloneqq \sum_{i=1}^{n} C_i,$$
$$W \coloneqq \sum_{i=1}^{n} W_i,$$
$$E \coloneqq \sum_{i=1}^{n} E_i.$$

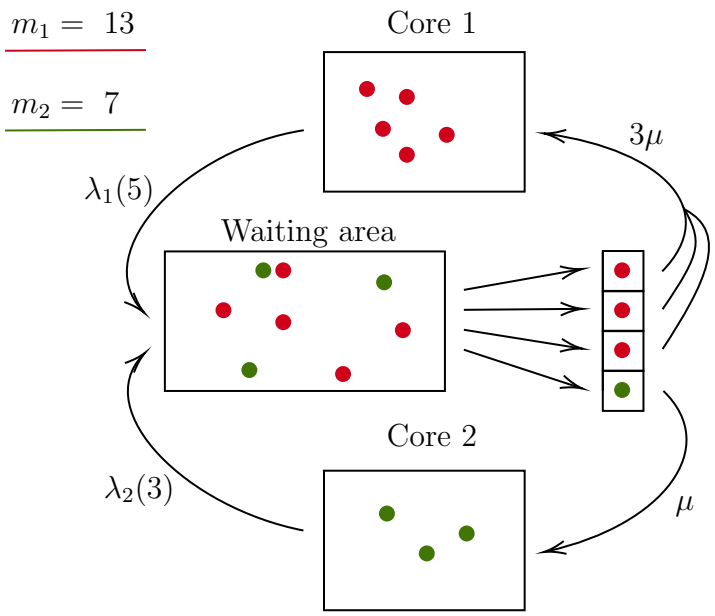

Figure 2: The overview of the Random Order of Service Memory Queue (ROS-MQ) model where the red and green dots are respectively the core1- and core2-requests. We have $B = 4$.

We have $m_i = C_i + W_i + E_i$ and $E \leq B$. We will once again consider the functions $\lambda_i(x) = \lambda x$ for all $i \leq n$. In equilibrium we must have an equal arrival and departure rate in all cores:

$$\lambda C_i = E_i \mu. \tag{2}$$

To calculate the equilibrium we need to distinguish between two cases. The first case is $W = 0$ (no requests in the waiting area) and the second case is $W > 0$. If $W > 0$ we must have $E = B$, because waiting requests imply that all execution slots are filled. If $W = 0$, then $E \leq B$. We will start by calculating the equilibrium in case of $W = 0$, because that is the easier case.

2.4.2 Empty waiting area

If $W = 0$ then $W_i = 0$ for all i, so $m_i = C_i + Q_i$ and the core-i-requests from core-i are directly send to the execution area. We can use this together with Equation 2 to obtain

$$\lambda(m_i - E_i) = E_i \mu \quad \Longrightarrow \quad E_i = \frac{\lambda m_i}{\mu + \lambda},$$

and we obtain

$$C_i = m_i - \frac{\lambda m_i}{\mu + \lambda} = \frac{\mu m_i}{\mu + \lambda}.$$

Since $Q \leq B$, we must have

$$\frac{\lambda \sum_{i=1}^{n} m_i}{\mu + \lambda} = \frac{\lambda m}{\mu + \lambda} \leq B.$$

2.4.3 Non-empty waiting area

If $W > 0$, then $E = B$. We can obtain that the arrival and departure rate of core-i-requests must be equal in core i and in the waiting area:

$$\begin{aligned} \lambda C_i &= E_i \mu && \text{(equal departure and arrival rate in core } i\text{)} \\ \lambda C_i &= \frac{W_i B \mu}{W} && \text{(equal arrival and departure rate in waiting area).} \end{aligned}$$

Furthermore, we must have $C_i + W_i + E_i = m_i$. This set of equations can be solved to obtain the solution

$$\begin{aligned} C_i &= \frac{m_i B \mu}{\lambda m} \\ W_i &= m_i - \frac{m_i(\mu + \lambda) B}{\lambda m} \\ E_i &= \frac{m_i B}{m}. \end{aligned}$$

Since $W > 0$, we must have

$$1 - \frac{(\mu + \lambda) B}{\lambda m} > 0 \quad \Longrightarrow \quad \frac{\lambda m}{\mu + \lambda} > B.$$

This means the two cases can be summarised to obtain

$$\begin{aligned} C_i &= \begin{cases} \frac{\mu m_i}{\mu + \lambda} & \text{if } \frac{\lambda m}{\mu + \lambda} \leq B \\ \frac{m_i B \mu}{\lambda m} & \text{otherwise} \end{cases} \\ W_i &= \begin{cases} 0 & \text{if } \frac{\lambda m}{\mu + \lambda} \leq B \\ m_i - \frac{m_i(\mu + \lambda) B}{\lambda m} & \text{otherwise} \end{cases} \\ E_i &= \begin{cases} \frac{\lambda m_i}{\mu + \lambda} & \text{if } \frac{\lambda m}{\mu + \lambda} \leq B \\ \frac{m_i B}{m} & \text{otherwise} \end{cases}. \end{aligned} \tag{3}$$

2.4.4 Similarities between BS and ROS-MQ model

Let $(C_i^{(B)}, M_i^{(B)})$ and $(C_i^{(R)}, W_i^{(R)}, E_i^{(R)})$ be the equilibrium of core-i-requests in respectively the BS and ROS-MQ model. These values can be found in Equations 1 and 3. For both the $\frac{\lambda m}{\mu+\lambda} \leq B$ and $\frac{\lambda m}{\mu+\lambda} > B$ case we have $C_i^{(B)} = C_i^{(R)}$ and $M_i^{(B)} = E_i^{(R)} + W_i^{(R)}$. Since $E_i^{(R)} + W_i^{(R)}$ is the total amount of core-i-requests in the memory, it seems that the ROS-MQ model is an extension of the BS model.

2.5 The FIFO Memory Queue model

The First In First Out Memory Queue (FIFO-MQ) model is illustrated in Figure 3. This model is very similar to the ROS-MQ model, but the difference is that there is a memory queue instead of a waiting area. The policy for the queue will be First In First Out (FIFO). In Figure 3 there are 3 red dots being served by the memory, so the core1-arrival rate is 3μ. Similarly, there is 1 green dot that is being served, so the core2-arrival rate is μ.

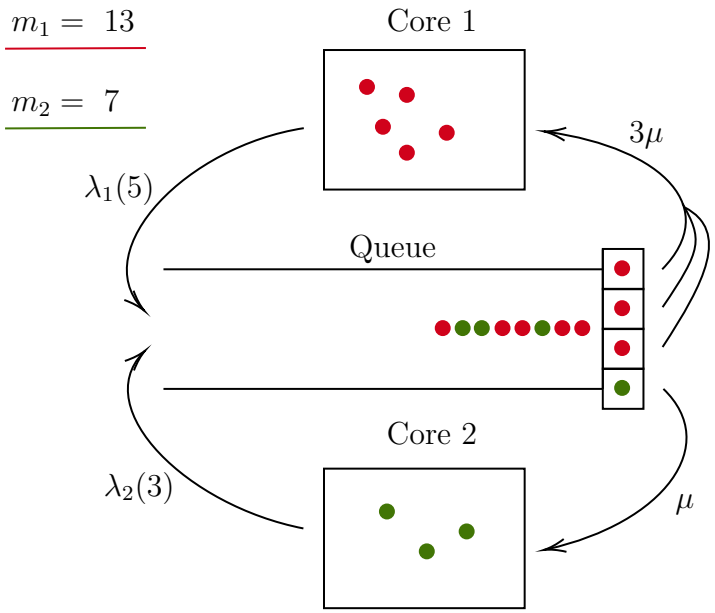

Figure 3: The overview of the FIFO Memory Queue (FIFO-MQ) model where the red and green dots are respectively the core1- and core2-requests. We have $B = 4$.