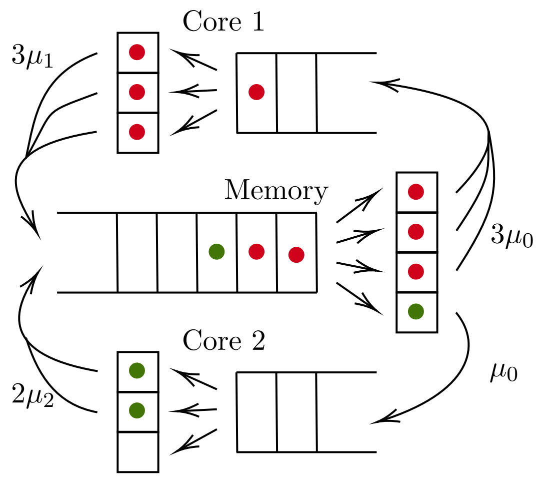
Processor modelling using queueing theory

Erik Leonards

May 21, 2023

Bachelor thesis Mathematics and Computer Science

Supervisor: prof. dr. (Rude)sindo Núñez Queija, prof. dr. Anuj Pathania



Informatics Institute
Korteweg-de Vries Institute for Mathematics
Faculty of Sciences
University of Amsterdam

Abstract

Schrijf een samenvatting van hoogstens een halve bladzijde waarin je kort uitlegt wat je hebt gedaan. De samenvatting schrijf je als laatste. Je mag er vanuit gaan dat je docent de lezer is.

Title: Processor modelling using queueing theory
Authors: Erik Leonards, erik.leonards@student.uva.nl, 13170007
Supervisors: prof. dr. (Rude)sindo Núñez Queija, prof. dr. Anuj Pathania
Second grader: dr. Florian Speelman, prof. dr.
End date: May 21, 2023

Informatics Institute
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.ivi.uva.nl`

Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.kdvi.uva.nl`

Contents

1. Introduction

Vertel iets over je artikel, noem je vraagstelling. Richt je tekst op medestudenten. (Richtlijn 2 bladzijden).

2. A computer system

A computer system is an electronic device that is used for storing and processing data according to computer instructions. A task is a long list of instructions. The tasks can be very complicated and many tasks require input from the user. Users can be very unpredictable and the response time of a user can be very long. This is why we will only consider user independent tasks. Furthermore, repeatability is an important concept in scientific research, so we will only consider deterministic tasks. For those tasks, the most important components of a computer are the processor and the memory. Section 2.1 and Section 2.2 describe respectively the processor and memory. Section 2.3 specifies how the tasks are executed and which resources they require.

2.1. The processor

The processor is a hardware component of the computer system that is responsible for the execution of instructions. The processor comprises a set of processor cores, which can independently execute tasks. This is called task parallelism. Moreover, each core can execute multiple instructions at once. This is called instruction level parallelism.

2.2. The memory

There are many different types of memory and there exists an entire memory hierarchy.

2.3. Task execution

As said earlier, we will only consider deterministic tasks.

2.4. Task simulation software

3. Queueing theory and Markov chains

This chapter discusses relevant prior knowledge concerning Markov chains and queueing theory. Markov chains are simple mathematical models that are able to describe random phenomena evolving in time. In the field of Markov chains, the number of possible states of the system can be infinite, but we only care about finite state spaces because the queueing network used to model a processor is finite.

There are many different types of Markov chains, but we will only discuss continuous-time Markov chains, because those Markov chains are the most relevant when discussing queueing theory.

We will start by discussing some basic properties and definitions of continuous-time Markov chains, before we can apply them on the relevant networks within queueing theory: the closed queueing networks.

3.1. Continuous-time Markov chains

This section introduces continuous-time Markov chains and discusses some basis properties. We will also state some theorems, but before we do that, we must first give the definition of a state space and a Q-matrix.

Definition 3.1.1 (State space). *A state space I is a countable set. The elements of I are called states.*

Definition 3.1.2 (Q-matrix). *Let I be a state space. A Q-matrix on I is a matrix $Q = (q_{ij} : i, j \in I)$ for which the following conditions hold:*

- $q_{ij} \geq 0$ *for all* $i \neq j$*;*
- $0 \leq -q_{ii} < \infty$ *for all* $i \in I$*;*
- $\sum_{j \in I} q_{ij} = 0$ *for all* $i \in I$*.*

Then a Q-matrix, state space and initial distribution $\lambda = (\lambda_i : i \in I)$ induce a continuous-time Markov chain.

Definition 3.1.3. *(Continuous-time Markov chain) A process $(X_t)_{t \geq 0}$ is called a continuous-time Markov chain (CTMC) and denoted by $X_t \sim \text{Markov}(\lambda, Q)$ if for time t and state i it holds that*

$$\mathbb{P}\left[X_t = i\right] = \left(\lambda e^{tQ}\right)_i .$$

Defining a CTMC in such a way may seem arbitrary at first, but with this definition it can be shown that:

- the time it takes for the system to leave state i is $E(-q_{ii})$ (exponentially with parameter $-q_{ii}$) distributed;
- the probability of transitioning from state i to state $j \neq i$ is 0 if $q_{ii} = 0$ and $\frac{q_{ij}}{-q_{ii}}$ if $q_{ii} \neq 0$;
- the process is memoryless: the future probability distribution over the state space only depends on the current probability distribution over the state space. This is caused by the memoryless property of the exponential distribution.

Those properties give rise to the following mapping of a CTMC to a closed queueing network with one customer:

- a state maps to a service center;
- the system being in state i with probability p maps to a customer being in center i with probability p.

Due to the definition of a CTMC, we can now conclude the following properties for the queueing network:

- service center i has service rate $-q_{ii}$ (the service time is $E(-q_{ii})$ distributed). The service of the customer will never be completed if $q_{ii} = 0$.
- When the customer has been serviced by center i, it travels with probability $\frac{q_{ij}}{-q_{ii}}$ to center $j \neq i$.

Notice how these properties uniquely define a process and thus this can also be used as a definition of a CTMC. This equivalent definition of a CTMC gives reason to define a transition matrix $P = (p_{ij} : i, j \in I)$, where

$$p_{ij} = \begin{cases} \frac{q_{ij}}{-q_{ii}} & : j \neq i, q_{ii} \neq 0 \\ 0 & : j \neq i, q_{ii} = 0 \end{cases}, p_{ii} = \begin{cases} 0 & : q_{ii} \neq 0 \\ 1 & : q_{ii} = 0 \end{cases}.$$

This matrix states the transition probabilities between the different states in the state space I.

The equivalent definition of a CTMC is also illustrated in Figure 3.1. Observe that the service rates and transition probabilities induce the Q-matrix and transition matrix which are given by

$$Q = \begin{pmatrix} -4 & 0 & 4 \\ 3 & -3 & 0 \\ 2 & 6 & -8 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0.25 & 0.75 & 0 \end{pmatrix}.$$

The advantage of thinking about CTMC's as networks of service centers is that the system can be extended by adding more customers and interference between the customers. Formalising those extensions will lead to the field of queueing theory, which will be discussed in Section 3.2.

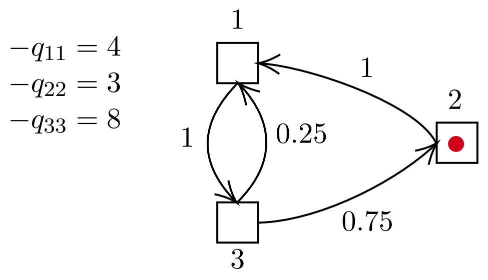


Figure 3.1.: An example of a CTMC illustrated as a network of service centers and a customer (red dot) travelling through the network. The service rates of the service centers are given by $-q_{ii}$ for service center i and the numbers on the arrows indicate transition probabilities for when the service has been completed. The current distribution of the system is given by $\lambda = (0, 1, 0)$.

3.1.1. The invariant distribution

For a process evolving in time, such as a CTMC, the equilibrium distribution is often a topic of interest. Definition 3.1.4 formalizes this concept.

Definition 3.1.4 (Invariant distribution)**.** *A distribution $\pi = (\pi_i : i \in I)$ is an invariant distribution over a Q-matrix Q if*

$$\pi Q = 0.$$

Defining an invariant distribution like this makes sense, because $\pi Q = 0$ implies that

$$\begin{aligned}
\mathbb{P}[X_t = i] &= \left(\pi e^{tQ}\right)_i \\
&= \left(\pi \left(Q^0 + tQ + \frac{t^2 Q^2}{2!} + \dots\right)\right)_i \\
&= \left(\pi + t\pi Q + \frac{t^2 \pi Q^2}{2!} + \dots\right)_i \\
&= \pi_i,
\end{aligned}$$

which means that the distribution over the state space stays the same with evolving time. Note that the condition $\pi Q = 0$ is equivalent to

$$\pi_i \sum_{j \in I \setminus \{i\}} q_{ij} = \sum_{j \in I \setminus \{i\}} q_{ji}, \quad i \in I, \tag{3.1}$$

due to the fact that $-q_{ii} = \sum_{j \in I \setminus \{i\}} q_{ij}$. Equation 3.1 will be referred to as the balance equation.

3.1.2. Convergence to equilibrium

A question one may ask is whether any initial distribution converges to the invariant distribution when time goes to infinity. There exists such a convergence theorem, but we need the definition of irreducibility in order to formulate the theorem.

Definition 3.1.5 (Irreducible)**.** *A CTMC is irreducible if for all states $i, j \in I$ there exists states $i_1, \dots, i_n$ such that $p_{i,i_1} p_{i_1,i_2} \cdots p_{i_n,j} > 0$.*

Irreducibility thus implies that you can travel from any state to any other state. Using this definition, we can formulate Theorem 3.1.6.

Theorem 3.1.6 (Convergence to equilibrium)**.** *Let $X_t \sim \text{Markov}(\lambda, Q)$ be irreducible, finite and having invariant distribution π. Then*

$$\mathbb{P}[X_t = i] \to \pi_i \quad \textit{as} \quad t \to \infty.$$

3.2. Closed Queueing networks

This section discusses closed queueing networks and their relations with CTMC's. Remember from Section 3.1 that a CTMC can be thought of as a network of service centers with one customers travelling through the network. If you let multiple customers of different classes travel through the network, you obtain a closed queueing network. To be more precise, a closed queueing network has a finite set of service centers, a finite set of customer classes and a finite number of customers of each class. We will first introduce some notation before we will formally define a closed queueing network.

Let I and R be respectively the number of service centers and customer classes. For convenience, we will index the service centers from 1 to M and we let $1 \leq i, j \leq M$ denote service centers while letting $1 \leq r, s \leq R$ denote customer classes. Let $\mathbf{N} = (N_1, \dots, N_R)$ denote the population vector where N_r corresponds to the total number of customers of class r.

3.2.1. The customers

The customers travel through the network according to transition probabilities. Those transition probabilities are the same for all customers within the same class. Let $p_{i,j}^{(r)}$ denote the transition probability of a customer of class r to travel from service center i to j. For simplicity, we assume $p_{i,i}^{(r)} = 0$.

3.2.2. The service centers

All service centers are exhibiting the First In First Out (FIFO) queueing discipline, with each center having multiple servers. This means that a customer arriving at a service station is send to the back of the queue and every customers moves up a spot when a customer leaves the service center. The queue contains both the non-served and served customers and the queue of service center i will be called queue i. The number of customers center i can serve simultaneously is denoted by C_i and will be called the capacity of center i. The service rate of a customer does not depend on its class. Here, a service rate of μ means that the service time is $E(\mu)$ distributed.

For n customers present at center i, let $\phi_i(n)$ and $\phi_{i,k}(n)$ be respectively the total service rate of center i and the service rate that the customer at position k in queue i

receives. The service rate will be equally divided between the first C_i customers in the queue, so we have

$$\phi_{i,k}(n) = \begin{cases} \frac{\phi_i(n)}{\min\{C_i,n\}} & : k \leq \min\{C_i, n\} \\ 0 & : \text{else} \end{cases} .$$

3.2.3. The state of the model

The state of the network is denoted by

$$x = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M),$$

where $\mathbf{x}_i$ denotes the state of service center i and

$$\mathbf{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,n_i(x)}\right).$$

Here, $n_i(x) := |\mathbf{x}_i|$ equals the number of customers in queue i and $x_{i,k}$ is the class of the customer at the k-th position in the queue. Formally, the state space is given by

$$\chi := \left\{ x : x = (\mathbf{x}_1, \ldots, \mathbf{x}_M), 1 \leq x_{i,k} \leq R, 1 \leq i \leq M, 1 \leq k \leq n_i(x), \sum_{i=1}^{M} \sum_{k=1}^{n_i(x)} \delta_{r,x_{i,k}} = N_r \right\},$$

where the last condition ensures that exactly N_r number of class r customers are in the system and

$$\delta_{i,j} = \begin{cases} 1 & : i = j \\ 0 & : i \neq j \end{cases},$$

denotes the Dirac delta function.

Notice how the transition probabilities $p_{i,j}^{(r)}$ and service rate functions $\phi_{i,k}(n)$ induce transition rates between the different states. This means that the closed queueing network is essentially a CTMC. The only non-zero transition rates are between the states where exactly one customer is being served by center i and travels to the back of queue $j \neq i$. This means that the Q-matrix is given by $Q = \left(q_{x,\underset{\sim}{y}} : x, \underset{\sim}{y} \in \chi\right)$, where

$$q_{x\underset{\sim}{y}} = p_{i,j}^{(x_{i,k})} \phi_{i,k}(n_i(x)), \quad j \neq i,$$

and $\underset{\sim}{y}$ is the state that is created from x when the customer at position k in queue i is served and moves to the back of queue $j \neq i$.

3.2.4. Aggregate states

The exact order of the customers in the queue is often not important to know, but only how many customers of each class are in each queue. Furthermore, the state space χ contains many states, so it makes sense to remove unnecessary information from the states in order to decrease the size of the state space.

An aggregate state is denoted by $\boldsymbol{n} = (\mathbf{n}_1, \ldots, \mathbf{n}_M)$ with $\mathbf{n}_i = (n_{i,1}, \ldots, n_{i,R})$ where $n_{i,r}$ denotes the number of class r customers at service center i. The aggregate state space for population vector $\mathbf{N}$ is defined by

$$\mathcal{N}(\mathbf{N}) \coloneqq \left\{ \boldsymbol{n} : \boldsymbol{n} = (\mathbf{n}_1, \ldots, \mathbf{n}_M), n_{i,r} \geq 0, 1 \leq i \leq M, 1 \leq r \leq R, \sum_{j=1}^{M} n_{j,r} = N_r \right\}.$$

Just like for states $\boldsymbol{x} \in \chi$, we will use $n_i(\boldsymbol{n})$ to denote the total number of customers at service center i. Thus we have

$$n_i(\boldsymbol{n}) = \sum_{r=1}^{M} n_{i,r}.$$

3.2.5. The invariant distribution

Remember that the invariant distribution $\pi = (\pi_{\boldsymbol{x}} : \boldsymbol{x} \in \chi)$ for a CTMC can be found by solving the balance equations. Before presenting the solution to the balance equations, we must first introduce the **one customer invariant measure terms**.

Definition 3.2.1. *The **one customer invariant measure terms** are given by*

$$e = (e_{i,r} : 1 \leq i \leq M, 1 \leq r \leq R),$$

which satisfy the equations

$$\sum_{j=1}^{M} e_{j,r} p_{j,i}^{(r)} = e_{i,r}. \tag{3.2}$$

In order to conclude using Theorem 3.1.6 that there exists at most one invariant distribution, we make the following assumptions:

- When in equilibrium, all class r customers visit the same set of centers and we will denote this set by $M(r)$, which is formally defined as

  $$M(r) \coloneqq \{i : e_{i,r} \neq 0\}.$$

  Similarly we can define $R(i)$ to be set of customer classes that visit center i, or formally

  $$R(i) \coloneqq \{r : e_{i,r} \neq 0\}.$$

- The transition matrix $P^{(r)} = \left(p_{i,j}^{(r)} : 1 \leq i, j \leq M\right)$ is irreducible over $M(r)$.

Theorem 3.2.2 gives the general solution to the balance equations. The theorem and its proof originates from [Baskett et al., 1975].

Theorem 3.2.2. *For a closed queueing network with service centers exhibiting the FIFO server discipline, it holds that the unique invariant distribution $\pi = (\pi_x : x \in \chi)$ is given by*

$$\pi_x = \frac{1}{G(\mathbf{N})} \prod_{i=1}^{M} \prod_{k=1}^{n_i(x)} \frac{e_{i,x_{i,k}}}{\phi_i(k)}, \tag{3.3}$$

where $G(\mathbf{N})$ is a normalising constant such that $\sum_{x\in\chi} \pi_x = 1$, i.e.

$$G(\mathbf{N}) = \sum_{x\in\chi} \prod_{i=1}^{M} \prod_{k=1}^{n_i(x)} \frac{e_{i,x_{i,k}}}{\phi_i(k)}.$$

Proof. Uniqueness follows from the fact that the transition matrix $P^{(r)}$ is irreducible over $M(r)$. The theorem can now be proven by showing that π satisfies the balance equations

$$\pi_x \sum_{y\in\chi\setminus\{x\}} q_{xy} = \sum_{y\in\chi\setminus\{x\}} \pi_y q_{yx}, \quad x \in \chi.$$

It holds for a state $x \in \chi$ that the only non-zero transition rates q_{xy} correspond to the transition of exactly one customer moving to another service center. This means that the left-hand side of the balance equations can be calculated by summing over all tuples (i, j, k) corresponding to the transition of the customer at position k of queue i to queue j and thus we have

$$\begin{aligned}
\pi_x \sum_{y\in\chi} q_{xy} &= \pi_x \sum_{i=1}^{M} \sum_{k=1}^{n_i(x)} \sum_{j=1}^{M} \phi_{i,k}(n_i(x)) p_{i,j}^{(x_{i,k})} \\
&= \pi_x \sum_{i=1}^{M} \sum_{k=1}^{n_i(x)} \phi_{i,k}(n_i(x)) \sum_{j=1}^{M} p_{i,j}^{(x_{i,k})} \\
&= \pi_x \sum_{i=1}^{M} \sum_{k=1}^{n_i(x)} \phi_{i,k}(n_i(x)) \\
&= \pi_x \sum_{i=1}^{M} \phi_i(n_i(x)).
\end{aligned}$$

In order to calculate the right-hand side of the balance equations, we define

$$T_x := \left\{(k, j, i) \in \mathbb{N}^3 : 1 \le i, j \le M, 1 \le k \le n_j(x) + 1, n_i(x) \ge 1\right\},$$

as the set of all possible transitions that result in state x. Here, (k, j, i) corresponds to the k-th customer of service center j moving to service center i. Notice how $n_i(x) \ge 1$ is a necessary condition, because no transition to center i can occur if there is no customer at center i in state x.

Now for $x \in \chi$ and $(k, j, i) \in T_x$ let $y \in \chi$ correspond to the state such that the k-th customer moving center j to center i results in state x. Then in state y there is one more customer in center j and one less in center i, thus

$$\pi_y = \pi_x \frac{e_{j,x_{i,n_i(x)}}}{\phi_j(n_j(x)+1)} \frac{1}{\frac{e_{i,x_{i,n_i(x)}}}{\phi_i(n_i(x))}} = \pi_x \frac{e_{j,x_{i,n_i(x)}}}{\phi_j(n_j(x)+1)} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}}.$$

The right-hand side of the balance equations can now be calculated by summing over all elements of T_x:

$$\begin{aligned}
\sum_{y \in \chi} \pi_y q_{yx} &= \pi_x \sum_{i=1, n_i(x) \geq 1}^{M} \sum_{j=1}^{M} \sum_{k=1}^{n_j(x)+1} \frac{e_{j,x_{i,n_i(x)}}}{\phi_j(n_j(x)+1)} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}} p_{j,i}^{(x_{i,n_i(x)})} \phi_{j,k}(n_j(x)+1) \\
&= \pi_x \sum_{i=1, n_i(x) \geq 1}^{M} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}} \sum_{j=1}^{M} \frac{e_{j,x_{i,n_i(x)}}}{\phi_j(n_j(x)+1)} p_{j,i}^{(x_{i,n_i(x)})} \sum_{k=1}^{n_j(x)+1} \phi_{j,k}(n_j(x)+1) \\
&= \pi_x \sum_{i=1, n_i(x) \geq 1}^{M} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}} \sum_{j=1}^{M} \frac{e_{j,x_{i,n_i(x)}}}{\phi_j(n_j(x)+1)} p_{j,i}^{(x_{i,n_i(x)})} \phi_j(n_j(x)+1) \\
&= \pi_x \sum_{i=1, n_i(x) \geq 1}^{M} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}} \sum_{j=1}^{M} e_{j,x_{i,n_i(x)}} p_{j,i}^{(x_{i,n_i(x)})} \\
&= \pi_x \sum_{i=1}^{M} \frac{\phi_i(n_i(x))}{e_i, x_{i,n_i(x)}} e_{i,x_{i,n_i(x)}} \qquad \text{(by 3.2)} \\
&= \pi_x \sum_{i=1}^{M} \phi_i(n_i(x)),
\end{aligned}$$

which is equal to the left hand side of the balance equations. This implies that $\pi = (\pi_x : x \in \chi)$ is the unique invariant distribution. $\square$

The invariant distribution of Equation 3.3 enables us to calculate the equilibrium state probabilities of the aggregate states $\mathcal{N}(\mathbf{N})$. Theorem 3.2.3 gives these probabilities and the theorem and proof originates from [Baskett et al., 1975].

Theorem 3.2.3. *The equilibrium state probabilities for $n \in \mathcal{N}(\mathbf{N})$ are given by*

$$\pi_n = \left[\prod_{i=1}^{N} \binom{n_i}{n_{i,1}, \ldots, n_{i,R}} \right] \pi_x,$$

for a $x \in \chi$ that induces the aggregate state n.

Proof. For $n \in \mathcal{N}(\mathbf{N})$ define χ_n as the set of states that induce the aggregate state n. Formally, this is defined as

$$\chi_n := \left\{ x \in \chi \; : \; \sum_{k=1}^{n_i(n)} \delta_{r,x_{i,k}} = n_{i,r}, 1 \leq i \leq M, 1 \leq r \leq R \right\}.$$

Notice how

$$|\chi_{\boldsymbol{n}}| = \prod_{i=1}^{M} \binom{n_i(\boldsymbol{n})}{n_{i,1}, \ldots, n_{i,R}},$$

because there are

$$\binom{n_i(\boldsymbol{n})}{n_{i,1}, \ldots, n_{i,R}}$$

different queues possible at center i if the number of customers of each class is known. For $x \in \chi_{\boldsymbol{n}}$ and service center i we have that the expression

$$\prod_{k=1}^{n_i(\boldsymbol{n})} e_{i,x_{i,k}} = \prod_{r=1}^{R} e_{i,n_{i,r}},$$

only depends on $\boldsymbol{n}$ and not on the exact element within $\chi_{\boldsymbol{n}}$. This implies that π_x is the same for all $x \in \chi_{\boldsymbol{n}}$. We can use this to obtain an equilibrium state probability of

$$\pi_{\boldsymbol{n}} = \sum_{x \in \chi_{\boldsymbol{n}}} \pi_x = |\chi_{\boldsymbol{n}}| \, \pi_x = \left[\prod_{i=1}^{M} \binom{n_i(\boldsymbol{n})}{n_{i,1}, \ldots, n_{i,R}} \right] \pi_x.$$

□

3.3. Mean value analysis

The calculation of the normalisation constant $G(\mathbf{N})$ in Equation 3.3 requires a summation over all possible states. Due to the extremely large number of possible states, this is computationally infeasible. However, most of the time the exact invariant distribution is not of interest and only the performance measures (e.g. utilization, throughput) of a queueing network are of interest. As it turns out, the calculation of those performance measures does not require the computation of the normalisation constant. An efficient algorithm to calculate the performance measures was found in 1980 by [Reiser and Lavenberg, 1980]. The algorithm is called Mean Value Analysis and this section is devoted to explaining and proving the correctness of the algorithm. The algorithm is only applicable to closed multiclass queueing networks, but it was later extended by [Bruell et al., 1984] to also allow open customer classes. We will only consider closed queueing networks in this thesis, so the theorems and results in this section originate from [Reiser and Lavenberg, 1980].

3.3.1. The normalising constant

The MVA algorithm starts by calculating the performance measures for the population vector $\mathbf{0}$ and we will inductively add customers until the desired population vector of $\mathbf{N}$ is reached. We will use $\mathbf{N}'$ and $\mathbf{N}''$ to denote arbitrary population vectors and we use $\mathbf{N}'' \leq \mathbf{N}'$ to denote that $N''_r \leq N'_r$ for all $1 \leq r \leq R$.

Remember that the normalisation constant for a population vector $\mathbf{N}'$ is given by

$$G(\mathbf{N}') = \sum_{n \in \mathcal{N}(\mathbf{N}')} \prod_{i=1}^{M} f_i(\mathbf{n}_i),$$

where

$$f_i(\mathbf{n}) := C(\mathbf{n}) E_i(\mathbf{n}) \Phi_i(|\mathbf{n}|),$$

with

$$\begin{aligned} |\mathbf{n}_i| &:= n_{i,1} + \ldots + n_{i,R}, \\ C(\mathbf{n}_i) &:= \binom{|\mathbf{n}_i|}{n_{i,1}, \ldots, n_{i,R}}, \\ E_i(\mathbf{n}_i) &:= \prod_{r=1}^{R} (e_{i,r})^{n_{i,r}}, \\ \Phi_i(n) &:= \prod_{k=1}^{n} \frac{1}{\phi_i(k)}, \quad n \in \mathbb{N}. \end{aligned}$$

For the sake of completeness, we define the normalisation constant $G_i(\mathbf{N}'', \mathbf{N}')$ by

$$G_i(\mathbf{N}'', \mathbf{N}') := \sum_{n \in \mathcal{N}_i(\mathbf{N}'', \mathbf{N}')} \prod_{j=1, j \neq i}^{M} f_j(\mathbf{n}_j),$$

where

$$\mathcal{N}_i(\mathbf{N}'', \mathbf{N}') := \left\{ n \in \mathcal{N}(\mathbf{N}') : \mathbf{n}_i = \mathbf{N}'' \right\}.$$

The normalisation constant $G_i(\mathbf{N}'', \mathbf{N}')$ can be interpreted as fixing $\mathbf{N}''$ out of $\mathbf{N}'$ number of customers at service station i.

3.3.2. The performance measures

For all $\mathbf{N}' \leq \mathbf{N}$ the MVA algorithm will calculate the following performance measures:

- The mean throughput $\overline{x}_{i,r}(\mathbf{N}')$ is the mean number of class r customers that leave center i in 1 unit of time.
- The utilization $u_i(\mathbf{N}')$ is the mean number of busy servers at service center i.
- The mean queue length $\overline{n}_i(\mathbf{N}')$ of center i.
- The mean waiting time $\overline{w}_{i,r}(\mathbf{N}')$ of class r customers at service center i.
- For $n \in \mathbb{N}$, the marginal queue length $p_i(n, \mathbf{N}')$ is the probability that there are exactly n customers at center i, while $p_i(\mathbf{N}'', \mathbf{N}')$ is the probability that the population vector at center i is given by $\mathbf{N}''$.

Little's law is an important law that we will use in a number of proofs. For a queueing network in equilibrium distribution, the law states that the mean number of customers in a component of the queueing network equals the product of the waiting time and throughput in that component of the network. We can either view the service center as a component, or only the C_i servers in the center. For the first case, we have that

$$\overline{n}_{i,r}(\mathbf{N}') = \overline{w}_{i,r}(\mathbf{N}')\overline{x}_{i,r}(\mathbf{N}'),$$

while for the second case, we obtain

$$u_i(\mathbf{N}') = \frac{1}{\mu_i}\sum_{r=1}^{R}\overline{x}_{i,r}(\mathbf{N}').$$

We will now continue by stating the relations between the different performance measures. Lemma 3.3.1 and Lemma 3.3.2 state respectively an equation for the throughput and the marginal probability. The notation $\mathbf{1}_r$ will be used to denote a R component vector whose components are zero except for the r-th component, which is 1.

Lemma 3.3.1. *For the throughput it holds that*

$$\overline{x}_{i,r}(\mathbf{N}') = e_{i,r}\frac{G(\mathbf{N}' - \mathbf{1}_r)}{G(\mathbf{N}')}. \tag{3.4}$$

Proof. (proof of the theorem) □

Lemma 3.3.2. *For the marginal queue length distribution it holds that*

$$p_i(n, \mathbf{N}') = \frac{1}{\phi_i(n)}\sum_{r=1}^{R}\overline{x}_{i,r}(\mathbf{N}')p_i(n-1, \mathbf{N}' - \mathbf{1}_r), \quad (n \geq 0). \tag{3.5}$$

Proof. We have that

$$\begin{aligned}
p_i(\mathbf{N}'', \mathbf{N}') &= E_i(\mathbf{N}'')\Phi_i(|\mathbf{N}''|)C(\mathbf{N}'')\frac{G_i(\mathbf{N}'', \mathbf{N}')}{G(\mathbf{N}')} \\
&= \sum_{r=1}^{R} e_{i,r}E_i(\mathbf{N}'' - \mathbf{1}_r)\frac{\Phi_i(|\mathbf{N}''| - 1)}{\phi_i(|\mathbf{N}''|)}C(\mathbf{N}'' - \mathbf{1}_r)\frac{G_i(\mathbf{N}'' - \mathbf{1}_r, \mathbf{N}' - \mathbf{1}_r)}{G(\mathbf{N}' - \mathbf{1}_r)}\frac{G(\mathbf{N}' - \mathbf{1}_r)}{G(\mathbf{N}')} \\
&= \frac{1}{\phi_i(|\mathbf{N}''|)}\sum_{r=1}^{R} e_{i,r}p_i(\mathbf{N}'' - \mathbf{1}_r, \mathbf{N}' - \mathbf{1}_r)\frac{G(\mathbf{N}' - \mathbf{1}_r)}{G(\mathbf{N}')} \\
&= \frac{1}{\phi_i(|\mathbf{N}''|)}\sum_{r=1}^{R}\overline{x}_{i,r}(\mathbf{N}')p_i(\mathbf{N}'' - \mathbf{1}_r, \mathbf{N}' - \mathbf{1}_r).
\end{aligned}$$

Equation 3.5 is now obtained by taking a summation over all $\mathbf{N}''$ with $|\mathbf{N}''| = n$. □

The main theorem of this section is considered to be Theorem 3.3.3, which states a recurrence for the mean waiting time.

Theorem 3.3.3. *A recurrence for the mean waiting time is given by*

$$\overline{w}_{i,r}(\mathbf{N}') = \sum_{n=1}^{|\mathbf{N}'|} \frac{n}{\phi_i(n)} p_i(n-1, \mathbf{N}' - \mathbf{1}_r)$$

Proof. The mean number of class r customers at center i is given by

$$\begin{aligned}
\overline{n}_{i,r}(\mathbf{N}') &= \sum_{\mathbf{N}''=\mathbf{0}}^{\mathbf{N}'} N''_r p_i(\mathbf{N}'', \mathbf{N}') \\
&= \sum_{\mathbf{N}''=\mathbf{1}_r}^{\mathbf{N}'} N''_r C(\mathbf{N}'')\Phi_i(|\mathbf{N}''|)E_i(\mathbf{N}'')\frac{G_i(\mathbf{N}'', \mathbf{N}')}{G(\mathbf{N}')} \\
&= \sum_{\mathbf{N}''=\mathbf{1}_r}^{\mathbf{N}'} |N''|C(\mathbf{N}'' - \mathbf{1}_r)\frac{\Phi_i(|\mathbf{N}''|-1)}{\phi_i(|\mathbf{N}''|)} e_{i,r} E_i(\mathbf{N}'' - \mathbf{1}_r)\frac{G_i(\mathbf{N}' - \mathbf{1}_r, \mathbf{N}'' - \mathbf{1}_r)}{G(\mathbf{N}' - \mathbf{1}_r)}\frac{G(\mathbf{N}' - \mathbf{1}_r)}{G(\mathbf{N}')} \\
&= \sum_{\mathbf{N}''=\mathbf{1}_r}^{\mathbf{N}'} \frac{|N''|}{\phi_i(|\mathbf{N}''|)} \overline{x}_{i,r}(\mathbf{N}') p_i(\mathbf{N}'' - \mathbf{1}_r, \mathbf{N}' - \mathbf{1}_r) \\
&= \overline{x}_{i,r}(\mathbf{N}') \sum_{n=1}^{|\mathbf{N}'|} \frac{n}{\phi_i(n)} \sum_{|\mathbf{N}''|=n} p_i(\mathbf{N}'' - \mathbf{1}_r, \mathbf{N}' - \mathbf{1}_r) \\
&= \overline{x}_{i,r}(\mathbf{N}') \sum_{n=1}^{|\mathbf{N}'|} \frac{n}{\phi_i(n)} p_i(n-1, \mathbf{N}' - \mathbf{1}_r).
\end{aligned}$$

By using Little's law we obtain

$$\overline{w}_{i,r}(\mathbf{N}') = \sum_{n=1}^{|\mathbf{N}'|} \frac{n}{\phi_i(n)} p_i(n-1, \mathbf{N}' - \mathbf{1}_r).$$

□

The waiting time equation in Theorem 3.3.3 can be simplified for a Constant Rate (CR) queueing network. In a CR network service center i has C_i number servers, each with a constant service rate of μ_i. This is formally defined in Definition 3.3.4.

Definition 3.3.4 (CR queueing network)**.** *A Constant Rate (CR) queueing network is a network for which holds that*

$$\phi_i(n) = \min\{n, C_i\}\mu_i, \quad i = 1, \ldots, M.$$

Corollary 3.3.5. *For a CR queueing network, we have*

$$\overline{w}_{i,r}(\mathbf{N}) = \frac{1}{C_i\mu_i}\left[1 + \overline{n}_i(\mathbf{N} - \mathbf{1}_r) + \sum_{n=0}^{C_i-2}(C_i - n - 1)p_i(n, \mathbf{N} - \mathbf{1}_r)\right].$$

Proof. For convenience we will use p_i to denote $p_i(n, \mathbf{N}' - \mathbf{1}_r)$. We then have that

$$
\begin{aligned}
\overline{w}_{i,r}(\mathbf{N}') &= \sum_{n=1}^{\mathbf{N}'} \frac{n}{\phi_i(n)} p_i(n-1, \mathbf{N}' - \mathbf{1}_r) \\
&= \frac{1}{\mu_i}\left[\sum_{n=1}^{C_i} p_i(n-1, \mathbf{N}' - \mathbf{1}_r) + \frac{1}{C_i}\sum_{n=C_i+1}^{|\mathbf{N}'|}(n-1+1)p_i(n-1, \mathbf{N}' - \mathbf{1}_r)\right] \\
&= \frac{1}{\mu_i}\left[\sum_{n=0}^{C_i-1} p_i + \frac{1}{C_i}\sum_{n=C_i}^{|\mathbf{N}'-\mathbf{1}_r|} np_i + \frac{1}{C_i}\sum_{n=C_i}^{|\mathbf{N}'-\mathbf{1}_r|} p_i - \frac{1}{C_i}\sum_{n=0}^{C_i-1} np_i + \frac{1}{C_i}\sum_{n=0}^{C_i-1} np_i\right] \\
&= \frac{1}{\mu_i}\left[\frac{1}{C_i}\sum_{n=0}^{C_i-1} C_i p_i + \frac{1}{C_i}\sum_{n=0}^{|\mathbf{N}'-\mathbf{1}_r|} np_i + \frac{1}{C_i}\left(1 - \sum_{n=0}^{C_i-1} p_i\right) - \frac{1}{C_i}\sum_{n=0}^{C_i-1} np_i\right] \\
&= \frac{1}{C_i\mu_i}\left[1 + \overline{n}_i(\mathbf{N}' - \mathbf{1}_r) + \sum_{n=0}^{C_i-1}(C_i - n - 1)p_i\right].
\end{aligned}
$$

□

3.3.3. The algorithm

We have already found recurrences for the mean waiting time $\overline{w}_{i,r}$ and the marginal queue length distribution p_i. This means it remains to find equations for the throughput $\overline{x}_{i,r}$ and mean queue length $\overline{n}_i$. Moreover, we also need the base case of the recurrence, that is, we need to find the values of $\overline{x}_{i,r}(\mathbf{0})$, $\overline{n}_i(\mathbf{0})$, $p_i(0, \mathbf{0})$ and $p_i(0, \mathbf{N}')$.

To decrease the computational complexity of the algorithm, we will choose a service center $l^*(r) \in M(r)$ such that and normalise $e_r = (e_{i,r} : i = 1, \ldots, M)$ such that $e_{l^*(r),r} = 1$.

The base case

We have that

$$
\begin{aligned}
p_i(0, \mathbf{0}) &= 1 \\
\overline{n}_i(\mathbf{0}) &= 0.
\end{aligned}
$$

The value of $p_i(0, \mathbf{N}')$ is a bit more difficult to obtain, but it can be done by using the fact that the mean number of idle servers is given by

$$
\sum_{n=0}^{C_i-1}(C_i - n)p_i(n, \mathbf{N}') = C_i - u_i(\mathbf{N}'),
$$

and the utilization can be expressed in terms of the throughput:

$$
u_i(\mathbf{N}') = \frac{1}{\mu_i}\sum_{r=1}^{R} \overline{x}_{i,r}(\mathbf{N}'). \tag{3.6}
$$

Combining these equations gives

$$p_i(0, \mathbf{N}') = 1 - \frac{1}{C_i}\left[u_i(\mathbf{N}') + \sum_{n=1}^{C_i-1}(C_i - n)p_i(n, \mathbf{N}').\right] \tag{3.7}$$

The mean queueing length and throughput

Notice that from the throughput equation 3.9 and the fact that $e_{l^*(r),r} = 1$ we can deduce that

$$\overline{x}_{i,r}(\mathbf{N}') = e_{i,r}\overline{x}_{l^*(r),r}(\mathbf{N}').$$

We can apply this equation with Little's law to obtain

$$\overline{n}_{i,r}(\mathbf{N}') = \overline{x}_{i,r}(\mathbf{N}')\overline{w}_{i,r}(\mathbf{N}') = e_{i,r}\overline{x}_{l^*(r),r}(\mathbf{N}')\overline{w}_{i,r}(\mathbf{N}'), \tag{3.8}$$

Summing Little's law over all service centers gives us

$$N_r = \sum_{i=1}^{M}\overline{n}_{i,r}(\mathbf{N}') = \sum_{i=1}^{M}\overline{x}_{i,r}(\mathbf{N}')\overline{w}_{i,r}(\mathbf{N}'),$$

which implies that

$$\overline{x}_{l^*(r),r}(\mathbf{N}') = \frac{N_r}{\sum_{i=1}^{M} e_{i,r}\overline{w}_{i,r}(\mathbf{N}')}. \tag{3.9}$$

The pseudo code

The obtained results can then be combined to obtain algorithm 1.

The alghorithmic complexity

In order to give a concise formula for the total time complexity of the algorithm we will introduce the constants

$$\begin{aligned} M' &:= \max\{|M(r)| : r = 1, 2, \ldots, R\} \\ C' &:= \max\{C_i : i = 1, 2, \ldots, M\} \\ R' &:= \max\{|R(i)| : i = 1, 2, \ldots, M\}. \end{aligned}$$

We have that the main for-loop of the algorithm iterates over $\prod_{r=1}^{R} N_r$ possible population vectors $\mathbf{N}'$. Then for each population vector the time complexities of calculating the performance measures are visible in Table 3.1.

This means that the total time complexity of the algorithm can be bounded above by

$$\mathcal{O}\left(N_1 N_2 \cdots N_R \max\left\{RM'C', R'MC'\right\}\right) \leq \mathcal{O}(N_1 N_2 \cdots N_R RMC').$$

This formula suggests that the execution time of the algorithm is largely determined by the number of customers of each class and not so much by the number of service centers

Algorithm 1 Calculate the performance measures of a CR queueing network

```
Require: $e_{l^*(r),r} = 1, r = 1, 2, \ldots, R$

  {initialization}
  for $i = 1, 2, \ldots, M$ do
    $\overline{n}_i(\mathbf{0}) \leftarrow 0$
    $p_i(0, \mathbf{0}) \leftarrow 1$
  end for

  {iteration over all possible population vectors (the main loop)}
  for $\mathbf{N}' \leq \mathbf{N}$ do

    {mean waiting time and throughput}
    for $r = 1, 2 \ldots, R$ do

      {mean waiting time calculation using Corollary 3.3.5}
      for $i \in M(r)$ do
        $\overline{w}_{i,r}(\mathbf{N}') \leftarrow \frac{1}{C_i \mu_i} \left[1 + \overline{n}_i(\mathbf{N}' - \mathbf{1}_r) + \sum_{n=0}^{\min\{C_i - 2, |\mathbf{N}'| - 1\}} (C_i - n - 1) p_i(n, \mathbf{N}' - \mathbf{1}_r)\right]$
      end for

      {throughput calculation using Equation 3.9}
      $\overline{x}_{l^*(r),r}(\mathbf{N}') \leftarrow \frac{N_r}{\sum_{i \in M(r)} e_{i,r} \overline{w}_{i,r}(\mathbf{N}')}$
    end for

    {mean queue length, utilization and queue length distribution}
    for $i = 1, 2, \ldots, M$ do

      {mean queue length calculation using Equation 3.8}
      $\overline{n}_i(\mathbf{N}') \leftarrow \sum_{r \in R(i)} e_{i,r} \overline{x}_{l^*(r),r}(\mathbf{N}') \overline{w}_{i,r}(\mathbf{N}')$

      {utilization calculation using Equation 3.6}
      $u_i(\mathbf{N}') \leftarrow \frac{1}{\mu_i} \sum_{r \in R(i)} \overline{x}_{l^*(r),r}(\mathbf{N}') e_{i,r}$

      {queue length distribution calculation using Lemma 3.3.2 and Equation 3.7}
      for $n = 1, 2, \ldots, \min\{C_i - 1, |\mathbf{N}'|\}$ do
        $p_i(n, \mathbf{N}') \leftarrow \frac{1}{\phi_i(n)} \sum_{r \in R(i)} \overline{x}_{i,r}(\mathbf{N}') p_i(n - 1, \mathbf{N}' - \mathbf{1}_r)$
      end for
      $p_i(0, \mathbf{N}') \leftarrow 1 - \frac{1}{C_i} \left[u_i(\mathbf{N}') + \sum_{n=1}^{\min\{C_i - 1, |\mathbf{N}|\}} (C_i - n) p_i(n, \mathbf{N}')\right]$
    end for
  end for
```

performance measure	time complexity for a $\mathbf{N}' \leq \mathbf{N}$	upperbound
mean waiting time	$\mathcal{O}\left(\sum_{r=1}^{R}\sum_{i\in M(r)} C_i\right)$	$\mathcal{O}\left(RM'C'\right)$
throughput	$\mathcal{O}\left(\sum_{r=1}^{R} \lvert M(r)\rvert\right)$	$\mathcal{O}\left(RM'\right)$
mean queue length	$\mathcal{O}\left(\sum_{i=1}^{M} \lvert R(i)\rvert\right)$	$\mathcal{O}\left(R'M\right)$
utilization	$\mathcal{O}\left(\sum_{i=1}^{M} \lvert R(i)\rvert\right)$	$\mathcal{O}\left(R'M\right)$
queue length distribution	$\mathcal{O}\left(\sum_{i=1}^{M} \lvert C_i\rvert \lvert R(i)\rvert\right)$	$\mathcal{O}\left(R'MC'\right)$

Table 3.1.: The time complexities of calculating the performance measures for a specific population vector $\mathbf{N}' \leq \mathbf{N}$. Concise upperbounds for the time complexities are also given.

or other parameters of the system. It can also be observed that even for networks with relatively small numbers of customer classes it can be computationally infeasible to calculate the performance measures. However, there exist approximations of the performance measures with time complexity

$$\mathcal{O}\left((N_1 + N_2 + \ldots + N_R)MC'\right),$$

which is a drastic improvement from the time complexity of MVA. An approximation for CR queueing networks is given by [Akyildiz and Bolch, 1988]. The paper and its algorithm are discussed in next section.

3.4. Performance measures approximation

An algorithm to estimate the performance measures from Section 3.3.2 is discussed in this section. The algorithm estimates the marginal probabilities $p_i(n, \mathbf{N}')$ from the average number of customers $n_i(\mathbf{N}')$ and has computational complexity

$$\mathcal{O}\left((N_1 + N_2 + \ldots + N_R)MC'\right).$$

4. Model description and analysis

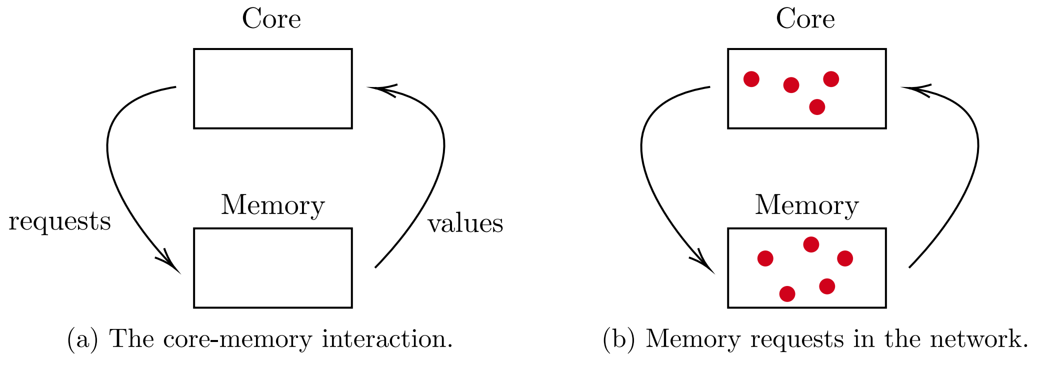
The queueing theory from Chapter 3 can now be used to create a model which imitates the dynamics of the processor as described in Chapter 2. Section 4.1.1 describes the model and the approach taken to arrive at the model, while Section 4.2 analyses the model and Section 4.3 presents derivations for the model parameters.

Note that the model is highly abstract and only aims at recreating the processor dynamics. This can give a non-trivial relation between the model parameters and the components and processes in the processor.

4.1. Model description and explanation

4.1.1. The model dynamics overview

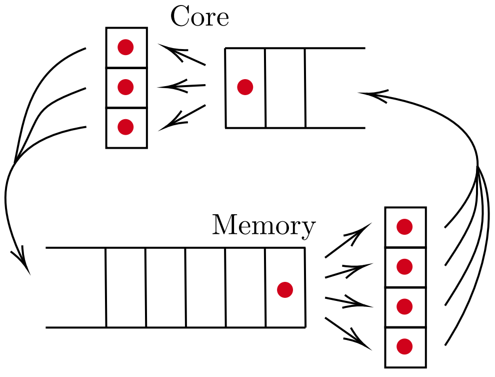
The most relevant interaction in the processor is the sending of memory requests by the core to the memory. There are two types of memory requests: local and DRAM requests. The local requests require the memory inside the core, and thus won't get effected by an increase in the number of tasks run in parallel. Therefore, those requests won't be considered in the model. The DRAM requests require the DRAM, which is the shared memory between the cores. The service rate of these requests is effected by the other tasks due to the finite memory bandwidth. This gives reason to consider those requests in the model. The dynamic induced by those requests is visualised in Figure 4.1a.



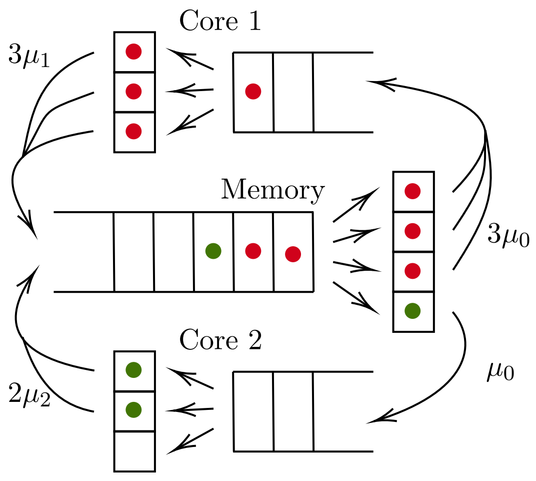
(a) The core-memory interaction. (b) Memory requests in the network.

Figure 4.1.: The system dynamics modelled with requests travelling through the system.

The model can then be made more explicit by letting the requests travel through the system as customers travelling through a closed queueing network (Figure 4.1b). The requests in the core then correspond to requests that are being created.



(a) Single core.



(b) Two cores.

Figure 4.2.: The core and memory depicted as queues serving requests in a system with one (a) and two cores (b).

The model can be further concretized by realising that the memory can only serve a finite number of requests. Similarly, the core can only handle a finite number of instructions at once, so the rate at which requests are send is bounded. This can be incorporated into the model by replacing the core and memory rectangles in Figure 4.1b with a queue to obtain Figure 4.2a. The resulting network is a closed queueing network and multiple cores can easily added to the network. The case of two cores is given in Figure 4.2b. The formal definition of the model is given in Definition 4.1.1, where service center 0 corresponds to the memory and centers $1 \leq i \leq M$ correspond to the cores.

Definition 4.1.1. *The M-core single memory model is defined as the triple $(\mathbf{N}, \mu, C)$, where*

$$\mathbf{N} = (N_i \in \mathbb{N} : 1 \leq i \leq M)$$

is the population vector,

$$\mu = (\mu_i : 0 \leq i \leq M),$$

is the service rate vector and

$$C = (C_i : 0 \leq i \leq M)$$

is the capacity vector. There are $R = M$ classes of customers and

$$p_{i,0}^{(i)} = p_{0,i}^{(i)} = 1, \quad 1 \leq i \leq M.$$

4.2. Model analysis

The theorems from Chapter 3 can now be used to get an expression for the terms $e_{i,r}$ and the invariant distribution. For the terms $e_{i,r}$ it must hold that

$$e_{i,r} = \sum_{j=0}^{M} e_{j,r} p_{j,i}^{(r)} = \begin{cases} \sum_{j=1}^{M} e_{j,r} & : i = 0 \\ e_{0,r} & : i \neq 0 \end{cases},$$

which means that we can simply let $e_{0,i} = 1$ and $e_{i,i} = 1$ for all $1 \leq i \leq M$, and all other terms are equal to 0. This means that

$$S(r) = \{0, r\}, \quad 1 \leq r \leq M,$$

and notice that the transition matrix $P^{(r)}$ is irreducible over this set. This directly implies that any $\boldsymbol{x} \in \chi$ with $x_{i,k} \neq i$ for $1 \leq i \leq M$ has 0 probability of occurring in equilibrium, because then $e_{i,x_{i,k}} = 0$. In other words, in equilibrium there are only class i customers in core i, which should be the case. We can use this to obtain for $\boldsymbol{x} \in \chi$ with $x_{i,k} = i$ for all $1 \leq i \leq M$ that

$$\begin{aligned} \pi_{\boldsymbol{x}} &= C \prod_{i=0}^{M} \prod_{k=1}^{n_i} \frac{e_{i,x_{i,k}}}{\min\{C_i, k\} \mu_i} \\ &= C \prod_{i=0}^{M} \prod_{k=1}^{n_i} \frac{1}{\min\{C_i, k\} \mu_i} \\ &= C \prod_{i=0}^{M} \frac{1}{\min\{C_i, n_i\}! C_i^{\min\{C_i - n_i, 0\}} \mu_i^{n_i}}. \end{aligned}$$

Furthermore, for the equilibrium state probabilities of the aggregate state $\boldsymbol{n} \in \mathcal{N}(\mathbf{N})$ we obtain

$$\begin{aligned} \pi_{\boldsymbol{n}} &= C \left[\prod_{i=1}^{M} \frac{n_i!}{\prod_{r=1}^{M} (n_{i,r}!)} \right] \prod_{i=0}^{M} \frac{1}{\min\{C_i, n_i\}! C_i^{\min\{C_i - n_i, 0\}} \mu_i^{n_i}} \\ &= C \frac{n_0!}{\prod_{r=1}^{M} (n_{0,r}!)} \prod_{i=0}^{M} \frac{1}{\min\{C_i, n_i\}! C_i^{\min\{C_i - n_i, 0\}} \mu_i^{n_i}} \\ &= C \frac{n_0!}{\prod_{r=1}^{M} ((N_r - n_r)!)} \prod_{i=0}^{M} \frac{1}{\min\{C_i, n_i\}! C_i^{\min\{C_i - n_i, 0\}} \mu_i^{n_i}}, \end{aligned}$$

where we used the fact that only class r customers are in service center r for $1 \leq r \leq N$. This implies that

$$n_{0,r} = N_r - n_r, \quad 1 \leq r \leq N$$

The constant C can then be calculated by summing over all $\boldsymbol{n} \in \mathcal{N}(\mathbf{N})$. Notice that those states are completely determined by $n_1, n_2, \ldots, n_M$. This means we have

$$C = \left[\sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} \cdots \sum_{n_N=0}^{N_M} \frac{n_0!}{\prod_{r=1}^{M} ((N_r - n_r)!)} \prod_{i=0}^{M} \frac{1}{\min\{C_i, n_i\}! C_i^{\min\{C_i - n_i, 0\}} \mu_i^{n_i}} \right]^{-1},$$

where

$$n_0 = \sum_{r=1}^{M} N_r - n_r.$$

4.3. Model parameter derivation

This section concerns the execution of a single benchmark and the data it generates. The obtained execution data of an application can then be used to determine the parameters of the model. Note that an application might invoke multiple cores, thus let M be the number of cores that the application invokes. The parameter values to be determined are then N_i, μ_i, C_i for $1 \leq i \leq M$ and μ_0, C_0. However, we will first need to discuss some assumptions made in the M-core single memory model.

4.3.1. The uniformness assumption

Note that any theorems in previous sections about closed queueing network assumed the system to be in the equilibrium distribution. Specifically for the M-core single memory model this means that the throughput and utilization of the DRAM stays constant throughout the time. This is certainly not the case for a benchmark, because some cores might only be active for a certain part of the benchmark execution. Furthermore, the flow of DRAM requests from the active cores to the memory is not uniform over time.

This problem is illustrated in Figure 4.3. The data originates from the memory intensive `parsec-bodytrack` benchmark. This benchmark uses 3 cores and it can be seen that core 1 is active during almost the entire task execution, while core 0 and core 2 are not. Furthermore, the data from core 1 illustrates the consequence of the non-constant flow of DRAM requests. Core 1 sends around 10000 requests per $1ms$ during the entire task execution, but the latency is much higher in the first part ($0 - 0.1$ seconds) than in the second part of the execution ($0.1 - 0.9$ seconds). Thus in order to accurately model this task, it could be necessary to create two models for core 1: the first model for the first 0.1 seconds of the task and another model for the last 0.8 seconds of the task. However, it could also be the case that core 1 can only be modeled accurately if a different model is generated for every peak in the time interval of $0 - 0.1$ seconds. Similarly, multiple models for core 0 and 2 are probably also necessary. Determining those intervals can actually be very challenging, so we will first determine how long we can make the time intervals before the model starts to become inaccurate.

Figure 4.4 can help to determine this, because it must be the case that there is a direct correlation between the number of DRAM requests currently in the DRAM and the access latency of a new request arriving in the DRAM. Figure 4.4a, 4.4b and 4.4c show that there still exists a correlation, while for time intervals of $10000ns$, the relation between the count and latency has changed. This means for a task taking $1 \times 10^9 ns = 1s$, it is probably accurate to generate a new model for every $1000ns$ of the task. This means each cores needs at most 1 million models and it is probably possible to decrease

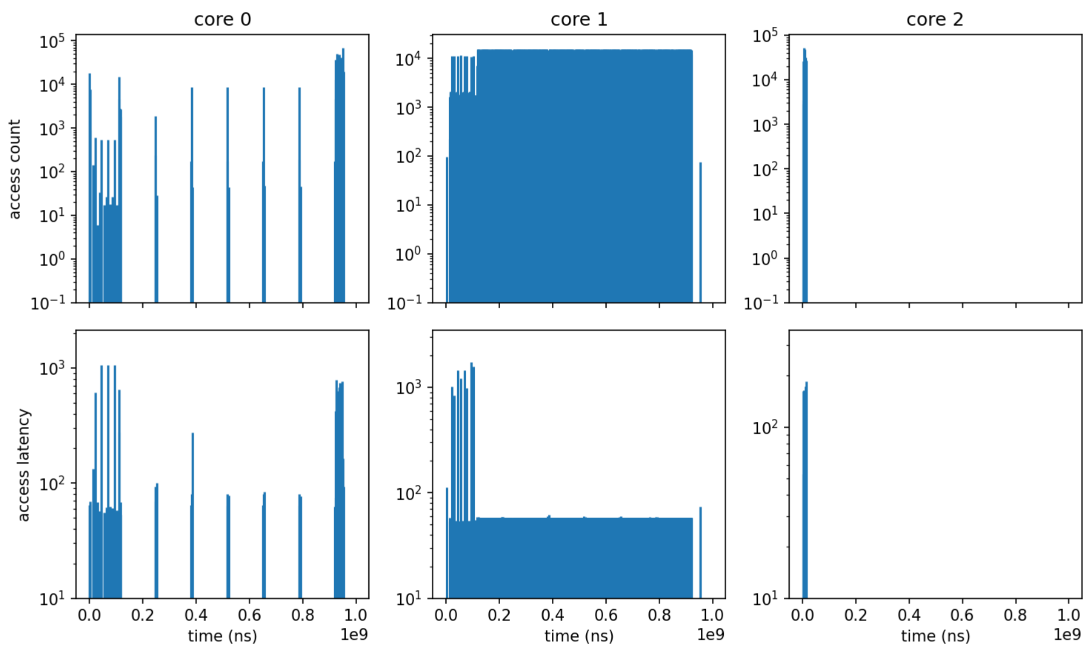


Figure 4.3.: The number of DRAM requests and average latency of a benchmark execution consisting of 3 cores. Each of the time intervals has a length of $1ms$.

this number a lot with some clever optimizations. We will now move onto a way of determining the parameters for each of the models.

4.3.2. The memory parameters

The `sim.cfg` file is the system configuration file that the HotSniper simulation software uses. It contains information about the DRAM memory such as the DRAM service time. This service time must be equal to the expected service time in the model of $\frac{1}{\mu_0}$, which means that

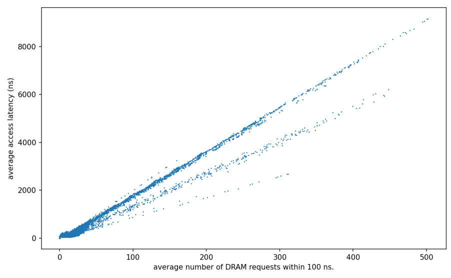
$$\mu_0 = \frac{1}{\text{DRAM service time}}.$$

There are then two different ways of computing the memory capacity C_0.

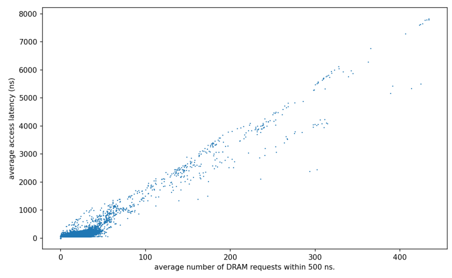
1. The first approach concerns using the `sim.cfg` file. This file contains the
   - number of DRAM memory controllers;
   - bandwidth in B/s of each DRAM memory controller;
   - average service time of a request.

   This information, together with the number of bytes a request receives, can be used to estimate the bandwidth of the DRAM by

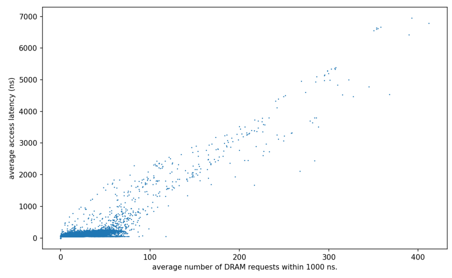
$$C_0 \approx \frac{\text{total bandwidth}}{\text{block size}} \times \text{average service time},$$



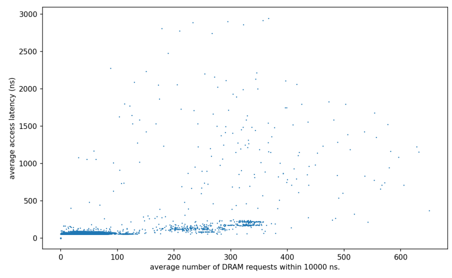
(a) Time intervals of $100ns$.



(b) Time intervals of $500ns$.



(c) Time intervals of $1000ns$.



(d) Time intervals of $10000ns$.

Figure 4.4.: The benchmark execution is divided into time intervals of $100, 500, 1000$ and $10000ns$. The relation between the average number of DRAM requests being present in the DRAM and the average DRAM access latency a new request arriving in the DRAM is plotted.

where

$$\text{total bandwidth} = \text{number of controllers} \times \text{bandwidth of a controller}.$$

2. The second approach can be done by simulating an application. For each core, HotSniper will return the number of DRAM requests, utilization of the DRAM and response time of the core. This can be used to estimate the DRAM bandwidth by

$$C_0 \approx \frac{\text{average number of requests per second}}{\text{utilization}} \times \text{average service time},$$

where

$$\text{average number of requests per second} = \frac{\text{total number of requests}}{\text{response time}}.$$

4.3.3. The core and task parameters

Due to the fact that the core doesn't serve memory requests in real life, it is certainly not trivial how to derive the values of N_i, μ_i, C_i for $1 \le i \le M$. The most important restriction on the parameters is that the total number of memory requests visits must be equal to the total number of DRAM requests that the core sends within the execution of a task. This criteria thus relates the theoretical throughput to the observed throughput of the DRAM. It is formally stated in Criteria 4.3.1.

Criteria 4.3.1 (Throughput)**.** *The throughput criteria is given by*

$$\overline{x}_{0,i}(\mathbf{N}) = \frac{\textit{number of served DRAM requests}}{\textit{time(ns)}} \tag{4.1}$$

The throughput criteria can be satisfied with μ_i for fixed values of N_i and C_i. The relation between the waiting time and different values of C_1 and N_1 is shown in Figure **??**, where the value of μ_1 is chosen to satisfy the throughput criteria for an observed throughput of 0.07. The figure suggests that the waiting time increases in both C_1 and N_1.

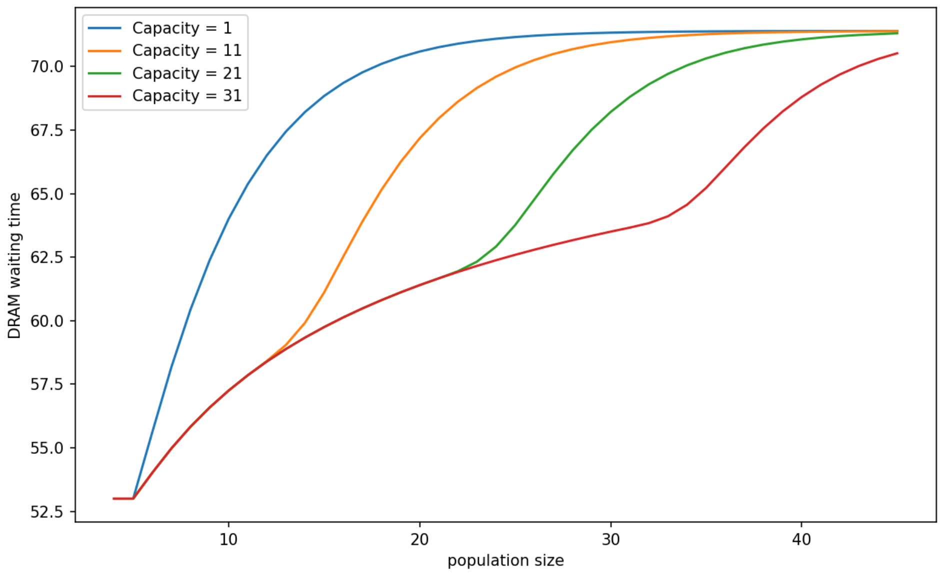


Figure 4.5.: The waiting time in the 1-core single memory model. For multiple values of N_1 (population size) and C_1 (capacity of core 1) the value of μ_1 is found such that the throughput criteria is satisfied for a throughput of 0.07.

Another important criteria is called the waiting time criteria, which relates the theoretical waiting time to the observed access latency of DRAM requests.

Criteria 4.3.2 (Waiting time)**.** *The waiting time criteria is given by*

$$\overline{w}_{0,i}(\mathbf{N}) = \textit{average DRAM access latency.}$$

The two criteria can thus be used to reduce the 'sort of' 3-dimensional space of all possible values for N_i, C_i and μ_i to a 1-dimensional set of values of N_i and C_i. Then the best value for N_i and C_i can be determined by executing the benchmark on different frequencies. If we let f_1 and f_2 be the two clock frequencies, then it makes sense to assume that the value of μ_i is $\frac{f_2}{f_1}$ times higher when run with f_2 as compared to f_1:

$$\mu_i^{(f_2)} = \frac{f_2}{f_1}\mu_i^{(f_1)}, \quad i = 1, \ldots, M.$$

5. Model accurateness and shortcomings

5.1. Task simulation results

5.2. Model parameters and parallel simulation predictions

5.3. Comparisons

6. Conclusion

Bibliography

[Akyildiz and Bolch, 1988] Akyildiz, I. F. and Bolch, G. (1988). Mean value analysis approximation for multiple server queueing networks. *Performance Evaluation*, 8(2):77–91.

[Baskett et al., 1975] Baskett, F., Chandy, K. M., Muntz, R. R., and Palacios, F. G. (1975). Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM (JACM)*, 22(2):248–260.

[Bruell et al., 1984] Bruell, S. C., Balbo, G., and Afshari, P. (1984). Mean value analysis of mixed, multiple class bcmp networks with load dependent service stations. *Performance Evaluation*, 4(4):241–260.

[Reiser and Lavenberg, 1980] Reiser, M. and Lavenberg, S. S. (1980). Mean-value analysis of closed multichain queuing networks. *Journal of the ACM (JACM)*, 27(2):313–322.

Populair summary

A. Proofs