BACHELOR

Efficient job assignment in cloud system

Lassche, E.C.

Award date:
2016

Efficient Job Assignment in Cloud System

Technische universiteit Eindhoven
Bachelor end project

16th June 2016

Author:
Eva Lassche, ID 0854229

Abstract

Assigning jobs to servers in the case of datacenters is something that would preferably be done very efficient. According to theoretical analysis one big super server would be the most efficient. Even though this would the lowest expected sojourn time this is not likely nor feasible. Between systems where the dispatcher sends the job to one of the N servers upon arrival of the job the Join-the-Shortest-Queue system is the most efficient, unfortunately this system has a very big overhead at the dispatcher, and is therefore not the solution for the problem either.
After Theoretical analysis and simulation it turns out to be efficient to work with tokens for servers that are idle (Join=the-Idle-Queue system), and combine this with a Power-of-d scheme in which d is low (1 or 2).

Contents

1 Introduction

Big internet sites such as Youtube and Google get millions of visits per minute. In each continent a datacenter is placed to handle the requests these visitors place. The companies behind those internet sites want the requests visitors place to be handled as quickly as possible.
In such a datacenter N servers are placed, where N is big, think about several thousands or more. We assume that the requests come in at a dispatcher as a Poisson process with rate $N\lambda$. The moment the requests enter the system we call them jobs. The dispatcher sends the jobs to one of the N servers following a scheme that belongs to a certain strategy. Each server has an exponentially distributed service time with rate μ. A visualisation of this system is given in figure 1.
How the dispatcher assigns the jobs to the servers can make the system more or less efficient. The first system described below is a super server, with one server with a service rate that is the combined rate of the N servers. In the second system the dispatcher places the job at a server that is not handling a job; if there are more than N jobs in the system, then the jobs wait at the dispatcher until one of the servers is free. In the last system the dispatcher picks one of the N servers randomly and assigns the job to that server. The queue arises at the servers therefore. This last system behaves as N different M/M/1 systems each with arrival rate λ and service rate μ.

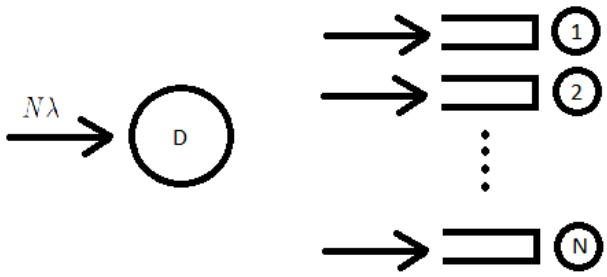

Figure 1: The visualisation of the queueing system

1.1 Research question

I am interested in comparing different schemes to dispatch jobs in a datacenter. Which of the schemes dispatches the jobs most evenly over the N servers, and which of the schemes does that without taking much time at the dispatcher?

1.2 Structure

To be able to answer these questions, we first analyse the three basic systems described above. After that we simulate four schemes to compare them to each other for different values of N, μ and λ. At the end we will say something about which of the systems is the most efficient.

2 Queueing Systems

2.1 M/M/1

The first and most basic queueing system is the so-called M/M/1 system. This system has 1 server. The jobs arrive as a Poisson process with rate $N\lambda$ and exponentially distributed service times with rate $N\mu$. Figure 2 shows the state diagram for this model. Because of the memoryless property of the Poisson process the total number of jobs in the system behaves as a Markov process.

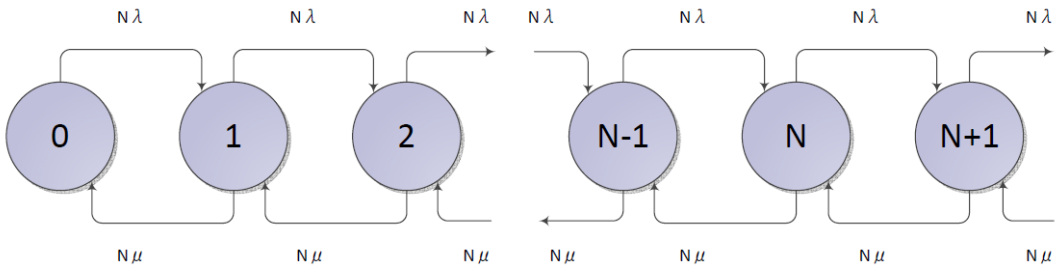

Figure 2: The state diagram for the M/M/1 system

Now define π_i as the steady-state probability that there are i jobs in the system. We assume $\lambda < \mu$ for stability of the system. For this model the following balance equations hold.

$$
\begin{aligned}
0 &= N\lambda\pi_{i-1} - (N\lambda + N\mu)\pi_i + N\mu\pi_{i+1} \\
0 &= -N\lambda\pi_0 + N\mu\pi_1 \\
\Rightarrow N\lambda\pi_{i-1} &= N\mu\pi_i \quad , \quad i = 1, 2, 3, \ldots \\
\Rightarrow \pi_i &= \frac{\lambda}{\mu}\pi_{i-1} = \ldots = \left(\frac{\lambda}{\mu}\right)^i \pi_0 \\
\Rightarrow \sum_{i=0}^{\infty} \pi_i &= \pi_0 \sum_{i=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^i = \frac{\pi_0}{1-\lambda\mu} = 1 \\
\Rightarrow \pi_0 &= 1 - \frac{\lambda}{\mu} \quad \Rightarrow \quad \pi_i = (1 - \frac{\lambda}{\mu}) \left(\frac{\lambda}{\mu}\right)^i
\end{aligned}
$$

If we now define $\rho = \frac{\lambda}{\mu}$ we can rewrite this into $\pi_i = (1-\rho)\rho^i$. We now see that the number of jobs in the system is geometrically distributed. This geometric distribution has parameter $\rho = \frac{\lambda}{\mu}$ and the expected value of this distribution is $\frac{\rho}{1-\rho}$. Thus we can say that the expected number of jobs in this system is $\frac{\rho}{1-\rho}$.

2.2 M/M/N

The second system to discuss is the so-called M/M/N system. The jobs arrive as a Poisson process with rate $N\lambda$ and exponentially distributed service times. To make it easier to understand this, a visualisation of this system is given in figure 3. Here it becomes clear that jobs will be finished at a higher rate when a higher state is reached. When there are more than N jobs in the system, all servers are busy, and the service rate will not grow anymore.
It is possible to derive the following equations from figure 3. Now define π_i as the steady-state probability that the system is in state i. We assume $\lambda < \mu$ for stability of the system. For $i < N$ the following balance equations hold.

$$0 = N\lambda\pi_{i-1} - (N\lambda + i\mu)\pi_i + (i+1)\mu\pi_{i+1}$$

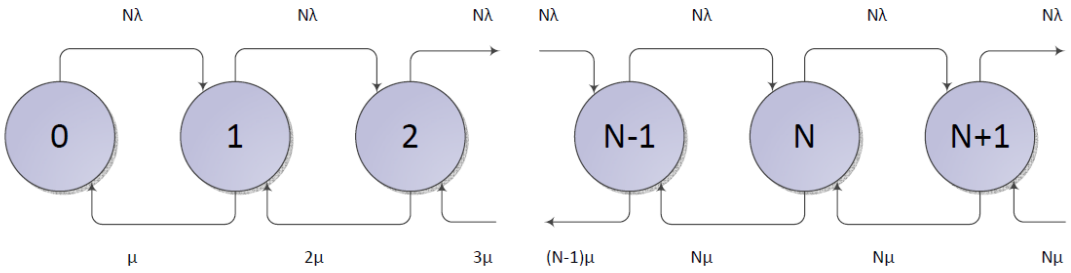

Figure 3: The state diagram for the M/M/N system

$$0 = -N\lambda\pi_0 + \mu\pi_1$$
$$\Rightarrow N\lambda\pi_{i-1} = i\mu\pi_i$$

And for $i \geq N$.

$$0 = N\lambda\pi_{i-1} - (N\lambda + N\mu)\pi_i + N\mu\pi_{i+1}$$
$$\Rightarrow \lambda\pi_{i-1} = \mu\pi_i$$

With these equations we can write π_1 in terms of π_0.

$$\pi_1 = \frac{N\lambda}{\mu}\pi_0$$

We can now do the same for π_2

$$N\lambda\pi_1 = 2\mu\pi_2 \Rightarrow \pi_2 = \frac{N\lambda}{2\mu}\pi_1$$

$$\pi_2 = \frac{1}{2}\left(\frac{N\lambda}{\mu}\right)^2\pi_0$$

In the same way it is possible to write π_3 and π_4

$$\pi_3 = \frac{1}{6}\left(\frac{N\lambda}{\mu}\right)^3\pi_0 \quad , \quad \pi_4 = \frac{1}{24}\left(\frac{N\lambda}{\mu}\right)^4\pi_0$$

Looking at these first four terms a pattern becomes visible. Using the balance equations given above it is possible to conclude that for $i < N$ the following holds:

$$\pi_i = \frac{1}{i!}\left(\frac{N\lambda}{\mu}\right)^i\pi_0$$

For $i \geq N$ we can use the fact that in the balanced state the following holds.

$$\lambda\pi_{i-1} = \mu\pi_i$$

This results in

$$\pi_i = \frac{\lambda}{\mu}\pi_{i-1} = \left(\frac{\lambda}{\mu}\right)^{i-N}\pi_N = \left(\frac{\lambda}{\mu}\right)^{i-N}\frac{1}{N!}\left(\frac{N\lambda}{\mu}\right)^N\pi_0 = \frac{N^N}{N!}\left(\frac{\lambda}{\mu}\right)^i\pi_0.$$

If we use $\rho = \frac{\lambda}{\mu}$ we can write $\pi_i = \frac{N^N}{N!}\rho^i\pi_0$. There is one last identity that can be used here to get results:

$$\sum_{i=0}^{\infty}\pi_i = \sum_{i=0}^{N-1}\frac{N^i}{i!}\rho^i\pi_0 + \sum_{i=N}^{\infty}\frac{N^N}{N!}\rho^i\pi_0 = 1$$

From this we can derive that

$$\pi_0 = \left(\sum_{i=0}^{N-1} \frac{N^i}{i!} \rho^i + \frac{(N)^N}{N!} \frac{1}{1-\rho} \right)^{-1},$$

$$\pi_i = \frac{1}{i!} \left(\frac{N\lambda}{\mu} \right)^i \pi_0 \quad \text{for } i < N$$

$$\pi_i = \frac{N^N}{N!} \left(\frac{\lambda}{\mu} \right)^i \left(\sum_{i=0}^{N-1} \frac{N^i}{i!} \rho^i + \frac{(N)^N}{N!} \frac{1}{1-\rho} \right)^{-1} \quad \text{for } i \geq N$$

2.3 N x M/M/1

The last system described here is the system where jobs arrive as a Poisson process with rate $N\lambda$. With chance $\frac{1}{N}$ they get assigned to one of the N servers. Every server processes jobs at exponential rate μ. This system is equivalent to N independent M/M/1 systems each with arrival rate λ and service rate μ. The arrival process at each of the servers is a Poisson process with rate $N\lambda \cdot \frac{1}{N} = \lambda$ because of the characteristics of Poisson processes, and each server processes jobs at the same exponential rate. This gives a state diagram as in figure 4 for each of the servers.

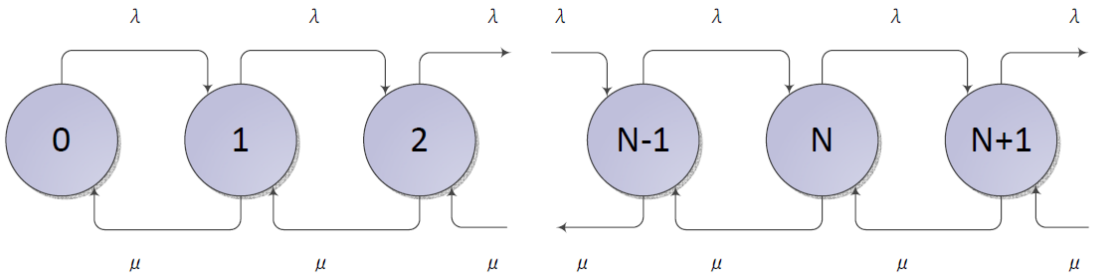

Figure 4: The state diagram for one server

In total the system states can be described as in figure 5, note here that this is not a Markov process. The smaller than or equal to signs are there because it is possible that when there are j jobs in the system, some of these jobs may be assigned to the same server, and less than j servers may be busy.

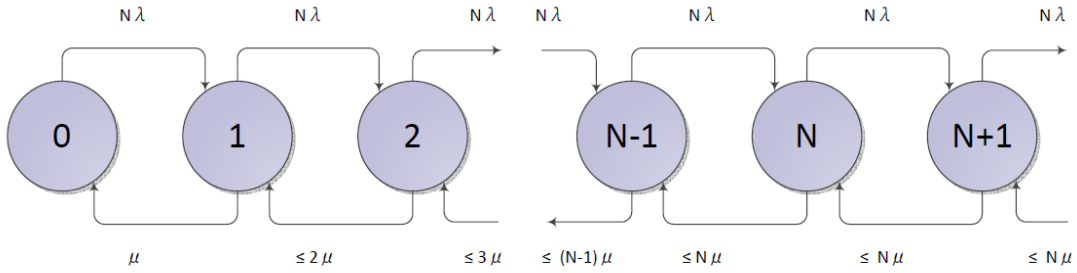

Figure 5: The visualisation for the N x M/M/1 system

For every M/M/1 system it is possible to determine the π_i, which is the probability that the system is in state i. For stability of the system we assume $\lambda < \mu$. These M/M/1 systems all have

the following balance equations.

$$0 = \lambda\pi_{i-1} - (\lambda+\mu)\pi_i + \mu\pi_{i+1}$$
$$0 = \lambda\pi_0 + \mu\pi_1$$
$$\Rightarrow \lambda\pi_{i-1} = \mu\pi_i \quad , \quad i = 1, 2, 3, \ldots$$

Using the same method as for the M/M/1 system in section 2.1 we can conclude that

$$\pi_i = \frac{\mu-\lambda}{\mu}\left(\frac{\lambda}{\mu}\right)^i = (1-\rho)\rho^i.$$

We see that this is equal to the π_i in the M/M/1 system, which is natural if you look at the systems. But if you take into account that the total arrival rate and total available service rate are equal in both cases it is rather strange. The M/M/1 system described above has an N times bigger arrival rate of jobs and an N times higher service rate. These N cancel out and give the same equation for π_i in the end. For the total N x M/M/1 system the $\pi_{(i_1,i_2,i_3,\ldots,i_N)} = \prod_{n=1}^{N}(1-\rho)\rho^{i_n} = (1-\rho)^N\rho^{\sum_{n=1}^{N} i_n}$ because all the N are independent. The total expected number of jobs in the system is easier to calculate because the N servers are stochastically identical. This gives

$$\mathbb{E}[\text{amount jobs in N x M/M/1}] = \mathbb{E}[N \cdot \mathbb{E}[\text{amount jobs in M/M/1}]] = N\frac{\rho}{1-\rho}.$$

2.4 Stochastic Comparison

We now want to know which of the three systems is the most efficient. To be able to say something about this we name the three systems. System I will be the M/M/1 system, system II will be the M/M/N system and system III will be the N x M/M/1 system. This means that there is a factor N difference between systems I and III.

2.4.1 Expected number of jobs

To be able to say something about the expected number of jobs in system II, we first define the probability that a job has to wait. A job has to wait when there are N or more jobs in the system, so

$$\prod_W = \pi_N + \pi_{N+1} + \pi_{N+2} + \ldots = \pi_N + \rho\pi_N + \rho^2\pi_N + \ldots = \sum_{i=0}^{\infty}\rho^i\pi_N$$
$$= \frac{\pi_N}{1-\rho} = \frac{N^N}{N!}\rho^N\left((1-\rho)\sum_{n=0}^{N-1}\frac{(N\rho)^n}{n!} + \frac{N^N}{N!}\right)^{-1}$$

Now we can derive the expected number of jobs in the whole system.

$$\mathbb{E}[L^{II}] = \sum_{n=0}^{\infty} n\pi_n = \sum_{n=0}^{N-1} n\frac{N^n}{n!}\rho^n\pi_0 + \sum_{n=N}^{\infty} n\frac{N^N}{N!}\rho^n\pi_0$$

Now we have two cases, ρ can go to 1 and ρ can go to 0. If ρ goes to zero, we are talking about the limit from above to zero. In this specific case the arrival rate will go to zero. This means that the system will most of the time be almost empty. Because of that for both π_0 and $\mathbb{E}[L^{II}]$ the part for $n \leq N$ is far more influential than the part where $n \geq N$. This leads to the following proposition, where we write $f(\rho) \sim g(\rho)$ for any two functions $f(\cdot)$ and $g(\cdot)$ if $\lim_{\rho\downarrow 0}\frac{f(\rho)}{g(\rho)} = 1$.

Proposition 2.1

$$\mathbb{E}[L^{II}] \sim N\rho$$

as $\rho \downarrow 0$ *and thus*

$$\lim_{\rho\downarrow 0}\frac{\mathbb{E}[L^{II}]}{\mathbb{E}[L^{III}]} = 1$$

Proof As $\rho \downarrow 0$

$$\pi_0 \sim \left(\sum_{i=0}^{N-1} \frac{N^i}{i!} \rho^i \right)^{-1}$$

$$\mathbb{E}[L^{II}] \sim \sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \pi_0$$

$$\mathbb{E}[L^{II}] \sim \sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \left(\sum_{i=0}^{N-1} \frac{N^i}{i!} \rho^i \right)^{-1}$$

because

$$\lim_{\rho \downarrow 0} \frac{\mathbb{E}[L^{II}]}{\sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \pi_0} = \lim_{\rho \downarrow 0} \frac{\sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \pi_0 + \sum_{n=N}^{\infty} n \frac{N^N}{N!} \rho^n \pi_0}{\sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \pi_0} = 1$$

now we can say that the following holds:

$$\left(\sum_{i=0}^{N-1} \frac{N^i}{i!} \rho^i \right)^{-1} \xrightarrow{\rho \downarrow 0} 1$$

$$\sum_{n=0}^{N-1} n \frac{N^n}{n!} \rho^n \xrightarrow{\rho \downarrow 0} N\rho$$

$$\lim_{\rho \downarrow 0} \frac{\mathbb{E}[L^{II}]}{\mathbb{E}[L^{III}]} = \lim_{\rho \downarrow 0} \frac{N\rho}{\frac{N\rho}{1-\rho}} = 1 \qquad \square$$

For the limit for ρ to 1 a similar thing holds. In this case the part where $n \leq N$ holds has very little influence. This means that for π_0 the following proposition is made, where we write $f(\rho) \sim g(\rho)$ for any two functions $f(\cdot)$ and $g(\cdot)$ if $\lim_{\rho \uparrow 0} \frac{f(\rho)}{g(\rho)} = 1$.

Proposition 2.2

$$\mathbb{E}[L^{II}] \sim \frac{1}{1-\rho}$$

as $\rho \uparrow 1$ *and thus*

$$\lim_{\rho \uparrow 1} \frac{\mathbb{E}[L^{II}]}{\mathbb{E}[L^{I}]} = 1$$

Proof As $\rho \uparrow 1$

$$\pi_0 \sim (1-\rho) \frac{N!}{N^N}$$

which implies that

$$\frac{\pi_0}{(1-\rho)\frac{N!}{N^N}} = 1$$

We can also say that

$$\mathbb{E}[L^{II}] \sim \sum_{n=N}^{\infty} n \frac{N^N}{N!} \rho^n \pi_0$$

For $\mathbb{E}[L^{II}]$ it then gives that:

$$\mathbb{E}[L^{II}] \sim \sum_{n=N}^{\infty} n \frac{N^N}{N!} \rho^n (1-\rho) \frac{N!}{N^N} = \sum_{n=N}^{\infty} n \rho^n (1-\rho) = (1-\rho) \sum_{n=N}^{\infty} n \rho^n$$

We know the following:

$$\sum_{n=N}^{\infty} n\rho^n = \sum_{n=N}^{\infty}(N+(n-N))\rho^n = \sum_{m=0}^{\infty}(N+m)\rho^{N+m} = \rho^N \sum_{m=0}^{\infty}(N+m)\rho^m = \rho^N(\frac{N}{1-\rho}+\frac{\rho}{(1-\rho)^2})$$

Using this we can say that for $\rho \uparrow 1$

$$\mathbb{E}[L^{II}] \sim (1-\rho)\rho^N(\frac{N}{1-\rho}+\frac{\rho}{(1-\rho)^2}) = \rho^N N + \rho^N \frac{\rho}{1-\rho} \xrightarrow{\rho\uparrow 1} \frac{1}{1-\rho}$$

$$\lim_{\rho\uparrow 1}\frac{\mathbb{E}[L^{II}]}{\mathbb{E}[L^I]} = \lim_{\rho\uparrow 1}\frac{\frac{1}{1-\rho}}{\frac{1}{1-\rho}} = 1 \qquad \square$$

We can now conclude that in terms of the expected number of jobs, and hence the mean delay system II behaves like system III when ρ is small. When ρ is large system II behaves like system I in terms of the expected number of jobs.

2.4.2 Coupling

To be able to compare the three different systems we use coupling. In this method we start with linking systems I and II. Looking at the state diagrams of these two systems, we see that the arrival rates are both $N\lambda$. The departure rates differ for $i < N$ and after that these are also equal. We create two counters for the number of jobs in the system. The first counter will be called $L^I(t)$ and will count the number of jobs in system I. Here the jobs arrive with rate $N\lambda$ and depart with rate $N\mu$. For system II we name the counter $L^{II}(t)$, the arrivals in this system are linked to the arrivals in system I. This means that when an arrival happens in system I, it will also happen in system II. The departures are also linked, only here it is a bit harder to link them because when $L^{II}(t) < N$ the arrival rate for system II is $L^{II}(t)\mu$ instead of $N\mu$. This means that when a departure happens in system I it will happen in system II with probability $\frac{L^{II}(t)}{N}$. For $L^{II}(t) \geq N$ a departure in system I means a departure in system II.
Mathematically we can describe this in the following way. We define $t_1^A, t_2^A, \ldots$ as the arrival times $\sim Poisson(N\lambda)$ and $t_1^D, t_2^D, \ldots$ as the departure times $\sim Poisson(N\mu)$. Δt will be used to add to the arrival or departure time, where $t_1^A + \Delta t$ is the time that the arrival has taken place. At t_1^A the arrival has not yet taken place. For the arrival events we can now say the following.

$$L^I(t_i^A + \Delta t) = L^I(t_i^A) + 1$$

$$L^{II}(t_i^A + \Delta t) = L^{II}(t_i^A) + 1$$

For the departures it becomes a bit more complicated. For $L^I(t)$ it holds that:

$$L^I(t_i^D + \Delta t) = \begin{cases} L^I(t_i^D) - 1 & \text{for } L^I(t_i^D) \geq 1 \\ 0 & \text{for } L^I(t_i^D) = 0 \end{cases}$$

If we now define $U \sim Unif[0,1]$ we can also give an expression for $L^{II}(t)$:

$$L^{II}(t_i^D + \Delta t) = \begin{cases} L^{II}(t_i^D) - 1 & \text{for } L^{II}(t_i^D) \geq N \\ L^{II}(t_i^D) - \mathbb{1}\{U \leq \frac{L^{II}(t_i^D)}{N}\} & \text{for } 1 \leq L^{II}(t_i^D) < N \\ 0 & \text{for } L^{II}(t_i^D) = 0 \end{cases}$$

Proposition 2.3 *If $L^I(0) \leq L^{II}(0)$, then $\{L^I(t)\}_{t\geq 0} \leq_{st} \{L^{II}(t)\}_{t\geq 0}$*

Proof If we now take time t for which it holds that $L^I(t) \leq L^{II}(t)$ we want to know what happens when an arrival or departure takes place. In case of an arrival at $t + t_1^A$ it holds that $L^I(t + t_1^A + \Delta t) = L^I(t) + 1$ and $L^{II}(t + t_1^A + \Delta t) = L^{II}(t) + 1$ and thus if $L^I(t) \leq L^{II}(t) \Rightarrow L^I(t + t_1^A + \Delta t) \leq L^{II}(t + t_1^A + \Delta t)$. In case of a departure at $t' = t + t_1^D$ two cases can occur:

(i) $L^I(t') = 0$

(ii) $L^I(t') \geq 1$

In case (i) there can not be a departure in system I, but in system II there could be if $L^{II}(t') \geq 1$. But $L^I(t' + \Delta t) = 0 \leq L^{II}(t' + \Delta t)$ because there can only be one departure from system II and there will not be a departure when $L^I(t') = 0 = L^{II}(t')$. In case (ii) $L^I(t' + \Delta t) = L^I(t') - 1$, while $L^{II}(t' + \Delta t) \geq L^{II}(t') - 1$ and $L^I(t') \leq L^{II}(t')$, hence $L^I(t' + \Delta t) \leq L^{II}(t' + \Delta t)$.
Either way $L^I(t' + \Delta t) \leq L^{II}(t' + \Delta t)$ holds for all $t \geq 0$, if $L^I(0) \leq L^{II}(0)$.
We can now conclude that $L^I(t) \leq L^{II}(t)$. To be able to link this to the M/M/1 and M/M/N systems we have to say something about the behavior of $L^I(t)$ and $L^{II}(t)$. $L^I(t)$ is constructed in such a way that it corresponds to the number of jobs in the M/M/1 system. $L^{II}(t)$ has the same arrivals as $L^I(t)$ but the departures depend on the number of jobs, which is the counter $L^{II}(t)$. When $L^{II}(t) = n < N$ a departure will take with rate $n\lambda$, which is the same as a departure with rate $N\lambda$ that happens with probability $\frac{n}{N}$. The last thing is what actually happens for $L^{II}(t)$. Thus $L^{II}(t)$ behaves in the same manner as the number of jobs in the M/M/N system. $\square$

If we now take $L^{III}(t)$ the counter for system III we can compare system III to system II. To be able to compare the two systems we have to go a little further here. We define $L_i^{III}(t)$ as the number of jobs at server i at time t. Now we can say that $L^{III}(t) = \sum_{i=1}^{N} L_i^{III}(t)$. We take $t_1^A, t_2^A, \ldots$ as the arrival times $\sim Poisson(N\lambda)$ and $t_1^D, t_2^D, \ldots$ as the departure times $\sim Poisson(N\mu)$. For the arrival events we can say the following:

$$L^{II}(t_i^A + \Delta t) = L^{II}(t_i^A) + 1$$
$$L^{III}(t_i^A + \Delta t) = L^{III}(t_i^A) + 1$$

The departures will be a bit harder to handle. If we introduce $U \sim Unif[0, 1]$ we can say the following for system II:

$$L^{II}(t_i^D + \Delta t) = \begin{cases} L^{II}(t_i^D) - 1 & \text{for } L^{II}(t_i^D) \geq N \\ L^{II}(t_i^D) - \mathbb{1}\{U \leq \frac{L^{II}(t_i^D)}{N}\} & \text{for } 1 \leq L^{II}(t_i^D) < N \\ 0 & \text{for } L^{II}(t_i^D) = 0 \end{cases}$$

Now we want to say something about the departures for system III. To be able to do so we define $\hat{L}^{III}(t) = \sum_{i=1}^{N} \mathbb{1}\{L_i^{III}(t) \geq 1\}$. Now we can say:

$$L^{III}(t_i^D + \Delta t) = \begin{cases} L^{III}(t_i^D) - \mathbb{1}\{U \leq \frac{\hat{L}^{III}(t_i^D)}{N}\} & \text{for } L^{III}(t_i^D) \geq 1 \\ 0 & \text{for } L^{III}(t_i^D) = 0 \end{cases}$$

Proposition 2.4 *If $L^{II}(0) \leq L^{III}(0)$, then $\{L^{II}(t)\}_{t\geq 0} \leq_{st} \{L^{III}(t)\}_{t\geq 0}$*

Proof If we now take time t for which it holds that $L^{II}(t) \leq L^{III}(t)$ we want to know what happens when an arrival or departure takes place. In case of an arrival at $t + t_1^A$ it holds that $L^{II}(t + t_1^A + \Delta t) = L^{II}(t) + 1$ and $L^{III}(t + t_1^A + \Delta t) = L^{III}(t) + 1$ and thus if $L^{II}(t) \leq L^{III}(t) \Rightarrow L^{II}(t + t_1^A + \Delta t) \leq L^{III}(t + t_1^A + \Delta t)$. In case of a departure at $t' = t + t_1^D$ we can again distinguish two cases:

(i) $L^{II}(t') = 0$

(ii) $L^{II}(t') \geq 1$

In case (i) there will be not be a departure from system II. If $L^{III}(t') > 0$ there could be a departure in system III, but there will only be one departure thus if $L^{II}(t') \leq L^{III}(t') \Rightarrow 0 = L^{II}(t' + \Delta t) \leq L^{III}(t' + \Delta t)$. In case (ii) there could be a departure in system II if $U \leq \frac{L^{II}(t')}{N}$ it follows that $L^{II}(t' + \Delta t) = L^{II}(t') - 1$. If $U \leq \frac{\hat{L}^{III}(t')}{N}$ there will also be a departure in

system III. The most critical situation to look at here is when $L^{II}(t') = L^{III}(t')$. In this case it holds that $\hat{L}^{III}(t') \leq L^{II}(t')$, thus there can not be an departure in system III when there is no departure in system II. If $L^{II}(t') > L^{III}(t')$ it could be the case that $\hat{L}^{III}(t') > L^{II}(t')$, but it is no problem if there is a departure in system III and not in system II here. So if $L^{II}(t') \leq L^{III}(t') \Rightarrow L^{II}(t' + \Delta t) \leq L^{III}(t' + \Delta t)$.
The last thing to do is matching $L^{III}(t)$ to system III. For the arrivals it is easy to see that it holds, for the departures you look at how many servers are having jobs to determine the departure rate. This is because the jobs are randomly assigned to one of the N servers, so you can not say that all the j servers are busy when there are j jobs in the system. To solve this we count how many servers are busy, which makes $L^{III}(t)$ fit to system III. □

2.5 More advanced schemes

We have now analysed the three systems above, and introduce a few additional systems.

2.5.1 Join-the-Idle-Queue

System IV that we introduce is the so called Join-the-Idle-Queue system. Here jobs arrive at the dispatcher as a Poisson process with rate $N\lambda$. There are N servers that each have an exponentially distributed service time with rate μ. The dispatcher has tokens from servers that do not have any jobs to serve at that moment. When a job arrives and there are no tokens available, the dispatcher randomly chooses one of the N servers and sends the job there. [7] [3]
Looking at the π_i, the steady-state probability that the system is in state i, where X is the number of servers that have jobs in their queue, we can say the following.

$$N\lambda\pi_n = X\mu\pi_{n+1}$$

The problem here is that X can not be expressed in terms of n or N. The approach we used before does not work here, because we will need information about every server individually which we do not have at the moment. Further more would information about every server make the equations N-dimensional, which makes it N times harder to solve the equations.
We however can say something about the system using coupling. We therefore introduce the counter $L^{IV}(t)$ for system IV. We also define $L_i^{IV}(t)$ as the number of jobs at server i at time t. It holds that $L^{IV}(t) = \sum_{i=1}^{N} L_i^{IV}(t)$. We again take $t_1^A, t_2^A, \ldots$ as the arrival times $\sim Poisson(N\lambda)$ and $t_1^D, t_2^D, \ldots$ as the departure times $\sim Poisson(N\mu)$. For the arrival events we can say the following:

$$L^{IV}(t_i^A + \Delta t) = L^{IV}(t_i^A) + 1$$

The departures will be a bit harder to handle. If we introduce $U \sim Unif[0,1]$ and $\hat{L}^{IV}(t) = \sum_{i=1}^{N} \mathbb{1}\{L_i^{IV}(t) \geq 1\}$ we can say:

$$L^{IV}(t_i^D + \Delta t) = \begin{cases} L^{IV}(t_i^D) - \mathbb{1}\{U \leq \frac{\hat{L}^{IV}(t_i^D)}{N}\} & \text{for } L^{IV}(t_i^D) \geq 1 \\ 0 & \text{for } L^{IV}(t_i^D) = 0 \end{cases}$$

Proposition 2.5 *If $L^{II}(0) \leq L^{IV}(0)$, then $\{L^{II}(t)\}_{t\geq 0} \leq_{st} \{L^{IV}(t)\}_{t\geq 0}$.*

Proof Using the same argument as for the comparison between systems II and III, we can conclude that $L^{II}(t) \leq L^{IV}(t)$, which implies that $L^{I}(t) \leq L^{IV}(t)$ because we also proved that $L^{I}(t) \leq L^{II}(t)$. □

Conjecture 2.6 *If $L^{IV}(0) \leq L^{III}(0)$, then $\{L^{IV}(t)\}_{t\geq 0} \leq_{st} \{L^{III}(t)\}_{t\geq 0}$.*

Comparing system IV to system III is a bit harder. The only difference between the two systems are the counters $\hat{L}^{III}(t)$ and $\hat{L}^{IV}(t)$. For the first N jobs, starting in an empty system, we can

say that $\hat{L}^{IV}(t) \geq \hat{L}^{III}(t)$ but in general no such equation holds. We do however suspect that this will hold in general, because we know that the dispatcher will send a job to an empty server, if such a server exists. This means that in the long term in system IV more servers will be used than in system III, even though the mean is the same. This makes us think that $L^{IV}(t) \leq L^{III}(t)$, but proving this conjecture is an interesting challenge for further research.

2.5.2 Join-the-Shortest-Queue and power-of-d scheme

System V is the Join-the-Shortest-Queue system. Jobs arrive at the dispatcher as a Poisson process with rate $N\lambda$. There are N servers that each have an exponentially distributed service time with rate μ. Using the power-of-d scheme, the dispatcher selects d servers at random and sends the job to the server that has the fewest jobs. How larger d, how better the jobs get distributed over the servers, but a larger d also gives a bigger overhead of the dispatcher. For $d = N$, the power-of-d scheme corresponds to the ordinary Join-the-Shortest-Queue (JSQ) policy.
We can again say something about the system using coupling. We therefore introduce the counter $L^V(t)$ for system V. We also define $L_i^V(t)$ as the number of jobs at server i at time t. It holds that $L^V(t) = \sum_{i=1}^{N} L_i^V(t)$. We again take $t_1^A, t_2^A, \ldots$ as the arrival times $\sim Poisson(N\lambda)$ and $t_1^D, t_2^D, \ldots$ as the departure times $\sim Poisson(N\mu)$. For the arrival events we can say the following:

$$L^V(t_i^A + \Delta t) = L^V(t_i^A) + 1$$

The departures will be a bit harder to handle. If we introduce $U \sim Unif[0,1]$ and $\hat{L}^V(t) = \sum_{i=1}^{N} \mathbb{1}\{L_i^V(t) \geq 1\}$ we can say:

$$L^V(t_i^D + \Delta t) = \begin{cases} L^V(t_i^D) - \mathbb{1}\{U \leq \frac{\hat{L}^V(t_i^D)}{N}\} & \text{for } L^V(t_i^D) \geq 1 \\ 0 & \text{for } L^V(t_i^D) = 0 \end{cases}$$

Proposition 2.7 *If $L^{II}(0) \leq L^V(0)$, then $\{L^{II}(t)\}_{t\geq 0} \leq_{st} \{L^V(t)\}_{t\geq 0}$.*

Proof Using the same argument as for the comparison between systems II and III, we can conclude that $L^{II}(t) \leq L^V(t)$, which implies that $L^I(t) \leq L^V(t)$ because we also proved that $L^I(t) \leq L^{II}(t)$. □

Comparing system V to system IV or III is a bit harder. The only difference between the three systems are the counters $\hat{L}^{III}(t)$, $\hat{L}^{IV}(t)$ and $\hat{L}^V(t)$.

Proposition 2.8 *If $d = N$ and $L^V(0) \leq L^{III}(0), L^{IV}(0)$ then $\{L^V(t)\}_{t\geq 0} \leq_{st} \{L^{III}(t)\}_{t\geq 0}$ and $\{L^V(t)\}_{t\geq 0} \leq_{st} \{L^{IV}(t)\}_{t\geq 0}$.*

Proof In 1978 Weber [5] proofed this already, after Winston [6] started the proof in 1977, Ephremides, Varavya and Walrand finalized this in 1980. These papers used a differend method than Towsley [4] who proved that $\{L^V(t)\}_{t\geq 0}$ is stochastically smaller than the number of jobs under any assignment policy that does not have advance knowledge of the service times. Towsley used sophisticated N-dimensional methods to prove this, and for that the proof is not further included in this document. □

2.5.3 Join-the-Idle-Queue combined with power-of-2

System VI will be the last system in this document. Here jobs arrive at the dispatcher as a Poisson process with rate $N\lambda$. There are N servers that each have an exponentially distributed service time with rate μ. The dispatcher has tokens from servers that do not have any jobs to serve at that moment. When a job arrives and there are no tokens available, the dispatcher randomly chooses two of the N servers and sends the job to the server that has the smallest queue. The power-of-2 system is described by Mitzenmacher [1] and Vvedenskaya, Dobrushin and Karpelevich [2].
Theoretically analysing this system will not be done because this will become way to complex. We therefore will introduce a simulation to be able to say something about the way the different systems

3 Simulation

To be able to say something about the mean amount of jobs in the last three systems, we use simulation. In these simulations we count the amount of jobs in the system, and at the end give the mean amount of jobs for that system. For the purpose of the simulation we will not take into account how much time the dispatcher takes to select a server for the job.
The four different simulations have a lot in common. Every system generates arrivals according to a Poisson(Nλ) process. This means that the inter arrival times are exponentially(Nλ) distributed. To start the system we generate the first arrival, after that a new arrival is generated after every arrival. For the departures it holds in general that a new departure will be scheduled when a jobs arrives at an idle server, or when a departure takes place at a server that has one or more jobs waiting in its queue. Departures are scheduled in an exponential(μ) time after the moment the service starts. In case of a system with tokens, a token will be send to the token queue when a departure takes place at the moment the queue of the server is empty.

3.1 Random

The first system to simulate will be system III. This is the system where the jobs are randomly send to one of the N servers. For this simulation the following pseudocode can be made.

Algorithm 1 System III

```
1:  procedure SIMULATE(maxTime, N, μ, λ)   ▷ maxTime, μ and λ are doubles, N is a integer
2:      Initialization                                          ▷ Results, FES, N Servers
3:      Schedule the first arrival in the system
4:      while t < maxTime do                                                ▷ Main loop
5:          e ← nextEvent      ▷ This is the event with the lowest time of all scheduled events
6:          t ← e.getTime
7:          if e.getType = arrival then                                  ▷ Event is an arrival
8:              add one to the counter for the number of jobs in the system
9:              Generate a random double a between 0 and 1, set integer b to 0
10:             while a > b/N do
11:                 b = b + 1
12:             Send the job to server number b-1
13:             if  amount of jobs at server b-1 is 1 then
14:                 Schedule the departure for the job at time t + exponential(μ)
15:             Schedule the next arrival at time t + exponential(λN)
16:         else                                                    ▷ Event is a departure
17:             Subtract one from the counter for the number of jobs in the system
18:             Subtract one from the amount of jobs at server
19:             if amount of jobs at server is bigger or equal to 1 then
20:                 Schedule the next departure at that server for the next job in line
21:     return results
```

In this system the dispatcher randomly selects one of the N servers. After selecting one of the servers, the departure of this job will be scheduled in case this is the only job at the server. When a departure happens the system will look if there are still jobs at the server after the departure. In this case the departure of the first job in the queue will be scheduled.

3.2 Join-the-Idle-Queue

The second system to simulate will be system IV. This is the system where the dispatcher has tokens from servers which do not have any jobs to serve at that moment. The dispatcher will send a job to an empty server if possible. When there are no tokens available the server wil randomly select one of the N servers and send the job there. The pseudcode for this system can be found here.

Algorithm 2 System IV

```
procedure SIMULATE(maxTime, N, μ, λ)   ▷ maxTime, μ and λ are doubles, N is a integer
    Initialization                      ▷ Results, FES, N Servers, token Queue
    Schedule the first arrival in the system
    while t < maxTime do                                              ▷ Main loop
        e ← nextEvent        ▷ This is the event with the lowest time of all scheduled events
        t ← e.getTime
        if e.getType = arrival then                                   ▷ Event is an arrival
            add one to the counter for the number of jobs in the system
            if the size of the token Queue is bigger or equal to 1 then
                Send the job to the server that belongs to the first token in the token Queue
                Schedule a departure for that job at time t + exponential(μ)
            else
                Generate a random double a between 0 and 1, set integer b to 0
                while a > b/N do
                    b = b + 1
                Send the job to server number b-1
            Schedule the next arrival at time t + exponential(λN)
        else                                                          ▷ Event is a departure
            Subtract one from the counter for the number of jobs in the system
            Subtract one from the amount of jobs at server
            if amount of jobs at server is bigger or equal to 1 then
                Schedule the next departure at that server for the next job in line
            else                          ▷ There are no jobs at the server anymore
                Send a token to the token Queue
    return results
```

We see here that when an arrival happens the dispatcher looks at the token queue first. If there are tokens the dispatcher sends the job to the server belonging to the first token in the queue and schedules a departure for this job. If there are no tokens the dispatcher will send the job to a randomly selected server. When a departure happens the system will look at the queue of the server. If the queue is empty after the departure a token will be sent to the token queue. Otherwise a new departure will be scheduled for the first job in the queue of the server.

3.3 Join-the-Shortest-Queue

The third system to simulate will be system V. This is the system where the dispatcher sends the job to the server that has the least jobs at that specific moment. Comparing all those N servers will in practice take way too much time to be efficient, but that will be ignored for the purpose of this simulation. The pseudcode for this system can be found here.

Algorithm 3 System V

```
1:  procedure SIMULATE(maxTime, N, μ, λ)   ▷ maxTime, μ and λ are doubles, N is a integer
2:      Initialization                                          ▷ Results, FES, N Servers
3:      Schedule the first arrival in the system
4:      while t < maxTime do                                                ▷ Main loop
5:          e ← nextEvent      ▷ This is the event with the lowest time of all scheduled events
6:          t ← e.getTime
7:          if e.getType = arrival then                                 ▷ Event is an arrival
8:              add one to the counter for the number of jobs in the system
9:              Select the server that has the least job in it's Queue
10:             Send the job to the selected server
11:             if amount of jobs at selected server is 1 then
12:                 Schedule the departure for the job at time t + exponential(μ)
13:             Schedule the next arrival at time t + exponential(λN)
14:         else                                                    ▷ Event is a departure
15:             Subtract one from the counter for the number of jobs in the system
16:             Subtract one from the amount of jobs at server
17:             if amount of jobs at server is bigger or equal to 1 then
18:                 Schedule the next departure at that server for the next job in line
19:     return results
```

Here the dispatcher selects the server with the shortest queue in case an arrival happens. When this is the only job assigned to that server the departure of this job will be scheduled. When a departure happens the job will leave the system. When the server has jobs in its queue after the job left, the departure of the first job in the queue will be scheduled. Otherwise nothing will happen.

3.4 Join-the-Idle-Queue combined with power-of-2

The last system to simulate will be the Join-the-Idle-Queue combined with Power-of-2 system. For the continuation of the counting we will call this system VI. In this system the dispatcher will use tokens from servers that do not have any jobs to help at that moment. If there are no tokens available the server will randomly pick two servers and send the job to the server that has the smallest queue. The pseudcode for this system can be found here.

Algorithm 4 System VI

```
1:  procedure SIMULATE(maxTime, N, μ, λ)   ▷ maxTime, μ and λ are doubles, N is a integer
2:      Initialization                      ▷ Results, FES, N Servers, token Queue
3:      Schedule the first arrival in the system
4:      while t < maxTime do                                              ▷ Main loop
5:          e ← nextEvent   ▷ This is the event with the lowest time of all scheduled events
6:          t ← e.getTime
7:          if e.getType = arrival then                                   ▷ Event is an arrival
8:              add one to the counter for the number of jobs in the system
9:              if the size of the token Queue is bigger or equal to 1 then
10:                 Send the job to the server that belongs to the first token in the token Queue
11:                 Schedule a departure for that job at time t + exponential(μ)
12:             else
13:                 Randomly pick two of the N servers
14:                 Send the job to the server that has the smallest Queue
15:             Schedule the next arrival at time t + exponential(λN)
16:         else                                                          ▷ Event is a departure
17:             Subtract one from the counter for the number of jobs in the system
18:             Subtract one from the amount of jobs at server
19:             if amount of jobs at server is bigger or equal to 1 then
20:                 Schedule the next departure at that server for the next job in line
21:             else                          ▷ There are no jobs at the server anymore
22:                 Send token to the token Queue
23:     return results
```

When an arrival takes place the dispatcher here will look at the token queue first. If there are tokens in this queue the dispatcher will send the job to the server belonging to the first token in the token queue. The departure for this job will also be scheduled. If there are no tokens in the token queue, the dispatcher will randomly select two servers and send the job to the server with the shortest queue. When a departure takes place the system will look at the queue of the server. If this queue is empty after the departure a token will be sent to the token queue. Otherwise the departure of the first job in the queue will be scheduled.

4 Results

To be able to compare the simulated systems we have to run the simulation for some N, λ and μ. To keep the system stable we have to keep in mind that $\lambda < \mu$ has to hold. We begin with taking $N = 100$, $\mu = 0.1$ seconds and λ from 0.0001 to 0.001 seconds in steps of 0.0001. We run the simulation until the time of the next event is above 1000000000 seconds after the beginning of the simulation. This gives a load of 0.1% to 1%. This gives the following results for system III, when we use the theoretical result.

λ	0.0001	0.0002	0.0003	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.001
$\mathbb{E}[L^{III}]$	0.1001	0.2004	0.3009	0.4016	0.5025	0.6036	0.7049	0.8065	0.9082	1.0101

Table 1: The expected amount of jobs in the system

If we now plot this against the results from the simulation, we see that all the simulation results and the theoretical results are very close together. Which is not so weird for a system that has a load of maximal 1%.

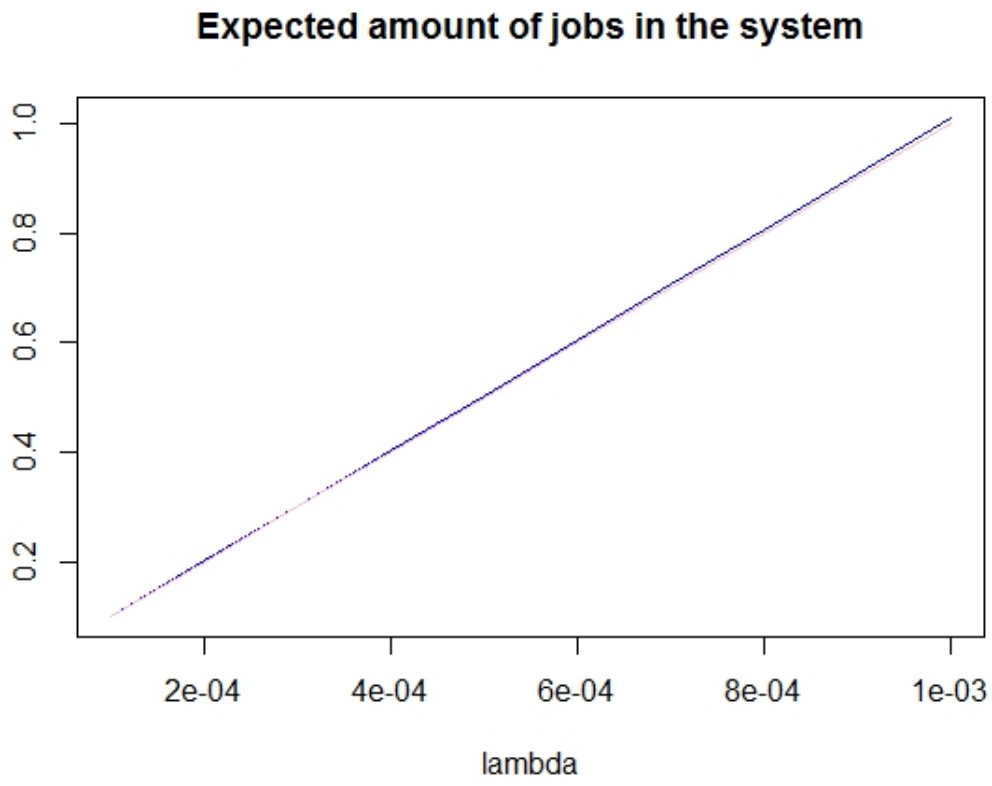

Figure 6: The expected amount of jobs in the system

To give a better picture of the four simulated systems we need a table with the results. This table is given is table 2, where we see that all the results are very close together, with no significant differences between the different systems.

λ	0.0001	0.0002	0.0003	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.001
$\mathbb{E}[L^{III}]$	0.1002	0.2003	0.3010	0.4017	0.5026	0.6037	0.7052	0.8064	0.9084	1.0107
$\mathbb{E}[L^{IV}]$	0.1000	0.1999	0.3000	0.4000	0.4999	0.6000	0.7000	0.7998	0.8998	1.0001
$\mathbb{E}[L^{V}]$	0.1000	0.2000	0.3000	0.4000	0.5000	0.6000	0.7000	0.7999	0.9000	1.0000
$\mathbb{E}[L^{VI}]$	0.1000	0.2001	0.2999	0.4000	0.4999	0.6002	0.7000	0.8001	0.8998	0.9998

Table 2: The simulated amount of jobs in the systems

To be able te generate a precise enough result we run the simulation again for the same N, λ and μ, but this time we calculate the mean amount of jobs in the system in 100 runs of 100000000 seconds. This gives the results given is table 3.

λ	0.0001	0.0002	0.0003	0.0004	0.0005
$\mathbb{E}[L^{III}]$	0.100122	0.200433	0.300918	0.401644	0.502512
$\mathbb{E}[L^{IV}]$	0.099994	0.199974	0.300063	0.400005	0.499932
$\mathbb{E}[L^{V}]$	0.099999	0.200012	0.299971	0.400002	0.500016
$\mathbb{E}[L^{VI}]$	0.099994	0.200020	0.300031	0.399978	0.500013
λ	0.0006	0.0007	0.0008	0.0009	0.001
$\mathbb{E}[L^{III}]$	0.603704	0.704957	0.806560	0.908318	1.010324
$\mathbb{E}[L^{IV}]$	0.599971	0.700065	0.799932	0.899764	1.000139
$\mathbb{E}[L^{V}]$	0.600023	0.700062	0.799989	0.900000	1.000068
$\mathbb{E}[L^{VI}]$	0.599979	0.699940	0.799466	0.899825	0.999828

Table 3: The simulated amount of jobs in the systems

Because we only have around 1% load of the system above, it is not weird that we do not see any difference between the different systems. To be able to see a difference in the performance of the different systems we therefore look at a load of $10, 50, 90, 95$ and 99%. This mean that for $N = 100$ and $\mu = 0.1$ we take $\lambda = 0.01, 0.05, 0.09, 0.095$ and 0.099 and a maxtime of 10000000 seconds. The results of these simulations are shown in table 4.

λ	0.01	0.05	0.09	0.095	0.99
Theoretical system III	11.11111	100	900	1900	9900
$\mathbb{E}[L^{III}]$	11.13602	954.5128	400121.0	451951.6	498859.2
$\mathbb{E}[L^{IV}]$	9.997544	50.01641	96.65752	120.7951	35419.73
$\mathbb{E}[L^{V}]$	10.00838	49.98874	96.01128	116.0047	213.1996
$\mathbb{E}[L^{VI}]$	10.00054	50.01180	96.13848	117.6119	235.6223

Table 4: The simulated amount of jobs in the systems

4.1 conclusion

For the higher λ we finally generated results that could give us a reason to conclude something. First of all we see that that system III explodes, and is by far the least efficiënt system of the four. Secondly we see that system V has the best results, this is what we expected, as it should be the best system of the four. System IV has good simulation results up until the point where the load of the system comes above the 95%. This is unfortunate, but makes it easier to conclude that system VI preforms better than system IV. System VI stays very close to the optimal system V which is good to see, because system VI has a significant smaller overhead than system V at the dispatcher.

5 Conclusion

Looking at the results above we can say that in the ideal situation we would have one super server, that could work at the speed of all the N servers added together. Unfortunately that is cost-prohibitive not feasible. The M/M/N system comes second in the ranking, but here we have a queue at the dispatcher, where jobs wait until one of the servers is idle. This cannot be easily implemented so the fact that this system is best is not useful either.
Between systems III, IV, V and VI system III ends on the bottom of the list. The third place is for system IV. Which is mostly because of the random assignment at the moment that there are no tokens available. System VI ends second in the ranking, as we expected. System V performs best, which was already proven by Towsley [4] in fact. Unfortunately this system has a very big overhead at the dispatcher.
All together we can conclude that combining the Join-the-Idle-Queue and Power-of-2 systems works very well. This system will have a relatively low overhead at the dispatcher and at same time will perform very well overall.

References

[1] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[2] F. Karpelevich N. Vvedenskaya, R. Dobrushin. Queueing systems with selection of the shortest of the two queues: An asymptotic approach. *Problems of Information Transmission*, 32(1):20–34, 1996.

[3] A.L. Stolyar. Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Systems*, 80:341–361, 2015.

[4] D. Towsley. Application of majorization to control problems in queueing systems. *Scheduling Theory and its Applications ed. P. Chrtienne, E.G. Coffman, J.K. Lenstra, and Z. Liu.*, 1995.

[5] R.R. Weber. On the optimal assignment of customers to parallel queues. *Journal of Applied Probability*, 15:406–413, 1978.

[6] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189, 1977.

[7] G. Kliot G. Geller A. Larus J.R. Greenberg Y. Lu, Q. Xie and A. Greenberg. Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation*, 68:1056–1071, 2011.