The shared memory model overview

Erik Leonards

March 13, 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 bandwidth 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 two possible ways of modelling the memory. You can either model it as a queue, where B number of memory requests can be handled at the same time, each with a service rate of μ. This will be called the Memory Queue (MQ) model. An alternative model is when the memory has B capacity, where each memory requests receives a service rate of

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

when there are m memory requests in the system. 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 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. 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. 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 $m_2 = 7$.

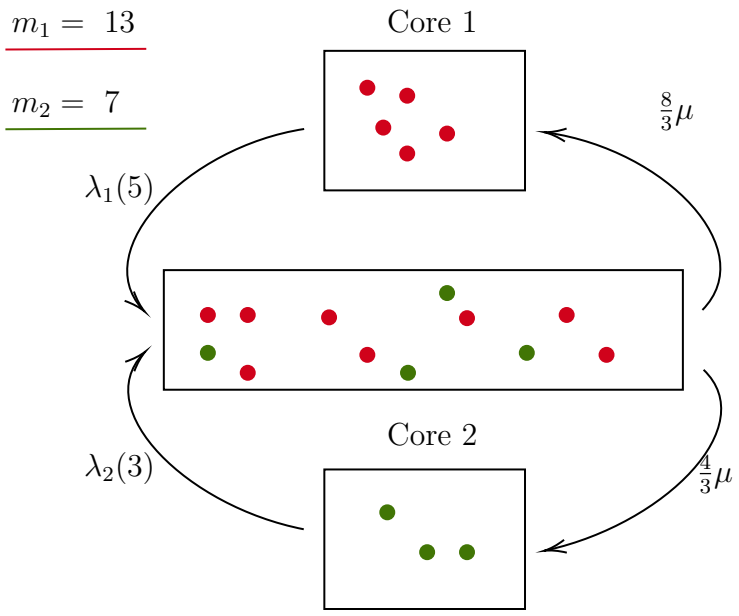

Figure 1: The overview of the bandwidth sharing model where the red and green dots are the requests. We have $B = 4$.

2.1.1 The departure of requests from the core

The requests in the core have just returned and are waiting to be sent back to the memory. If there are x requests in core i, then requests are send with

a rate of $\lambda_i(x)$, with $\lambda_i(x)$ increasing. This means that the rate is dependent on the number of requests 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.

2.1.2 The departure of requests from the core

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 $\frac{4}{3}\mu$.

2.2 The equilibrium

Suppose a system has n active cores and let C_i and M_i be the number of core-i-requests in respectively core i and the memory. If the core-i-memory rate $\lambda_i(C_i)$ is greater than the memory-core-i rate, then C_i will decrease. This means that $\lambda_i(C_i)$ will decrease. Furthermore, we have that M_i 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, 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 $i \leq n$.

In the equilibrium it must be the case for all $i \leq n$ that the arrival and

departure rate at the core are equal:

$$\begin{aligned}\lambda_i(C_i) &= \lambda C_i \\ &= \lambda(m_i - M_i) \\ &= \min\left\{M_i, M_i\frac{B}{\sum_{j=1}^n M_j}\right\}\mu \\ &= M_i \min\left\{1, \frac{B}{\sum_{j=1}^n M_j}\right\}\mu,\end{aligned}$$

which means that

$$\lambda(m_i - M_i) = \begin{cases} M_i\mu & \text{if } \sum_{j=1}^n M_i < B \\ M_i\frac{B}{\sum_{j=1}^n M_j}\mu & \text{otherwise}\end{cases}. \tag{1}$$

If $\sum_{i=1}^n M_i < B$, then all requests can run at full rate and there is no bandwidth sharing between requests. This means we have for all $i \leq n$ that

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

Since $\sum_{i=1}^n M_i < B$ we must have

$$\frac{\lambda}{\mu + \lambda}\sum_{i=1}^n M_i < B.$$

We will now consider the other case ($\sum_{i=1}^n M_i \geq B$) for a 2-core system and a n-core system.

2.2.1 A 2-core system

For the case of $M_1 + M_2 \geq B$ we have by Equation 1 that

$$\lambda(m_1 - M_1) = M_1\frac{B}{M_1 + M_2}\mu, \quad \lambda(m_2 - M_2) = M_2\frac{B}{M_1 + M_2}\mu.$$

These equations can be solved (with WolframAlpha) to obtain

$$M_1 = m_1 - \frac{\mu B m_1}{\lambda(m_1 + m_2)}, \quad M_2 = m_2 - \frac{\mu B m_2}{\lambda(m_1 + m_2)}.$$

Since $M_1 + M_2 \geq B$, we must have

$$m_1 + m_2 - \frac{\mu(m_1 + m_2)}{\lambda(m_1 + m_2)}B \geq B \quad \Longrightarrow \quad \frac{\lambda(m_1 + m_2)}{\mu + \lambda} \geq B.$$

The two cases can be summarised to obtain for $i = 1, 2$

$$M_i = \begin{cases} \frac{\lambda m_i}{\mu + \lambda} & \text{if } \frac{\lambda(m_1+m_2)}{\mu+\lambda} < B \\ m_i - \frac{\mu B m_i}{\lambda(m_1+m_2)} & \text{otherwise}\end{cases}.$$

2.2.2 A n-core system

Notice how the formula for the 2-core system has a certain structure. This means we can make an educated guess for the values of M_i. For the case of $\sum_{i=1}^{n} M_i \geq B$ we can make the ansatz for $i \leq n$

$$M_i = m_i - \frac{\mu B m_i}{\lambda \sum_{j=1}^{n} m_j} = m_i \left(1 - \frac{\mu B}{\lambda m}\right)$$

where $m \coloneqq \sum_{j=1}^{n} m_j$. The correctness of this formula can be checked by substituting it into Equation 1. We obtain for the left-hand side

$$\lambda \left(m_i - M_i\right) = \lambda \left(m_i - \left(m_i - \frac{\mu B m_i}{\lambda m}\right)\right) = \frac{\mu B m_i}{m}$$

The right-hand side is equal to

$$\begin{aligned} \frac{M_i B \mu}{\sum_{j=1}^{n} M_j} &= \frac{m_i \left(1 - \frac{\mu B}{\lambda m}\right) B \mu}{\sum_{j=1}^{n} m_j \left(1 - \frac{\mu B}{\lambda m}\right)} \\ &= \frac{\mu B m_i}{\sum_{j=1}^{n} m_j} \\ &= \frac{\mu B m_i}{m}, \end{aligned}$$

which is equal to the left-hand side. This implies that the ansatz is correct. Since $\sum_{i=1}^{n} M_i \geq B$, we must have

$$\begin{aligned} &\sum_{i=1}^{n} m_i \left(1 - \frac{\mu B}{\lambda m}\right) \geq B \\ \implies &\sum_{i=1}^{n} m_i \geq B + \frac{\mu B}{\lambda m} \sum_{i=1^n} m_i \\ \implies &m \geq B + \frac{\mu B}{\lambda} \\ \implies &m \geq B \frac{\mu + \lambda}{\lambda} \\ \implies &\frac{\lambda}{\mu + \lambda} \geq B. \end{aligned}$$

We can summarise the cases by

$$M_i = \begin{cases} \frac{\lambda m_i}{\mu + \lambda} & \text{if } \frac{\lambda}{\mu + \lambda} < B \\ m_i \left(1 - \frac{\mu B}{\lambda m}\right) & \text{otherwise} \end{cases}.$$

2.3 The memory queue model

The Memory Queue (MQ) model is illustrated in Figure 2. This model is very similar to the BS model, but the difference is that there is a memory queue in this model. The policy for the queue will be First In First Out (FIFO). Each request in the memory is now either in the queue, 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.

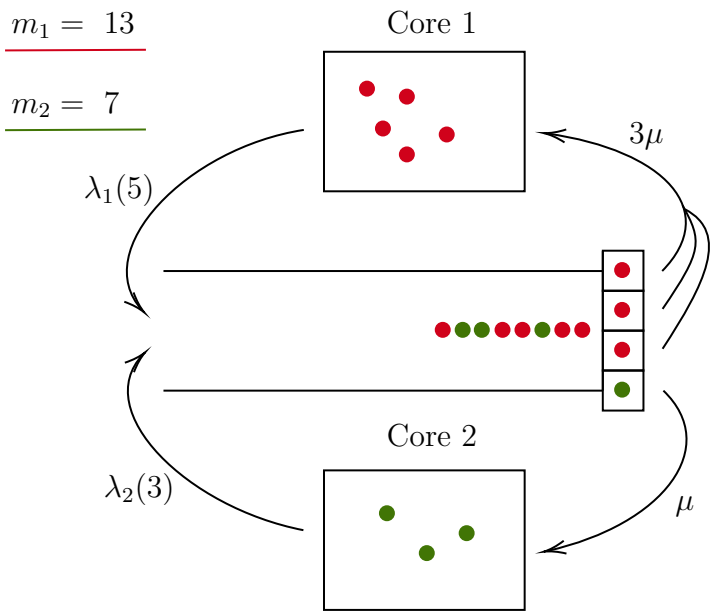

Figure 2: The overview of the Memory Queue (MQ) model where the red and green dots are respectively the core1- and core2-requests. We have $B = 4$.