The behaviour of server clusters for different load balancing principles

Author: Márton Mészáros
Supervisor: Illés Antal Horváth

Abstract

A server cluster consists of multiple simultaneously operating servers. Demands arrive to this cluster, which are distributed by a dispatcher between the queues of the different servers by some load balancing principle. The demands wait in these queues until they are served by the servers. These clusters offer an excellent way to model real life systems, like computer servers, the cash registers of shops or transportation networks.

When we examine the performance of these systems, the load balancing principle is a quite important parameter. We will examine five different principles in this paper. The Random assignment principle is the most trivial of them, it simply means that the arriving demands join a queue at random (uniformly). The Join-Idle-Queue (JIQ) principle means that the arriving demand joins an empty queue, if there is one, and otherwise it joins a queue randomly. The Join-Shortest-Queue principle is the most efficient of the discussed ones, it means that the arriving demand joins one of the queues with the fewest demands. A modified version of the previous is the Join-Shortest-Queue(d) (JSQ(d)) principle, which means that the dispatcher chooses d queues at random, and sends the demand to the shortest one. The last principle we discuss is Join-Below-Threshold (JBT), it means that a threshold is assigned to the servers (it may differ between the different servers), and if there are servers with fewer demands than the threshold, then the demand is sent to one of these, and otherwise it is sent randomly.

In this paper we examine a simpler mathematical model of these clusters. We ignore some problems real clusters face, like the cost of communication between the servers and the dispatcher. We also consider the system to be a continuous Markov chain (where the state space is given by the queue lengths of the servers), and for that we need that the arrival and the service of the demands are both Poisson point processes.

It is possible that inside the cluster all the servers have the same properties (in this case we talk about a homogeneous cluster), or there can be different server types as well (in this case the cluster is heterogeneous). The speed of the servers is determined by their service rates, which can be constant, or it can depend on the length of the queues, and in the case of heterogeneous clusters they can differ between the server types.

The state of the cluster converges to a given set of functions (in which the functions represent the number of the servers of a given queue length in a server type in respect to the number of all servers in the cluster, depending on the elapsed time). These functions also converge to a stationary state, as the elapsed time goes to infinity.

In this paper we examine different clusters, especially focusing on their stationary states, simulating them and also numerically solving the differential equations which determine the functions, and this way we can compare the different load balancing principles.

Acknowledgments

I would like to express my deep gratitude to Illés Horváth, my supervisor, for introducing me to queueing theory, teaching me the concepts and theorems which form the basis of our research. This paper, based on our common work, couldn't be made without his patient guidance, encouragement and constructive critiques.
I would also like to express my great appreciation to Dr. László Spissich, who guided me in the field of mathematics while I was a high school student, laid the foundations to my knowledge of higher mathematics and encouraged me to pursue higher education in this discipline.
I would also like to extend my thanks to Bálint Vető, who introduced me to the concept of Markov chains while he was my supervisor in the ÚNKP program, and also taught me about stochastic processes, a topic essential to this paper.

Contents

1 Introduction **4**
1.1 Markov chains 4
1.2 Birth-death processes and queues 5
1.3 Density-dependent population processes 6

2 Server clusters **8**
2.1 Mean system time 12

3 Load balancing principles **13**
3.1 Random assignment 14
3.2 Join-Idle-Queue 15
3.3 Join-Shortest-Queue 17
3.4 Join-Shortest-Queue(d) 19
3.5 Join-Below-Threshold 20

4 Numerical experiments **21**
4.1 Transient mean field diagrams 22
4.2 Other interesting plots 29
4.3 Mean system times 31

5 Conclusion and outlook **32**

1 Introduction

The topic discussed here falls under a field of mathematics called queueing theory. It is a fairly recent field, as the first paper on it was published in 1909 by Danish engineer Agner Krarup Erlang [3]. Erlang modelled the number of telephone calls arriving at an exchange as a Poisson-process. Later he also solved some simpler queue models, such as the M/D/1 and M/D/k queues.

Queueing theory is the study of waiting lines (queues), systems of queues and their behaviours. It is often considered to be a part of operations research as it is often used in business decisions about resources needed to provide a service, but it has its roots in stochastics.

For large-scale service systems, management of resources, and more specifically, load balancing becomes an important issue that can largely effect the performance of such systems.

The contributions of the paper are the following:

1. Providing a high-level mathematical framework for modelling load balancing systems that accommodates several different load balancing principles.

2. Computation of the mean service time in the mean-field limit. Standard computation techniques need to be adapted for discontinuities for some of the load balancing principles; these modified formulas are, to the best of our knowledge, novel.

3. Numerical comparison of the various load balancing principles via simulation and theoretical computations for the mean-field limit.

Certain aspects of load balancing (like the cost of communication overhead between the servers and the dispatcher) are not included in the modelling and are subject to further research.

The rest of the paper is structured as follows: the rest of this section is dedicated to the discussion of mathematical background used in queueing theory to be utilised in the paper. Section 2 describes the general setup of the server cluster we are interested in. Section 3 describes the various load balancing principles. Section 4 contains numerical experiments and comparison of the various load balancing principles, and Section 5 concludes the work.

1.1 Markov chains

As we mentioned it before in the abstract, the queues we model are continuous time Markov chains, more specifically continuous time birth-death processes. A Markov chain is a stochastic model describing a series of events in which the stochastic distribution of the next event only depends on the current state of the system. We usually denote the state space by Ω, which consists of the possible states visited by the chain.

As the next visited state only depends on the current state, for finite state spaces we can work with the transition probability matrix P (in discrete time),

whose (i, j) element $P(i, j)$ is the probability of the chain stepping to state j from state i, or the infinitesimal generator matrix G, whose element $G_{i,j}$ describes the infinitesimal transition rate from i to j. For infinite state spaces, we have operators instead of matrices. That said, we will stick to finite state space in this paper.

If the (finite state space and continuous time) Markov-chain is irreducible, that is, it can get from any state to any state in a finite time, then it has a stationary distribution π which it converges to as time goes to infinity. This means that the probability that the chain will be in state i after n steps converges to $\pi(i)$ as $n \to \infty$.

A continuous Markov chain have its transitions in time as a Poisson point process. This ensures that even the time until the next transition does not depend on the time passed since the last one, as this time is exponential (its parameter can and often does depend on the current space).

A quite common model for continuous Markov chains is the racing clocks model. In this we imagine that when we start at a chain, we start an alarm clock for each possible transitions, and whichever rings first, its transition is going to happen (and the process starts all over for the next state). The clock corresponding to state i will ring in $Y_i \sim \text{Exp}(\mu_i)$ time (where $\mu_i := 0$ if transition to state i is not possible, which corresponds to $Y_i = \infty$, so the clock never rings). μ_i is called the transition's rate. Let us denote the time until the first ring with Y, so $Y = \min_{i \in \Omega}(Y_i)$. Then it follows that

$$Y \sim \text{Exp}\left(\sum_{j \in \Omega} \mu_j\right),$$

and the transition to state i will have a probability

$$\mathbb{P}(Y = Y_i) = \frac{\mu_i}{\sum\limits_{j \in \Omega} \mu_j}.$$

1.2 Birth-death processes and queues

The demands arriving to and leaving a server's queue can be modelled with a birth-death process. A birth-death process is a Markov-chain, in which states are denoted by nonnegative integers and from each state i besides state 0 (from which the chain can only transition to state 1) the chain can transition to state $i - 1$ or $i + 1$.

The rate of transition from state i to $i - 1$ (a death) is denoted by μ_i, and the transition from state i to state $i + 1$ (a birth) is denoted by λ_i. In queueing theory we often utilise a slightly modified version where there is a maximal queue length (state) B, such that for every $i \geq B$, $\lambda_i := 0$. With this, the state space becomes finite with $\Omega = \{0, 1, 2, \ldots, B\}$.

Previously we mentioned M/D/1 and M/D/k queues. These are using Kendall's notation: the first character (here M) marks the type of the arrival

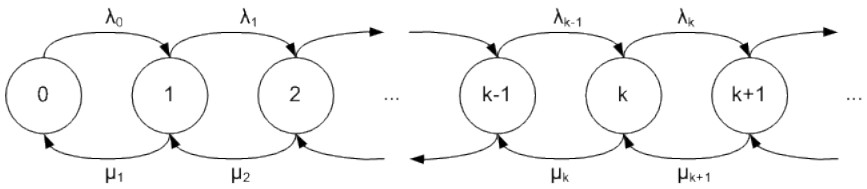

Figure 1: The states and transitions of a birth-death process. [1]

process, the second character (D) marks the serving process, and the third character (1 or k) marks the number of servers associated to the queue. Sometimes a fourth character is also included (e.g. M/D/1/B), marking the buffer size or the maximal queue length (if a demand arrives to a queue with length B, it's simply lost). Queues are basically a special case of birth-death processes, where state transitions are either a demand arriving, making the queue longer by 1, or a demand leaving, making the queue shorter by 1.

In this paper we will mostly examine queues with the service principle First-In-First-Out (FIFO), which means that the server always serves the first demand of a queue, while the other demands wait. Whenever the first demand has finished service, the server immediately starts serving the next demand. Another interesting service principle is Limited Processor Sharing (LPS), where the server can work on multiple demands simultaneously. The maximum number of demands served simultaneously is called the multi-programming level (MPL).

M as the first or second character of the queue's type stands for Markov. If the Markov-property holds for arrivals, then the arrival of the demands is a Poisson point process with some parameter λ, and if the second character is M, the service time of a demand is exponential with some parameter μ (or, in case the service rate depends on the queue length, it's μ_i for queue length i).

D stands for deterministic interarrival or service times, and G is also used for general time distributions. Only the M/M queues are Markov-chains. In this paper we will focus on clusters of M/M/1/B queues.

1.3 Density-dependent population processes

Our server clusters, which are groups of co-dependent servers with a common dispatcher, can be best described as density-dependent population processes. In these the population consists of N interacting components, each of which is in a state from a finite set of local states S. The global state of the system is defined as the total number of individuals in each state, that is, a vector $X^N \in \{0, 1, \ldots, N\}^{|S|}$ with $X_1^N + \cdots + X_{|S|}^N = N$. The normalized global state of the system can be defined as

$$x^N = \frac{X^N}{N},$$

so $x^N \in [0,1]^S$ with $x_1^N + \cdots + x_{|S|}^N = 1$.
Each component acts as a continuous Markov chain. The rate of its transition from $i \in S$ to $j \in S$ is r_{ij}^N (for $i \neq j$). The rates are assumed to be *density dependent*, that is

$$r_{ij}^N = r_{ij}(x)$$

for some function $r_{ij} : [0,1]^{|S|} \to [0,\infty]$. x is the $|S|$ dimensional vector with elements x_i^N for $i \in \{1,2,\ldots,|S|\}$. In the classic setup defined by Kurtz [7, 8], the functions r_{ij} are Lipschitz-continuous and independent of N. With this setup $x^N(t)$ is a continuous time Markov-chain. We define the mean field equation of the system as the following:

$$\frac{\mathrm{d}}{\mathrm{d}t} v_i(t) = \sum_{j \in S} v_j(t) r_{ji}(v(t)), \quad i \in S, \tag{1}$$

where

$$r_{ii} := -\sum_{j \in S, j \neq i} r_{ij}.$$

and

$$x_i(0) \to v_i(0) \quad (\text{for } i \in S), \quad \text{in probability.}$$

Lipschitz-continuity guarantees existence and uniqueness of the solution of (1). The following result of Kurtz states mean field convergence in the transient regime under the same condition [7, 8, 4]:

Theorem 1 (Transient mean field convergence)**.** *Assuming r_{ij} $(i,j \in S)$, are Lipschitz-continuous and*

$$x_i(0) \to v_i(0) \quad i \in \{0,1,\ldots,|S|\}, \quad \textit{in probability,}$$

then for any $T > 0$ we have

$$\lim_{N \to \infty} \mathbb{P}\left(\sup_{t \in [0,T]} \|\bar{\mathbf{x}}^N(t) - \mathbf{v}(t)\| > \epsilon \right) = 0$$

Kurtz also proved that the standard deviation of x^N is of order $\frac{1}{\sqrt{N}}$, which will be of use later.

We also have stationary mean field convergence.

Theorem 2 (Stationary mean field convergence)**.** *Given the following assumptions:*

- *r_{ij} are Lipschitz-continuous,*
- *the Markov process $x^N(t)$ has a unique stationary distribution π^N for each N, and*
- *(1) has a unique stable attractor $(\nu_0,\ldots,\nu_B)$,*

we have that the probability measure π^N on S converges in probability to the Dirac measure concentrated on ν.

Theorems 1 and 2 have been generalized further in several ways during recent years. Benaïm and Le Boudec elaborated a framework applicable for a wider range of stochastic processes, which also allows the r_{ij} functions to have a mild dependency on N [2].

The condition on Lipschitz-continuity can also be weakened. For discontinuous r_{ij}'s, 1 turns into a differential inclusion. A formal setup for differential inclusions is quite technical in general, which is omitted from the present paper. For a fully detailed setup, we refer to [5], specifically Theorems 4 and 5 for the corresponding version of Theorem 1 for discontinuous drifts, and [13], Theorem 3.5 and Corollary 3.9 for the corresponding version of Theorem 1.

For a corresponding version of Theorem 2, we refer to Gast and Gaujal, where one additional condition is that the unique attractor lies inside a domain where r_{ij} is continuous [5].

The applicability of Theorems 1 and 2 will be addressed more in Section 2.

From Theorem 2 it also follows that

$$\mathbb{E}(\pi^N) \to \nu,$$

so ν can be used as an approximation for $\mathbb{E}(\pi^N)$ for large N. $\mathbb{E}(\pi^N)$ here is basically an $|S|$ dimensional vector of distributions, which converges to a constant $|S|$ dimensional vector, which can be interpreted as a distribution on S, and is the stable attractor ν.

2 Server clusters

As it was stated previously, the server clusters we examine in this paper are systems of N queues of type M/M/1/B with a common dispatcher and service principle FIFO. The initial state of the observed servers will be assumed to be empty. In the cases we examine, the arrival rate to the cluster is independent of its state and is constant, with its value being $N\lambda$ (such that the average arrival rate is λ for every server). The service rate of the servers can be constant or it can depend on the current queue length of the server. We usually speak of service rate curves when the service principle is LPS, and the server can serve batches of demands simultaneously more efficiently than one-by-one.

In this paper, we will discuss server clusters of different sizes and examine the limit object as $N \to \infty$, referred to as the mean-field limit (in accordance with Section 1.3).

The cluster may have K different server types with different service rate curves, denoted by $\mu_i^{(k)}$, where $i \in \{0, 1, \ldots, B^{(k)}\}$ is the queue length, and $k \in \{1, 2, \ldots, K\}$ denotes the type of the server. We assume K is independent from N.

For service rate curves, it is natural to assume that the rate is growing monotonously with the queue length (if not, we can set the MPL at the turning

point, so the server does not slow down), and that the per-demand service rate decreases, so the following holds:

$$\mu_1^{(k)} \leq \mu_2^{(k)} \leq \mu_3^{(k)} \leq \ldots, \quad \mu_1^{(k)} \geq \frac{\mu_2^{(k)}}{2} \geq \frac{\mu_3^{(k)}}{3} \geq \ldots \quad k \in \{1, 2, \ldots, K\} \quad (2)$$

As stated before, the server cluster basically functions as a density-dependent population process, where the current state of a server is simply the number of the demands in its queue. The global state is usually denoted by the numbers

$$X_i^{(k),N}(t), \qquad \left(0 \leq i \leq B^{(k)}, \qquad 1 \leq k \leq K\right),$$

where $X_i^{(k),N}(t)$ is the number of servers with i demands in its queue at time t. We will, however use its normalized version

$$x^N(t) = x_i^{(k),N}(t), \qquad \left(0 \leq i \leq B^{(k)}, \qquad 1 \leq k \leq K\right),$$

where

$$x_i^{(k),N}(t) = \frac{X_i^{(k),N}(t)}{N}.$$

The number of servers of type k is denoted by N_k and the ratio of each server type is denoted by

$$\gamma_k^N = \frac{N_k}{N}, \qquad k = 1, \ldots, K.$$

γ_k^N may depend on N, but we will assume they converge to some fixed values γ_k as $N \to \infty$. We also want the cluster to be stable, so

$$\lambda < \sum_{k=1}^{K} \gamma_k^N \mu_B^{(k)}. \quad (3)$$

(We note that due to the finite buffer size assumption, the cluster is technically always stable, but we will nevertheless assume (3) holds.)

The evolution of $x^N(t)$ can be formally defined using Poisson representation. Let

$$P_{i\to(i+1),k}(t), \quad 0 \leq i \leq B^{(k)} - 1$$
$$P_{i\to(i-1),k}(t), \quad 1 \leq i \leq B^{(k)}$$

denote independent Poisson processes with a rate of 1. $P_{i\to(i+1),k}(t)$ corresponds the arrival of a new demand to a queue of type k with length i, and $P_{i\to(i-1)}(t)$ corresponds to a demand leaving a queue of type k with length i. The Poisson

representation of $x^N(t)$ is then

$$\begin{aligned} x_i^{(k),N}(t) =& \frac{1}{N} P_{(i-1)\to i,k}\left(N\int_0^t \lambda f_{i-1}^{(k)}(x^N(s))\mathrm{d}s\right) \\ &- \frac{1}{N} P_{i\to(i+1),k}\left(N\int_0^t \lambda f_i^{(k)}(x^N(s))\mathrm{d}s\right) \\ &+ \frac{1}{N} P_{(i+1)\to i,k}\left(N\int_0^t \mu_{i+1}^{(k)} x_{i+1}^{(k),N}(s)\mathrm{d}s\right) \\ &- \frac{1}{N} P_{i\to(i-1),k}\left(N\int_0^t \mu_i^{(k)} x_i^{(k),N}(s)\mathrm{d}s\right), \end{aligned} \tag{4}$$

where $f_i^{(k)}(x^N(t))$ is the probability of a new arriving demand to enter a queue with length i of type k. The $f_i^{(k)}$ functions depend on the load-balancing principle, which will be described later. Formally, $f_i^{(k)}$ are functions defined on the normalized state $x^N(t)$, which are all contained in the domain

$$\{x : x \in \mathbb{R}^{\sum_{k=1}^K (B^{(k)}+1)}, \sum_{k=1}^K \sum_{j=0}^{B^{(k)}} x_j^{(k)} = 1\}.$$

The four possible changes in the number of queues with length i which appear in (4) are:

- A demand arrives to a queue with length $i-1$
- A demand arrives to a queue with length i
- A demand leaves a queue with length $i+1$
- A demand leaves a queue with length i

Of course on the border, certain changes can not happen: there is no service in empty queues, that is, $\mu_0^{(k)} = 0\,(k = 1, \ldots, K)$, and no arrival to full queues, $f_{B^{(k)}}(.) \equiv 0\,(k = 1, \ldots, K)$.

The general mean field equation corresponding to (4) is

$$\begin{aligned} v_i^{(k)}(t) = v_i^{(k)}(0) &+ \int_0^t \lambda f_{i-1}^{(k)}(v(s))\mathrm{d}s - \int_0^t \lambda f_i^{(k)}(v(s))\mathrm{d}s \\ &+ \int_0^t \mu_{i+1}^{(k)} v_{i+1}^{(k)}(s)\mathrm{d}s - \int_0^t \mu_i^{(k)} v_i^{(k)}(s)\mathrm{d}s. \end{aligned} \tag{5}$$

in integral form, or

$$\frac{\mathrm{d}}{\mathrm{d}t} v_i^{(k)}(t) = \lambda f_{i-1}^{(k)}(v(t)) - \lambda f_i^{(k)}(v(t)) + \mu_{i+1}^{(k)} v_{i+1}^{(k)}(t) - \mu_i^{(k)} v_i^{(k)}(t) \tag{6}$$

in differential form. An empty initial cluster corresponds to the initial condition

$$v_i^{(k)}(0) = \begin{cases} \gamma_k & \text{for } i = 0, \\ 0 & \text{otherwise.} \end{cases}$$

In the integral form, if we assume that we start from an empty cluster, we can leave $v_i^{(k)}(0)$ out of the equation, as its value is 0.

Theorem 1 applies to these systems whenever the $f_i^{(k)}$ functions are Lipschitz-continuous. It turns out that the conditions of the general version of Theorem 1 are mild enough so that transient mean field convergence holds for all the discontinuous choices of $f_i^{(k)}$ in the present paper.

Theorem 2 applies whenever $f_i^{(k)}$ are Lipschitz-continuous. In the discontinuous setting, the most relevant question is whether the $f_i^{(k)}$ functions are continuous at the unique fixed point ν or not. If ν lies inside a region where $f_i^{(k)}$ are continuous, then the conclusion of Theorem 1 applies. However, when the $f_i^{(k)}$ functions are discontinuous at ν, Theorem 2 does not apply; in fact, little is known in this case rigorously. Based on this, it makes sense to distinguish the following 3 cases that will cover all of the models examined:

1. the functions $f_i^{(k)}$ are Lipschitz-continuous;
2. the functions $f_i^{(k)}$ are discontinuous in general, but continuous at ν;
3. the functions $f_i^{(k)}$ are discontinuous at ν.

When the functions $f_i^{(k)}$ are Lipschitz-continuous, or discontinuous in general, but continuous at ν, the equations for the mean field stationary distribution can be obtained from (6) by setting $\frac{\mathrm{d}}{\mathrm{d}t} v_i^{(k)}(t) = 0$:

$$\begin{aligned} 0 = \lambda f_{i-1}^{(k)}(v(t)) - \lambda f_i^{(k)}(v(t)) + \mu_{i+1}^{(k)} v_{i+1}^{(k)}(t) - \mu_i^{(k)} v_i^{(k)}(t) \\ i \in \{0, \dots, B^{(k)}\}, \qquad k \in \{1, \dots, K\} \end{aligned} \tag{7}$$

which are equivalent to the dynamic balance equations

$$\mu_i^{(k)} \nu_i^{(k)} = \lambda f_{i-1}^k(\nu), \qquad i \in \{1, \dots, B^{(k)}\}, \qquad k \in \{1, \dots, K\}. \tag{8}$$

We also have equations for the ratio of each server type

$$\sum_{i=0}^{B^{(k)}} \nu_i^{(k)} = \gamma_k \qquad k \in \{1, \dots, K\}, \tag{9}$$

and the system (8) + (9) can be solved jointly.

When the $f_i^{(k)}$ are discontinuous at ν, more considerations are needed to derive the dynamic balance equations. This will be addressed separately for each load balancing principle.

2.1 Mean system time

We can inspect a wide variety of parameters in a server cluster to investigate its efficiency. One of the most obvious ones is the mean system time: it denotes the average time a demand spends in the system between its arrival and service. We aim to calculate the mean system time H in the stationary mean field regime. Let $H_{i,j}^{(k)}$ denote the mean time until service for a demand that is in position i in a queue with j demands of type k (so $1 \leq i \leq j \leq B^{(k)}$, $1 \leq k \leq K$). In the case of constant service curves, $H_{i,j}^{(k)} = \frac{i}{\mu^{(k)}}$ holds. For non-constant service rates curves however, the service rate may change due to later arrivals, so we need to keep track of both the length of the queue and the position of the demand within it. We will write a system of linear equations using total expectation and the Markov property. For simplicity, we assume FIFO service principle, but due to Little's law, this assumption does not affect the mean system time.

$$
\begin{aligned}
H_{i,j}^{(k)} &= \frac{1}{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}} + \frac{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)}}{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}} H_{i,j+1}^{(k)} + \\
&\qquad \frac{\mu_j^{(k)}}{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}} H_{i-1,j-1}^{(k)} \qquad (2 \leq i \leq j \leq B^{(k)} - 1), \\
H_{i,B^{(k)}}^{(k)} &= \frac{1}{\mu_{B^{(k)}}^{(k)}} + H_{i-1,B^{(k)}-1}^{(k)} \qquad (2 \leq i \leq B^{(k)}), \qquad (10) \\
H_{1,j}^{(k)} &= \frac{1}{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}} + \frac{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)}}{\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}} H_{1,j+1}^{(k)} \\
&\qquad (1 \leq j \leq B^{(k)} - 1), \\
H_{1,B^{(k)}}^{(k)} &= \frac{1}{\mu_{B^{(k)}}^{(k)}}.
\end{aligned}
$$

This equation system (which can actually be solved individually for each k for $1 \leq k \leq K$) makes use of the standard one step argument. We focus on a single queue of a given type k in the mean field limit while assuming the environment to be stationary, and look for the next possible change in that queue. Demands arrive to type k servers of queue length j with a rate of $\lambda f_j^{(k)}(\nu)$, and $1/\nu_j^{(k)}$ of these will be sent to one specific server, so the arrival rate will be $\lambda f_j^{(k)}(\nu)/\nu_j^{(k)}$, while the service rate is $\mu_j^{(k)}$, so the rate of any change for a queue of length j is $\lambda f_j^{(k)}(\nu)/\nu_j^{(k)} + \mu_j^{(k)}$. The change will either increase or decrease the length of the queue by 1, and we can apply total expectation.

We first compute ν from either the balance equations (7) when possible, or by numerically solving the transient mean field equations (6) and setting t large enough. Then we put ν in (10), which, for ν given, is linear for $H_{i,j}^{(k)}$. Once (10) is solved, the mean system time H is just a linear combination of the values $H_{j,j}^{(k)}$ according to the probabilities with which a demand will be scheduled to a

queue of length $j-1$ of a k-type server, that is,

$$H = \sum_{j=1}^{B^{(k)}} \sum_{k=1}^{K} f_{j-1}^{(k)}(\nu) H_{j,j}^{(k)}. \tag{11}$$

(10) and (11) are only valid if the functions $f_i^{(k)}$ are continuous at ν. In other cases, we may need to tweak the formulas. We will provide the version of (10) and (11) on a case-by-case basis whenever the functions $f_i^{(k)}$ are discontinuous at ν. These versions will be heuristic in the sense that no formal rigorous proof will be provided, but the results nevertheless agree with the results from the simulations.

3 Load balancing principles

Load balancing principle describes the method the dispatcher uses to distribute the arriving demands relatively evenly between the servers. It is quite important in large scale systems where the resources such as computing capacity are distributed between a large number of individual servers, where it can make a big difference in the efficiency of the system.

The general goal of load balancing is to avoid long queues, directing incoming jobs to shorter queues instead.

There are several load balancing principles in use. Perhaps the simpler group of these are static policies which do not consider the state of the system, they only focus on the incoming demands. One example would be the round-robin load balancing policy, where incoming demands are directed to the next server cyclically. These are generally easier to operate, as they require minimal communication with the servers. Out of the principles observed in this paper, Random assignment falls into this category.

However, often there is a certain degree of randomness in these chains, so principles which take into account the current state of the system can be more efficient. In real clusters, there is a trade-off: complicated policies require more communication and computation, generating a higher overhead communication cost, slowing down the entire system. That said, in the mathematical framework we present, the cost of communication overhead is not modeled. Including the cost of overhead communication to provide an analytical framework for more realistic models is subject to further research.

In some systems it is possible to reassign demands that have been already assigned to new servers. In some systems, it is also possible that several servers "team up" to serve a single job. In our setting, we do not explore these options, and stick to a scenario where all demands are assigned to a single server immediately upon arrival. On the other hand, in addition to the usual FIFO service principle, the framework does allow for limited processor sharing, where a single server can serve multiple jobs simultaneously.

In this paper we will examine 5 load balancing principles:

- Random assignment, where demands are distributed randomly. With this principle, there is no actual load balancing. This principle will serve mostly as a baseline for comparison.
- Join-Idle-Queue, where demands are directed to idle queues if possible. A relatively recent idea [10], further explored in [12].
- Join-Shortest-Queue, where demands are directed to the server with the fewest number of jobs waiting in queue. One of the earliest load balancing policies that has been widely used for decades [9]. It provides very even balancing, but at the cost of high overhead communication, as the dispatcher needs to keep track of the queue length in every single server at all times.
- Join-Shortest-Queue(d), where demands are directed to the server with the fewest number of jobs waiting in queue from among d servers selected randomly. Also referred to as power-of-d, this is a version of JSQ that aims to reduce communication overhead at the cost of less strict balancing. It has been thoroughly explored, and has certain asymptotical optimality properties already for $d = 2$ [11].
- Join-Below-Threshold, where demands are directed to servers with a queue length below a prescribed threshold [6].

All of the above principles are based on natural intuitions that aim towards directing demands to shorter queues, but they differ in the details and execution of doing so. In this section, we overview these load balancing principles from the literature. We present a high-level mathematical framework based on Poisson representation that is applicable to all of them, with the only difference being the $f_i^{(k)}(.)$ functions. For each load balancing policy, we identify $f_i^{(k)}(.)$, then write the mean field equations corresponding to (5), then also identify the mean field stationary distribution ν whenever available explicitly, and, whenever the $f_i^{(k)}(.)$ functions are discontinuous at ν, we also rewrite the formulas (10) and (11) so that they can be used to compute the mean system time.

3.1 Random assignment

This is the most simple principle that we observe, and it does not lead to any balancing. With this setup the queues basically operate, and thus can be analyzed independently of each other. For random assignment,

$$f_i^{(k)}(x) = x_i^{(k)}, \quad k \in \{1, \dots, K\},$$

and accordingly, the mean field equation is

$$\begin{aligned} v_i^{(k)}(t) = &\int_0^t \lambda v_{i-1}^{(k)}(s)\mathrm{d}s - \int_0^t \lambda v_i^{(k)}(s)\mathrm{d}s \\ &+ \int_0^t \mu_{i+1} v_{i+1}^{(k)}(s)\mathrm{d}s - \int_0^t \mu_i v_i^{(k)}(s)\mathrm{d}s. \end{aligned} \tag{12}$$

The mean field balance equations, obtained from (8), are

$$\mu_i^{(k)}\nu_i^{(k)} = \lambda\nu_{i-1}^{(k)} \qquad k \in \{1,\dots,K\}, \qquad i \in \{1,\dots,B\}. \tag{13}$$

Solving (13) (along with (9)) gives the mean field stationary distribution

$$\nu_i^{(k)} = c\gamma_k \prod_{j=1}^{i} \lambda/\mu_j^{(k)}, \qquad i \in \{0,\dots,B^{(k)}\},$$

which is in accordance with the queues being independent.

Since the rates $f_i^{(k)}$ are continuous, (10) and (11) can be used to compute the mean system time H.

3.2 Join-Idle-Queue

For Join-Idle-Queue (JIQ), incoming demands are assigned to an idle server at random. If none of the servers are idle, a server is selected at random.

For JIQ, using the notation

$$y_0 = \sum_{k=1}^{K} x_0^{(k)},$$

we have

$$f_i^{(k)}(x) = \begin{cases} \frac{x_i^{(k)}}{y_0} & \text{if } i = 0,\ y_0 > 0, \\ 0 & \text{if } i > 0,\ y_0 > 0, \\ x_i^{(k)} & \text{if } y_0 = 0. \end{cases} \tag{14}$$

This system has been addressed in [12] for constant service rate curve and a homogeneous cluster.

The structure of the mean field stationary distribution ν depends on the relation between λ and $\sum_{k=1}^{K}\gamma_k\mu_1^{(k)}$. When $\lambda < \sum_{k=1}^{K}\gamma_k\mu_1^{(k)}$, there will always be idle queues in the mean field stationary limit, so all demands will be directed to idle queues. $\nu_0^{(k)}$ and $\nu_1^{(k)}$ will be the only nonzero values in ν (for $k \in \{1,\dots,K\}$), and from (8) we have

$$\mu_1^{(k)}\nu_1^{(k)} = \lambda\frac{\nu_0^{(k)}}{\sum_{k=1}^{K}\nu_0^{(k)}}. \tag{15}$$

We do not have an explicit solution of (15), but it can be solved numerically, and numerical experiments suggest a single fixed point ν. In this region, the functions f_i are continuous, so (10) and (11) can be used to compute the mean system time H:

$$H = \sum_{k=1}^{K} \frac{\nu_0^{(k)}}{\sum_{k=1}^{K}\nu_0^{(k)}} H_{1,1}^{(k)}.$$

For $\lambda = \sum_{k=1}^{K} \gamma_k \mu_1^{(k)}$, the mean field stationary distribution is concentrated on queues of length 1, so we simply have

$$\nu_1^{(k)} = \gamma_k, \quad k \in (1, \dots, K). \tag{16}$$

The functions $f_i^{(k)}$ are discontinuous at ν, so (10) and (11) does not apply. Instead, in the dynamic balance, whenever a queue of length 1 finishes service, a new demand will enter immediately. With this, we can write the equivalent of (10) for JIQ:

$$\begin{aligned} H_{i,j}^{(k)} &= \frac{1}{\mu_j^{(k)}} + H_{i-1,j-1}^{(k)} \qquad (2 \le i \le j \le B^{(k)}), \\ H_{1,j}^{(k)} &= \frac{1}{\mu_j^{(k)}} \qquad (1 \le j \le B^{(k)} - 1), \end{aligned} \tag{17}$$

As we can see it is basically equivalent with (10) in this case, because the discontinuity would only affect the arrival rate, and it is multiplied by 0 for every relevant term. In the mean field limit, all demands go to queues of length 0 (which will then stay at length 1 for a positive amount of time), and there are no queues with 2 or more demands. Accordingly, instead of (11), we have

$$H = \sum_{k=1}^{K} \frac{\mu_1^{(k)} \nu_1^{(k)}}{\lambda} H_{1,1}^{(k)}. \tag{18}$$

In case $\lambda > \sum_{k=1}^{K} \gamma_k \mu_1^{(k)}$, there will be no idle queues, so $\nu_0^{(k)} = 0$ for $k \in (1, \dots, K)$. We note that $f_i^{(k)}$ are discontinuous at any point with $\sum_{k=1}^{K} \nu_0^{(k)} = 0$ and $\sum_{k=1}^{K} \nu_1^{(k)} > 0$; an intuitive explanation of this discontinuity is the following. Whenever a server with a single demand finishes service, it will become idle. In the mean field limit, a demand will enter the idle queue instantly, so once again, we do not observe idle queues for any positive amount of time. However, similar to the $\lambda = \sum_{k=1}^{K} \gamma_k \mu_1^{(k)}$ case, a positive percentage of all incoming demands will go to an idle queue. To compute this percentage, we once again observe that in the mean field stationary distribution, service from queues of length 1 has to be balanced out completely by arrivals to idle queues.

The total service rate in queues of type k of length 1 is $\mu_1^{(k)} \nu_1^{(k)}$, which is thus completely balanced out by an equal amount of arrivals, and the remaining arrival rate $(\lambda - \sum_{k=1}^{K} \mu_1^{(k)} \nu_1^{(k)})$ is distributed randomly. For longer queues, there are no discontinuities. Accordingly, the dynamic balance equations are

$$\left(\lambda - \sum_{k=1}^{K} \mu_1^{(k)} \nu_1^{(k)} \right) \nu_i^{(k)} = \mu_{i+1}^{(k)} \nu_{i+1}^{(k)}, \quad i \in (1, \dots, B^{(k)} - 1). \tag{19}$$

The system (19) is nonlinear, but can be solved numerically. Then we can write a modified version of (10) for the calculation of $H_{i,j}^{(k)}$. For this, we introduce $z_1 =$

$\sum_{k=1}^{K} \mu_1^{(k)} \nu_1^{(k)}$, which stands for the portion of arrivals which is balanced out by the service in servers with queue length 1, thus becoming idle and receiving a new arrival instantly. According to JIQ policy, the remaining arrival rate $\lambda - z_1$ is distributed randomly for the rest of the system. Accordingly, (10) becomes

$$\begin{aligned}
H_{i,j}^{(k)} &= \frac{1}{(\lambda - z_1) + \mu_j^{(k)}} + \frac{(\lambda - z_1)}{(\lambda - z_1) + \mu_j^{(k)}} H_{i,j+1}^{(k)} + \\
&\qquad \frac{\mu_j^{(k)}}{(\lambda - z_1) + \mu_j^{(k)}} H_{i-1,j-1}^{(k)} \qquad (2 \le i \le j \le B^{(k)} - 1), \\
H_{i,B^{(k)}}^{(k)} &= \frac{1}{\mu_{B^{(k)}}^{(k)}} + H_{i-1,B^{(k)}-1}^{(k)} \qquad (2 \le i \le B^{(k)}), \qquad (20) \\
H_{1,j}^{(k)} &= \frac{1}{(\lambda - z_1) + \mu_j^{(k)}} + \frac{(\lambda - z_1)}{(\lambda - z_1) + \mu_j^{(k)}} H_{1,j+1}^{(k)} \quad (1 \le j \le B^{(k)} - 1), \\
H_{1,B^{(k)}}^{(k)} &= \frac{1}{\mu_{B^{(k)}}^{(k)}}.
\end{aligned}$$

To obtain the mean system time H, instead of (11), we now have

$$H = \sum_{k=1}^{K} \frac{\mu_1^{(k)} \nu_1^{(k)}}{\lambda} H_{1,1}^{(k)} + \left(1 - \sum_{k=1}^{K} \frac{\mu_1^{(k)} \nu_1^{(k)}}{\lambda}\right) \sum_{k=1}^{K} \sum_{j=2}^{B^{(k)}} \nu_{j-1}^{(k)} H_{j,j} \qquad (21)$$

since $\frac{\sum_{k=1}^{K} \mu_1^{(k)} \nu_1^{(k)}}{\lambda}$ is the portion of the arrival rate that is used to balance out the service in queues of length 1 and the remaining portion of the incoming rate is distributed randomly.

3.3 Join-Shortest-Queue

For Join-Shortest-Queue (JSQ), incoming demands are assigned to the shortest queue from among all queues; in case of multiple shortest queues of the same length, one is selected randomly.

For JSQ,

$$f_i^{(k)}(x) = \begin{cases} 0 & \text{if } \exists\, i' < i\ \exists\, k' : x_{i'}^{(k')} > 0, \\ 0 & \text{if } \sum_{k=1}^{K} x_i^{(k)} = 0, \\ \frac{x_i^{(k)}}{\sum_{k=1}^{K} x_i^{(k)}} & \text{otherwise.} \end{cases}$$

For the stationary mean field analysis, let i_0 denote the smallest i for which

$$\sum_{k=1}^{K} \gamma_k \mu_i^{(k)} \ge \lambda.$$

Such an i exists if the stability condition (3) holds. Then the mean field stationary distribution ν will be concentrated on queues of length i_0 and $i_0 - 1$: starting from an arbitrary point, queues shorter than $i_0 - 1$ will receive the entire load of arrivals, which is larger than they can process, so these queues will "fill up" to level $i_0 - 1$, while queues longer than i_0 do not receive any load at all, so these queues will go down, until they reach level i_0.

The total service rate in queues of length (i_0-1) is $\sum_{k=1}^{K} \mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)}$, which is completely balanced out by an equal amount of arrivals. The remaining arrival rate $(\lambda - \sum_{k=1}^{K} \mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)})$ goes to queues of length $i_0 - 1$, with the queue type k chosen at random with probabilities proportional to $\nu_{i_0-1}^{(k)}$. For each server type k, these arrivals are balanced out by the service in queues of type k and length i_0, leading to the balance equations

$$\mu_{i_0}^{(k)} \nu_{i_0}^{(k)} = \left(\lambda - \sum_{k=1}^{K} \mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)} \right) \frac{\nu_{i_0-1}^{(k)}}{\sum_{k=1}^{K} \nu_{i_0-1}^{(k)}} \qquad k \in (1, \dots, K), \tag{22}$$

which, along with (9), gives a (nonlinear) system of equations for ν, which can be solved numerically.

To compute the mean system time H, we utilise the following equation system to compute $H_{i,j}^{(k)}$. Similarly to JIQ, we introduce

$$z_{i_0-1} = \sum_{k=1}^{K} \mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)},$$

which plays a similar role to its JIQ counterpart, standing for the portion of arrivals which is balanced out by the service in servers with queue length $i_0 - 1$. Whenever a server with queue length $i_0 - 1$ finishes service, it will become the single shortest queue and receives a new arrival instantly. Rate $\lambda - z_{i_0-1}$ remains for the rest of the system, which will be directed entirely to queues of length $i_0 - 1$. To ease notation, we also introduce

$$y_{i_0-1} = \sum_{k=1}^{K} \nu_{i_0-1}^{(k)}.$$

Then

$$
\begin{aligned}
H_{i,j}^{(k)} &= H_{i,j+1}^{(k)} \qquad (1 \le i \le j < i_0 - 1),\\
H_{1,i_0-1}^{(k)} &= \frac{1}{((\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}) + \mu_{i_0-1}^{(k)}} + \\
&\qquad \frac{(\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}}{((\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}) + \mu_{i_0-1}^{(k)}} H_{1,i_0}\\
H_{i,i_0-1}^{(k)} &= \frac{1}{((\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}) + \mu_{i_0-1}^{(k)}} + \\
&\qquad \frac{(\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}}{((\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}) + \mu_{i_0-1}^{(k)}} H_{1,i_0} + \\
&\qquad \frac{\mu_{i_0-1}^{(k)}}{((\lambda - z_{i_0-1})/y_{i_0-1}^{(k)}) + \mu_{i_0-1}^{(k)}} H_{i-1,i_0-2} \quad (2 \le i \le i_0 - 1)\\
H_{1,j}^{(k)} &= \frac{1}{\mu_{B^{(k)}}^{(k)}} \qquad (i_0 - 1 < j \le B^{(k)}),\\
H_{i,j}^{(k)} &= \frac{1}{\mu_j^{(k)}} + H_{i-1,j-1} \quad (i_0 - 1 < j \le B^{(k)}, \quad 1 \le i \le j).
\end{aligned}
\tag{23}
$$

The first equation in (23) addresses the fact that if a server has fewer than $i_0 - 1$ demands in it, it will immediately fill up to $i_0 - 1$ demands. We also adjust the effective arrival rate to $\lambda - z_{i_0-1}$, similarly to JIQ. If $i_0 = 1$, the $f_i^{(k)}$ are continuous at ν, so we can use (10) instead of (23). If $i_0 = 2$, there will of course not be any equation with the condition $(2 \le i \le i_0 - 1)$.
If the functions $f_i^{(k)}$ are continuous at ν, we can use (11) to calculate the mean system time. In case $i_0 = 1$, ν is in the inside of a continuous domain of the functions $f_i^{(k)}$, so this is the case, and (11) simplifies to

$$H = \sum_{k=1}^{K} \frac{\nu_0^{(k)}}{\sum_{k=1}^{K} \nu_0^{(k)}} H_{1,1}^{(k)}.$$

On the other hand, if $i_0 > 1$, the functions f_i are not continuous at ν, and (11) is not applicable; instead, we have

$$H = \sum_{k=1}^{K} \frac{\mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)}}{\lambda} H_{i_0-1,i_0-1}^{(k)} + \left(1 - \sum_{k=1}^{K} \frac{\mu_{i_0-1}^{(k)} \nu_{i_0-1}^{(k)}}{\lambda}\right) \sum_{k=1}^{K} \frac{\nu_{i_0-1}^{(k)}}{\sum_{k=1}^{K} \nu_{i_0-1}^{(k)}} H_{i_0,i_0}^{(k)}.$$

3.4 Join-Shortest-Queue(d)

JSQ(d) is a version of JSQ where the dispatcher first selects d servers randomly, and dispatches the incoming demand to the shortest from among the d queues. If we set d to 1, we get Random assignment, and if we set d to N, we get the JSQ

principle. Of course in the mean field case as $N \to \infty$, the $f_i^{(k)}$ functions we get with $d \to \infty$, will be discontinuous (like for JSQ), although they are continuous for any finite d. We will show some figures that depicts this convergence of JSQ(d) to JSQ in Section 4.

For JSQ(d), we introduce the auxiliary variables

$$y_i^{(k),N} = \sum_{j=i}^{B^{(k)}} x_j^{(k),N}, \qquad z_i^N = \sum_{k=1}^{K} y_i^{(k),N},$$

and then inclusion-exclusion shows

$$f_i^{(k),N}(x^N) = \frac{x_i^{(k),N}}{\sum_{k=1}^K x_i^{(k),N}} \times \left[z_i^N \left(z_i^N - \frac{1}{N} \right) \dots \left(z_i^N - \frac{d-1}{N} \right) - z_{i+1}^N \left(z_{i+1}^N - \frac{1}{N} \right) \dots \left(z_{i+1}^N - \frac{d-1}{N} \right) \right].$$

The above version of $f_i^N(.)$ is N-dependent, but it converges to

$$f_i^{(k)}(x) = \frac{x_i^{(k)}}{\sum_{k=1}^K x_i^{(k)}} ((z_i)^d - (z_{i+1})^d).$$

Due to the dependency on N, we refer to [2], where this type of dependence on N is allowed. Also, both $f_i^{(k),N}$ and $f_i^{(k)}$ are continuous. Overall, the conclusions of Theorems 1 and 2 apply.

The mean field balance equations are

$$\frac{\lambda \nu_i^{(k)}}{\sum_{k=1}^K \nu_i^{(k)}} \left(\left(\sum_{k=1}^{K} \sum_{j=i}^{B^{(k)}} \nu_j^{(k)} \right)^d - \left(\sum_{k=1}^{K} \sum_{j=i+1}^{B^{(k)}} \nu_j^{(k)} \right)^d \right) = \mu_i^{(k)} \nu_i^{(k)}. \qquad (24)$$

Since the rates $f_i^{(k)}$ are continuous, (10) and (11) can be used to compute the mean system time H.

3.5 Join-Below-Threshold

Join-Below-Threshold (JBT) sets a threshold M_k which may depend on the server type k; servers of type k with queue length $< M_k$ are considered available and servers of type k with queue length $\geq M_k$ are full. Tasks will be dispatched to a server randomly from among all available servers. If there are no available servers, tasks will be dispatched at random from among all servers.

JBT is commonly used in accordance with limited processor sharing (LPS) for servers which can serve multiple tasks simultaneously in an efficient manner. This is reflected in an increasing service rate curve $\mu_i^{(k)}$. If $\mu_i^{(k)}$ would start to decrease for large i, then this is countered by setting a maximum for the number of tasks served simultaneously in a single server, while further tasks

wait in queue (so we set the MPL). Overall, this setup ensures the service rate curve $\mu_i^{(k)}$ is increasing up to M_k and constant for $M_k \le i \le B^{(k)}$.
If we set the threshold to 1, we get the JIQ principle, and if we set it to $B^{(k)}$, we get Random assignment.

We introduce the auxiliary variable

$$y = \sum_{k=1}^{K} \sum_{j=0}^{M_k - 1} x_j^{(k)},$$

which is the ratio of available servers. For JBT,

$$f_i^{(k)}(x) = \begin{cases} 0 & \text{if } y > 0,\ i \ge M, \\ x_i^{(k)}/y & \text{if } y > 0,\ i < M_k, \\ x_i^{(k)} & \text{if } y = 0. \end{cases}$$

The mean field balance equations are

$$\mu_i^{(k)} \nu_i^{(k)} = \frac{\lambda \nu_{i-1}^{(k)}}{y}, \qquad i \in \{1, \dots, M_k - 1\}, \qquad k \in \{1, \dots, K\},$$

with $\nu_i^{(k)} = 0$ for $i > M_k$.

For a full, detailed mean field analysis of JBT, we refer to [6]; it turns out that if the stability condition (3) holds, then ν is unique, the $f_i^{(k)}$ are continuous at ν, and an efficient numerical method to compute ν is also provided in [6]. As a side note, we mention that [6] also shows examples where (3) does not hold, and there are multiple attractors in the mean-field system corresponding to quasi-stationary states of a system with a finite N, and mean field convergence fails completely.

Once ν is computed, (10) and (11) can be used to compute the mean system time H.

4 Numerical experiments

We ran several simulations to give an intuition of the behaviour of the server clusters. We also solved the transient system of equations (6) numerically. All computations were coded in Python.

In Section 4.1 several plots of these runs are included to display transient mean field convergence as N is increased. Also, as t is increased, each system will converge to its stationary state.

In Section 4.2, we focus on a few examples where certain connections between the various load balancing can be highlighted visually.

In Section 4.3, we compare the mean service times in both simulations and the mean-field settings.

Figures	λ	μ_1	μ_2	μ_3	μ_4	μ_5	$\mu_6, \ldots, \mu_{20}$
$2, 3, 8$	0.95	1	1.1	1.2	1.3	1.4	1.5
$4, 5, 6, 7$	1.25	1	1.1	1.2	1.3	1.4	1.5

Table 1: Parameter setup for the transient plots

4.1 Transient mean field diagrams

In this section, we plot the solutions of the mean field equations as well as the corresponding $x_i^{(k),N}$ curves for $N = 1000$ and $N = 10000$, resulting from simulations. We will focus on homogeneous clusters with $K = 1$ (also dropping (k) from the notation). $B = B^{(k)}$, the maximal queue length will be set to 20. The rest of the parameter setup is shown in Table 1. All parameter setups adhere to the monotonicity assumption (2) and also the stability condition (3) (in fact, the system load can be computed as λ/μ_B in a homogeneous cluster).

In Figures 1–8, the solutions of the mean field differential equations are less saturated smooth lines, while the curves corresponding to the simulation results are more saturated and have some natural fluctuations. The different color lines correspond to the ratio of queues of various length. There are two figures for every setup of parameters, one for $N = 1000$ and one for $N = 10000$. We will use these to compare the simulation with the solutions of the mean field equations.

In Figure 2 we show a system using Random assignment. We can see that the quite long queues even with the λ being quite small compared to the μ_i values. Overall, this principle is rather inefficient. Later we will see how more efficient principles handle these conditions.

We can also see the fluctuations of the simulations decrease as N is increased. Actually, as mentioned after Theorem 1, the fluctuations are guaranteed to be of order $\frac{1}{\sqrt{N}}$ for x^N (or, equivalently, order $\sqrt{N}$ for X^N). However, the constant factor can be different for the various load balancing principles. For Random assignment, the fluctuations are relatively mild.

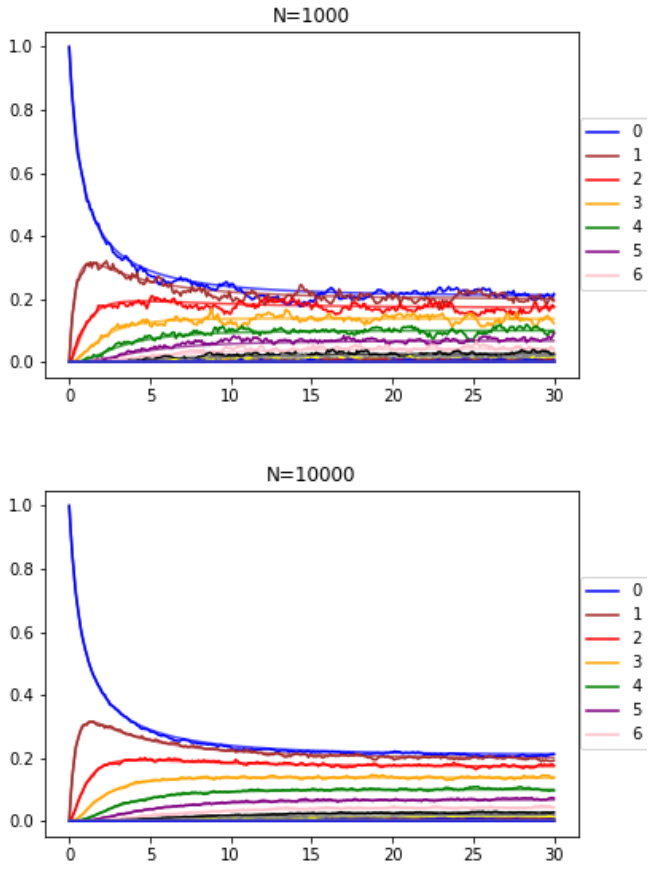

Figure 2: Random assignment

Figure 3 shows a system using Join-Idle-Queue. JIQ is more effective than Random assignment, as most of the queues in the simulation and all of the queues in the mean field limit have a length of 0 or 1. Due to $\lambda < \mu_1$, the transient solutions never reach the discontinouity point of the $f_i^{(k)}$'s (see also Section 3.2), so the solutions of the mean field equations are smooth. We also note that in the $\lambda < \mu_1$ case, JSQ is effectively identical to JIQ as the shortest queue in such systems is almost always an idle one.

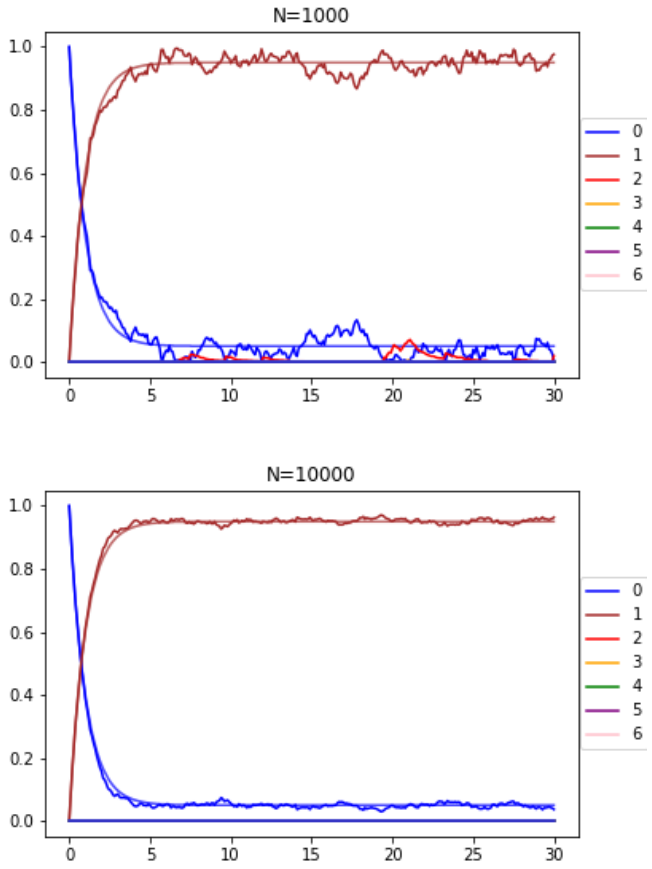

Figure 3: JIQ with $\lambda < \mu_1$

In Figure 4 we show a system also using JIQ, but with $\lambda > \mu_1$, so the system does reach the discontinuity point of the $f_i^{(k)}$'s, where the proportion of length 1 queues reaches 1. After this point, the behaviour is similar to the system with Random assignment.

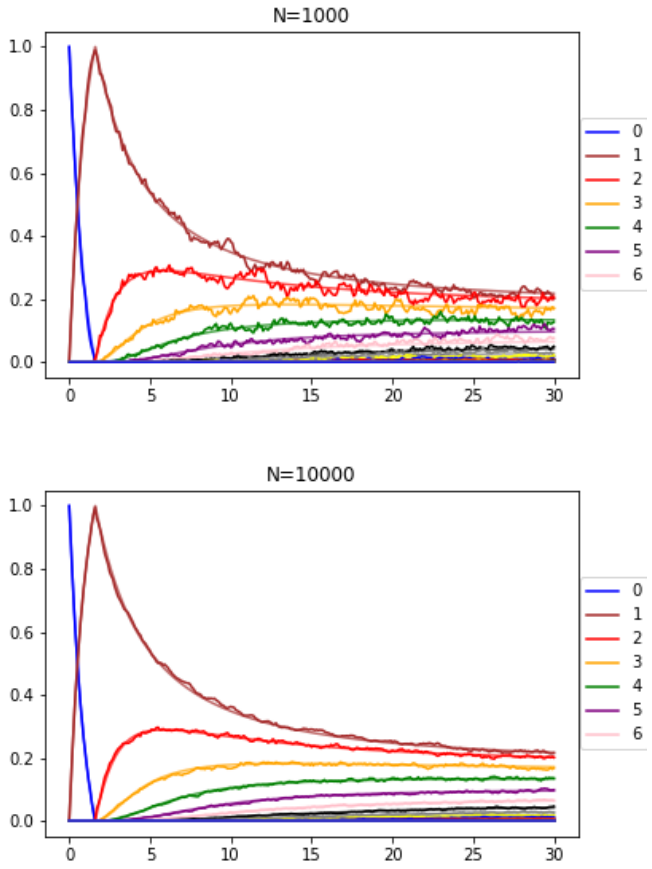

Figure 4: JIQ with $\lambda > \mu_1$

In Figure 5 we show a system using Join-Shortest-Queue. As we can see, this system stabilises with queues of length 3 and 4 due to $\mu_3 < \lambda < \mu_4$, and it has 3 discontinuity points before that. Actually, due to $\lambda = (\mu_3 + \mu_4)/2$, the mean field stationary distribution is $\nu_3 = \nu_4 = 0.5$. We also note that the fluctuations are considerably larger then for either Random or JIQ. An intuitive explanation is that the higher level of control provided by JSQ will generally focus any fluctuations in either the arrival or service on a single queue length: if the arrivals outweigh the service for a short period of time, the surplus arrivals will all go to servers of length i_0-1, and if the service outweighs the arrivals, that will affect queues of length i_0. Overall, the strict control introduces a positive correlation between the length of the queues, resulting in larger fluctuations (which are, once again, of order $1/\sqrt{N}$, but with a higher constant factor).

Principles with less strict control generally distribute this fluctuation among several different queue lengths, resulting in smaller fluctuations.

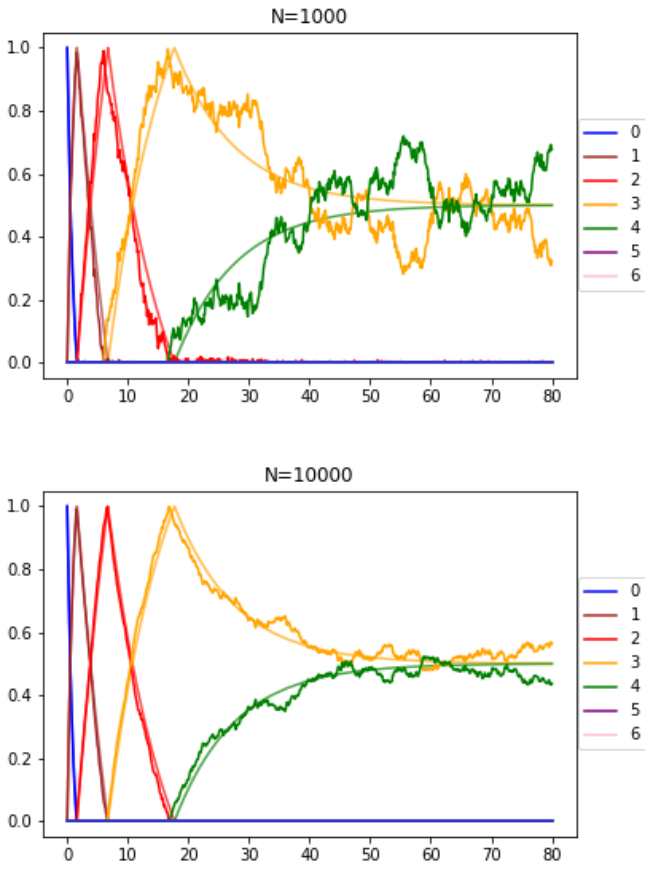

Figure 5: JSQ with $\mu_3 < \lambda < \mu_4$

In Figure 6 we show a system using Join-Shortest-Queue(2). This principle has continuous $f_i^{(k)}$ functions, just like Random assignment, so the mean field transient limit is smooth. We note that already for $d = 2$, the result is markedly different from Random assignment. This is a known phenomenon, referred to as power-of-2 [11].

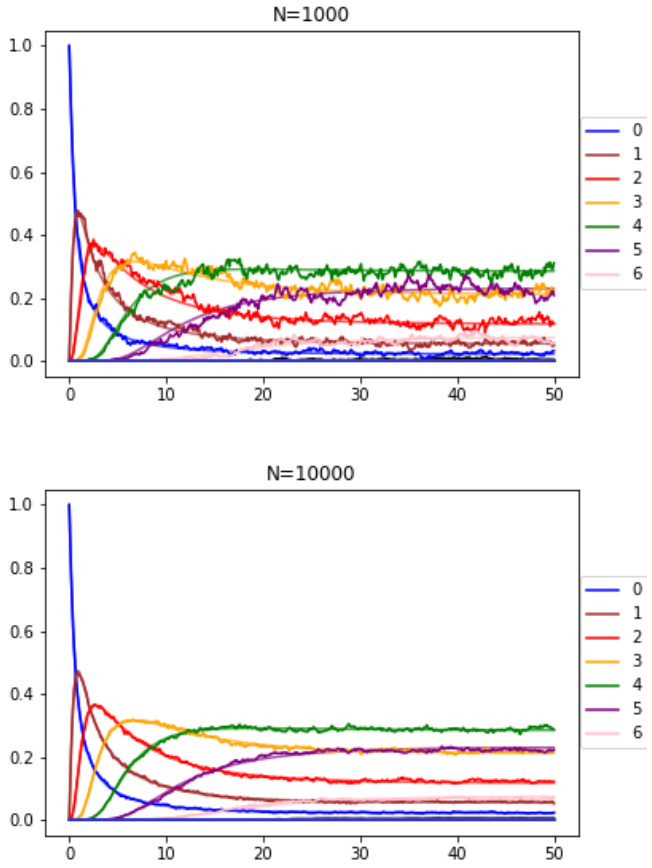

Figure 6: JSQ(2)

In Figure 7 we show a system using Join-Shortest-Queue(5). As we can see this bigger d leads to a more efficient server cluster. It is also much further from a random assignment, and closer to JSQ.

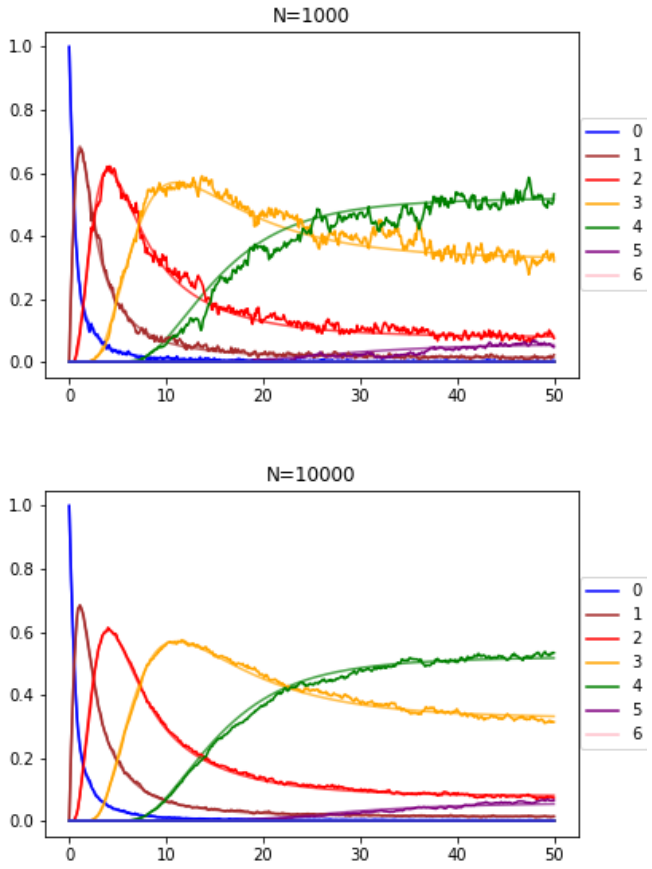

Figure 7: JSQ(5)

In Figure 8 we show a system using Join-Below-Threshold with a threshold of 3. In this setup the cluster reaches stability before filling up to its threshold. This is the intended usage of this principle, and as we can see, it is a quite effective principle. With this parameter setup, the mean field system reaches its attractor before the discontinuity point, so the functions remain continuous.

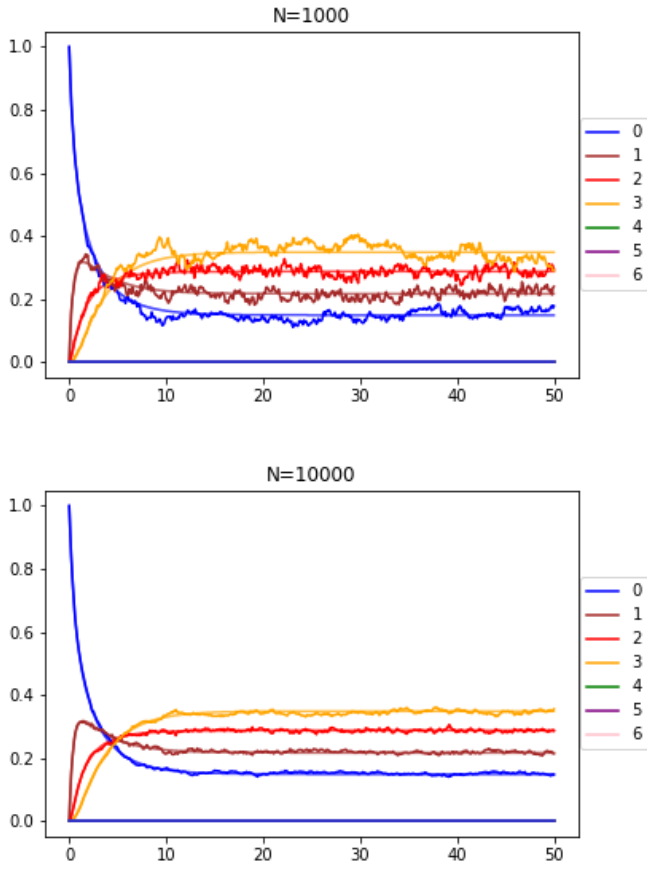

Figure 8: JBT under threshold

4.2 Other interesting plots

Here we will show several additional diagrams, not crucial to the paper's main topic. Figure 9 displays a system with Join-Below-Threshold load balancing policy, but with the demands overflowing the threshold. This, as we discussed earlier is no the intended way of utilising this principle, thus it is not very practical. The μ-curve is the usual one we used for the previous diagrams, $\lambda = 1.25$ and the threshold is set to 3. We also note that this system would be in the continuous region with the threshold set to 4, but the threshold can be set all the way up to 6, to the maximum point of the μ curve.

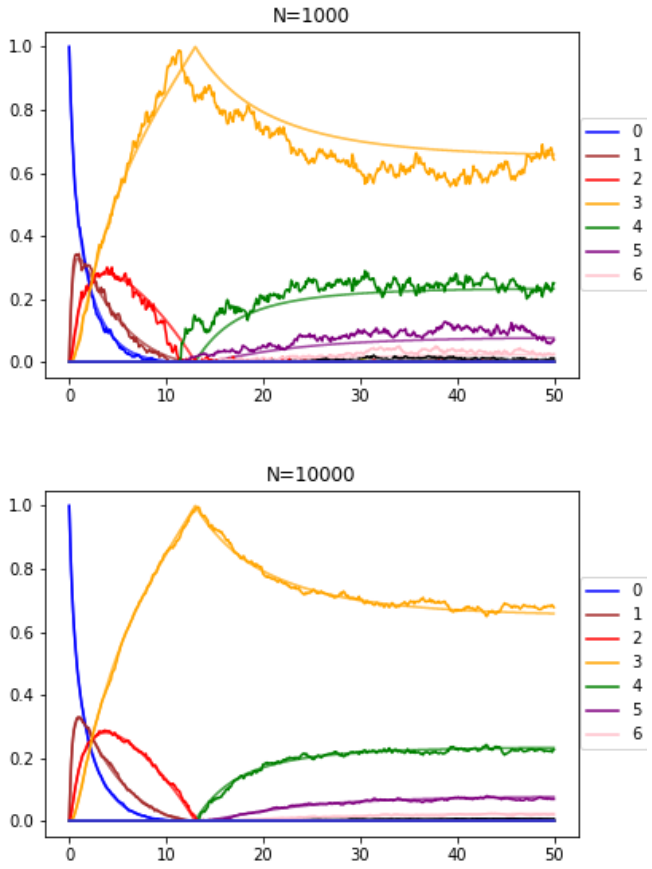

Figure 9: JBT above threshold

Figure 10 shows an interesting visualisation of JSQ(d)'s "convergence" to JSQ as $d \to \infty$. As we can see, in practice JSQ(d) is quite close to JSQ already for moderately large values of d. We also note that the mean field transient solutions are smooth for JSQ(d), but not for JSQ. Figure 10 only displays the solutions of the mean field equations, no simulations.

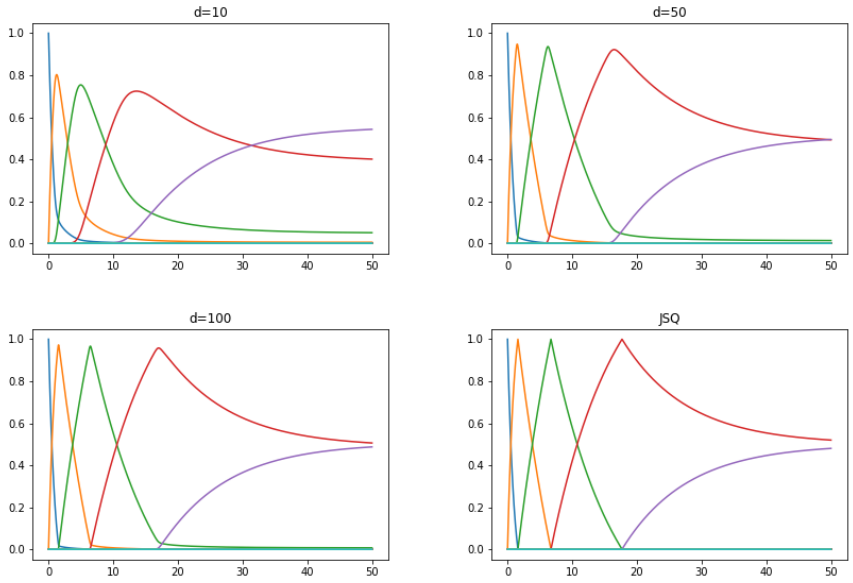

Figure 10: JSQ(d)'s 'convergence'

4.3 Mean system times

Table 2 lists the mean system times from both simulations, and calculated from the mean field limit using equations (10) and (11) (or in the discontinuous cases, their corresponding versions listed in Section 3). For this table, we use the same μ-curve from Table 1. JBT's threshold is set to 3.

λ	Load balancing	$N = 100$	$N = 1000$	$N = 10000$	$N = \infty$
1.05	Random	3.2149	3.1925	3.1764	3.1797
	JIQ	1.5526	1.446	1.4301	1.4293
	JSQ	1.4769	1.4401	1.4333	1.4286
	JSQ(2)	2.2012	2.1784	2.184	2.1835
	JSQ(5)	1.8492	1.8187	1.8076	1.8075
	JBT	2.0769	2.0755	2.0787	2.076
1.25	Random	4.421	4.6852	4.6859	4.6552
	JIQ	3.1856	3.2269	3.2374	3.2452
	JSQ	2.7051	2.8124	2.7888	2.8
	JSQ(2)	2.9876	2.9741	2.9607	2.9620
	JSQ(5)	2.897	2.8328	2.8165	2.8167

Table 2: Mean system time in the stationary mean-field limit

JBT method with $\lambda = 1.25$ was left out the table, as that system is over the threshold.

As expected, JSQ is the most effective principle in both cases. We can also see that JIQ is very effective with underloaded systems, almost on par with JSQ. However, with a higher load, JIQ falls behind.

We can also observe that relative to the others, JSQ(d) becomes more effective as the load increases, getting almost on par with JSQ. As expected, JSQ(d) is more effective with a higher d. It is also interesting to see that JSQ(2), which appears to be similar to Random assignment at first sight, is actually significantly more effective.

As long as N is finite, there are fluctuations which do not vanish even as time increases and the systems converge to their stationary limit. As expected, fluctuations are bigger for smaller values of N. For smaller values of N, the mean system time is generally above the mean field mean system time; an intuitive explanation for this is that the limited number of servers offers less 'room' to balance out short periods of overflow (coming from the natural fluctuations of arrivals and service), causing the system to operate with longer queues for said short periods.

5 Conclusion and outlook

In this paper we examined the mean field transient and stationary convergence of systems with 5 different load-balancing principles. We can say that without taking the overhead communication cost into consideration, Join-Shortest-Queue is the most efficient. However the other methods, which needs much less communication, can also be quite effective based on the circumstances. The simulations offered an intuition that the convergences occur even with discontinuous $f_i^{(k)}$ functions, and that Theorems 1 and 2 apply. We have also examined the mean system time, which also served as a good performance measure of the various

load balance principles.

Of course there is a lot of room for further work in this topic. For example, no mathematically rigorous proofs were provided for Theorems 1 and 2 for some of the discussed systems.

In addition to the mean system time, the entire service time distribution could also be calculated with the help of the Laplace transform, adapting (10) and (11) for the Laplace transforms of the system times.

One could also consider other load balancing principles: if some job size information is available about the demands, that information can be used to estimate the load of each queue more precisely.

We could also consider adding a geometrical dimension to the server cluster, with load balancing principle taking into account the distance of the arriving demand to the queues (similar to a supermarket, where customers are more likely to choose a queue physically closer to their arrival point).

We could also make the model more realistic, even if more complicated, by considering the dispatcher's communication overhead cost, or allowing different types for the demands which can be served more efficiently by certain server types.

All in all, this is a vast topic that has a lot of potential for further development.

References

[1] CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=672872

[2] M. Benaïm and J.-Y. Le Boudec. A class of mean field interaction models for computer and communication systems. Performance Evaluation, 65(11):823-838, 2008.

[3] A. K. Erlang. Sandsynlighedsregning og Telefonsamtaler, Nyt tidsskrift for matematik, 20:33-39, 1909.

[4] S. N. Ethier and T. G. Kurtz. Markov Processes: Characterization and Convergence. Wiley, 2005.

[5] N. Gast and B. Gaujal. Markov chains with discontinuous drifts have differential inclusion limits. Performance Evaluation, 69(12):623-642, 2012.

[6] I. Antal Horváth, Z. Scully, and B. Van Houdt. Mean field analysis of join-below-threshold load balancing for resource sharing servers. *SIGMETRICS Perform. Eval. Rev.*, 48(1):41–42, 2020.

[7] T. Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. Journal of Applied Probability, 7:49-58, 1970.

[8] T. G. Kurtz. Strong approximation theorems for density dependent Markov chains. Stochastic Processes and their Applications, 6(3):223-240, 1978.

[9] H.-C. Lin and C. S. Raghavendra. An analysis of the join the shortest queue (JSQ) policy. Proceedings of the 12th International Conference on Distributed Computing Systems, 362-366, 1992.

[10] Yi Lu et al. Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services. Performance Evaluation, 68(11):1056-1071, 2011.

[11] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, P. A. Whiting. Asymptotic Optimality of Power-of-d Load Balancing in Large-Scale Systems. Mathematics of Operations ResearchVol. 45(4):1535-1571, 2020.

[12] M. Mitzenmacher. Analyzing distributed Join-Idle-Queue: A fluid limit approach. In 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 312-318, Sept 2016.

[13] G. Roth, W.H. Sandholm. Stochastic approximations with constant step size and differential inclusions - SIAM Journal on Control and Optimization, 2013 - SIAM