Queueing theory methods in applications for analysis and improvement of data pipelines

João Mota

30 augusti 2023

1 Introduction

In the contemporary landscape of data-driven decision-making, organisations worldwide are grappling with the ever-increasing volume, variety, and velocity of data. Data pipelines have emerged as indispensable components in data engineering, facilitating the seamless flow of information and empowering businesses to extract valuable insights from vast data sets. Amidst the complexities of data pipeline management, data queues play a pivotal role in ensuring the efficient, reliable, and scalable movement of data from producers to consumers.

As data pipeline architectures become more sophisticated, the strategic utilisation of data queues gains paramount importance in optimising data flow and preserving data integrity. However, the dynamic nature of data demands, coupled with the diversity of data queue technologies available, poses significant challenges for organisations seeking to identify the most effective data queue solutions tailored to their unique requirements.

And thus arose the need to analyse data pipelines and similar data systems, with multiple predominant options such as queueing theory, deep learning neural networks and machine learning models.

2 Objectives

The main aim of this dissertation fall primarily on developing a queueing theory model to analyse and optimise a data pipeline created for this case study, using queueing theory parameter estimation, performance metrics and sensitivity analysis. Through these methods and their metrics, it is be possible to derive insight on a data queue and better understand its data flow dynamics, detect any possible bottlenecks that are reducing performance and ultimately understand the required resource allocation to improve the data pipeline efficiency and reliability the most efficiently manner possible.

3 Background

Queueing theory, the mathematical study of queues and waiting lines first started at the beginning of the 20th century by Agner Krarup Erlang, a Danish engineer and mathematician, formulated the mathematical model of the number of telephone call arriving a calling station using a Poisson process in 1909 solving what now is considered a M/D/1 queue in Kendall's notation and later solving the multi-queue M/D/k for an arbitrary k number of server in 1920 [Kin09a].

Queueing theory was further expanded in 1930 with the solving of the M/G/1 queue by Aleksandr Khinchin and now known as the Pollaczek–Khinchine formula and in 1953 by David George Kendall solved the GI/M/k queue and introduced the now commonly used queueing notation, known as Kendall's notation [Kin09b].

During the second half of the 20th century interest on the field was renewed from the potential applications outside the field of telecommunications in 1950s, the interest to analyse computer systems and components of these systems, in the 1960s. In the mid of the 1970s queueing theory had become a well established tool for performance analysis of computer systems, which continues to grow with the discovery of the Jackson network [ENKK^{+}10].

Nowadays, queueing theory has a wide range of applications in multiple field, such as telecommunications, traffic and transportation engineering, manufacturing, computer systems, networks, healthcare system and financial services [New82] [HB14a].

With the steady increase of datafication in the 21th century, it has been estimated that data has doubled every 1,5 years from 2008 to 2018 [OLBO15], increases the need for structures that are capable of swiftly and efficiently, pull, save and gather data for analysis and derive insights from such newly sourced data.

Therefore, multiple tools were developed to deal with these new requirements such as data pipelines. Data pipelines provide the foundation for a range of data projects which can range from exploratory data analyses, data visualisations and machine learning tasks. [IBM] However, with the total volume of data predicted grow rates, ensuring the continuous and error-free is a significant challenge.

As data pipelines performance depends on multiple variable factors, queueing theory has emerged as the mathematical framework to analyse and address these new challenges. Queueing theory, is the mathematical study of queues, for systems with a steady inflow of entries (customers) and a limited number of servers where the analyst wants to know if the current system is capable of serving all the inflow demands. The aim is to calculate the multiple queue performance metrics [Tho12]. In this research the most relevant performance metrics are Response time (T) also commonly known as Sojourn time, average number of customers in the system (E[N]), traffic intensity (ρ) which is equivalent to device utilisation in single server queues and device throughput (Xi) [HB14b]. Response time is calculated as T = $t_{departure} - t_{arrival}$, containing both service time and waiting time and gives us insights on the completion of the cycles of the system. The average number of customers in the system is calculated using Little's Law, a fundamental law of operations which are explained later in analysis, and together with the traffic intensity gives us insights on the systems capability of future incoming customers without resorting to additional queueing. Traffic intensity is the estimated parameter using the chosen estimation paradigm and explains how busy the system is and when nearing the critical value of $\rho = 1$, what is the capacity of the system. Lastly, device throughput is the rate of task completion and reflects the system ability to process any incoming customers.

4 Model Key assumptions and limitations

For the analysis of our case study, I have selected a M/D/1 queueing model, whose assumptions best fit the our case study. Starting from the M"in the first slot stands for memoryless" [HB14a], in this model interval of arrival of new entries into the queue (customers) follow a Poisson distribution , the D"in the second slot stands for deterministic, where the server processes each customer taking a fixed amount of time, idling when there are no queueing customers [ANT21a], the "1"in the third slot represents the number of servers, which is this case is a single server and the forth slot is empty to represent that this queue has an infinite capacity and follows a First-Come-First-Served (FCFS) policy [HB14a].

The limitations of the model are not having any wait room capacity and thus assuming infinite capacity, not taking into account customer balking and renegading behaviour, homogeneous service time, lack of customer priority as it follows the FCFS policy. While these limitations do not directly affect this case study analysis, it is important to acknowledge them as they are vital for future research and model expansion on a more realistic study case.

The model lack of a finite and established queue makes it impossible to take into account balking and renegading. Balking and renegading is the behaviour where the customer leaves the queue after experiencing what they consider excessive waiting times. This is usually taken into account by setting a deterministic balking probability and threshold reneging structure that would discard these customers [WZ18].

The homogeneity of service time in the model assumption, although are convenient for this research may not always hold true in all states that the data system is subjected to, since a system that is periodically overloaded and is not capable of always maintain a deterministic service time.

Lastly, this queue does not consider any priority scheduling as the queue has a FCFS policy, but in reality there can be multiple types of priority classes, such as size based policies or high priority service customers which can either increase or reduce the performance of the data pipeline [HB14c].

5 Literature Review

5.1 Previous Research

In this section I provide an overview of the current key studies and methodologies that have contributed to the study of queueing theory and M/D/1 models, highlighting the main contributions to the field defined in our scope.

TODO Starting from the Performance modelling and design of computer systemsbook by Mor Harchol-Balter, who has started to offer more practical insights and examples on how to apply queueing theory on multiple simple models such as M/M/1, M/D/1 and M/M/1/N, which have finite queue capacity compared to the M/M/1 model. In his book he delves in the importance of performance analysis, its predictive power. It further explains the relevance of performance metrics to derive insights from data queue, their estimation through operational laws such as Little's Law. Finally it gives insights on the usage of the Åhat-ifänalysis to better understand data systems and possible modifications to such systems. [HB14a].

Focusing solely on M/D/1 data queues, the first notorious piece of research is the article ML and UMVU estimation in the M/D/1 queueing system", whose main research focus is deriving the Maximum Likelihood (ML) and Uniformly Minimum Variance Unbiased (UMVU) estimation for M/D/1 queueing system and the stable M/D/1 queueing system, traffic intensity estimation, performance measures such as the expected number of customers in the queue (Lq) comparatively with number of customers in the data system (L), estimation of the correlation functions, transaction probabilities and finishing by comparing these new methods [SK16a].

This previous article research has been further developed in "Classical and Bayes estimation in the M/D/1 queueing system", with new insights on the estimation such an Bayesian approach to the estimation of the previously researched parameters for the M/D/1 queue and the usage of asymmetric loss functions for estimator as the loss is not likely to remain symmetric in real world cases. [CVDY20]

The latest contribution on the M/D/1 and current state of the art literature is ”A survey of parameter and state estimation in queues” which not only builds on the previous 2 iterations of the research, of using the classical Maximum Likelihood approach and the Bayesian approach for parameter estimation. It starts by separating the type of inference and modelling in research papers into 4 categories, the "Inference Activities", such as model selection, parameter estimation and predictions, the Models, where the models are introduced and they key aspects are highlighted, the observation scheme, where the type of data is selected for observation, and the statistical methods and principles, such as the statistical approach to estimate and analyse the model, as for example, method of moments, maximum likelihood or Bayesian inference. [ANT21b]

It further elaborates on multiple paradigms for parameter estimation such as the "Classical sampling approach", where we sample for a processed a fixed or variable number of samples, the Inverse problem estimation, where we observed attributes of the system and use those observed attributed to estimate the parameters, the Inference for Non-Interacting Systems, for models where the customers do not interact in a way that conforms to the generalised M/G/∞ queue, the Inference with Discrete Sampling, for systems that are discretely sampled over time, Inference with Queueing Fundamentals, where the fundamentals of queueing theory can apply, such as Little's Law and other operational laws, Queue Inference Engine Problems, to infer the trajectory of a queue within a given cycle of transformations, the Bayesian approach, utilising well-known queueing performance analysis formulas and considering the posterior distributions as sensible priors, Online Prediction, where we observe the states up to a certain period and make future predictions based on the observed states, Implicit Models, which is a combination of data science and queueing theory where queue like models are created without the explicitly modelling every component, and Control, Design, and Uncertainty Quantification for parameter estimation where design and control decisions are made. [ANT21c]

TODO maybe talk about the trancient research.

During a similar time period, data pipeline history started with the construction of the IBM 7030 Stretch, the first general purposed pipeline machine, built in 1961 and considered the fastest computer at the time. These innovations lead to the general pipelining, the implementation technique of overlapping instructions for execution, and its development of its description, terminology and basic techniques between late 1970s and early 1980s. During this period the development of RISC (Reduced Instruction Set Computer) architecture, refined the notion of compiler scheduled pipelines, allowing multiple operations to overlap during an instruction cycle. This instruction-level parallelism (ILP) is the basis of the modern data pipeline concept. With the advancement in multiple-issue processors, floating point systems could applied the concept of software pipelines in the compiler and in assembly, a low level programming language capable of producing one multiple operations per machine instruction as it can be observed in chapters 3.12 and 4.11 in [PH96].

With the increasing data streaming of real time data in the 2010s lead to the development of stream processing frameworks such as Apache Kafka and Apache Flink which bring us the currently available technology to process software pipelines as described in [ACL18].

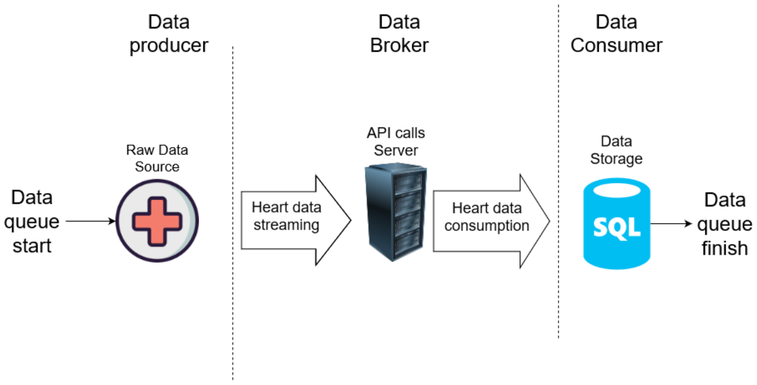
6 Methodology Overview

6.1 Research Design

My case study start by designing a prototype of a data pipeline that simulates the streaming of critical heart health metrics data that are then processed and stored in the hospital data centre to be further analysed.

The data processed in this case study is real data collected from a hospital in Boston, Massachusetts and was submitted to the Santa Fe Time Series Competition of 1991 [GAG$^+$00]. Each row has 3 parameters, the first is heart rate, the second is chest volume also referred as respiratory force and the third one is blood oxygen concentration. Each of these samples are recorded with a 0,5 second interval [RGO+93].

The data processing is divided into 3 steps, as it can be seen in figure 1, the data producer, which is responsible for feeding the heart data to the data pipeline, emulating a patient whose data is being recorded for a diagnosis and sending it to the second step, the data broker. The data broker is responsible for receiving the data from multiple patients and emulate central control system that receives all requests and forwards each request to the correct service handler, sending the heart data to the third and final step. On this final step the data of the patience, as well as the data from the metrics collected during the life cycle of this customer are stored in the database for future analysis.



Figur 1: Data queue

Each of these steps are further divided into a server located in a different geographical locations to further simulate a realistic travel time between different service handlers.

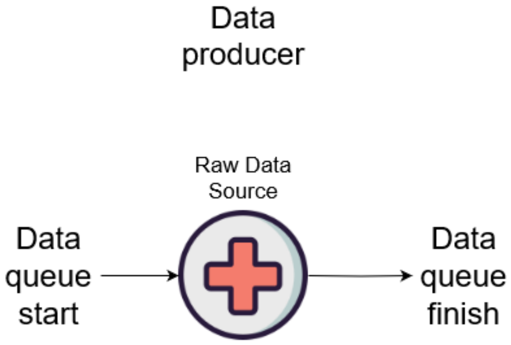
Due to the nature of the data, being processed as 3 strings of data that are identical for every single customer, in all stages of the data processing workflow, it is assumed that the service time of each individual customer is identical as the single server only serves these type of identical customers, which lead us to the main deterministic assumption of the M/D/1 model selection.

The data pipeline was simulated using Apache Kafka, a popular centralised event pipe lining platform, originally developed at Linked-in [WKS+15]. The log-centric architecture imbued into Kafka further allows the ability to more precisely record the performance metrics on the second step of the case study's prototype.

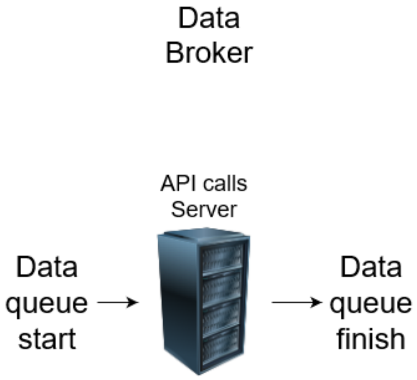
Through the processing of the data through these multiple methods, I am able to record the necessary metrics for the parameter estimation using the most estimation paradigm. Furthermore, I am able to use this initially estimated parameters to calculate additional performance metrics through operational laws, such as Little's Law.

With these performance metrics, I am also able to perform sensitivity analysis to derive insights about the impact of future changes or improvements that we might want to subject the data pipeline through, allowing us to understand which metrics are the most impactful in the data system and improve any future decision making regarding the data pipeline.

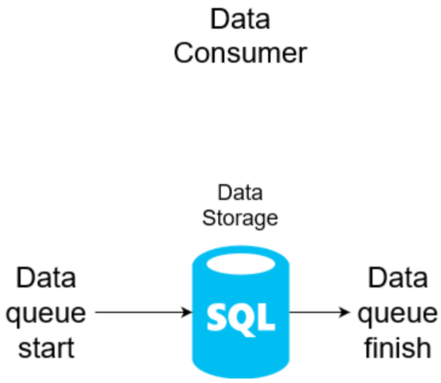
Lastly, I partitioned the data queue into 3 individual simpler queues, as it can be seen in figures 2, 3 and 4, with each queue representing one of the processing steps of the initial queue, allowing a better understanding of each of the steps performance, identify possible bottlenecks present on the queue and derive important insights for the scalability of the data pipeline. In these new queues I repeat the process of estimation of parameters through the metrics collected, utilise operational laws to calculate performance metrics relevant to the M/D/1 queue and perform sensitivity analysis.



Figur 2: Individual partitioned data producer queue.



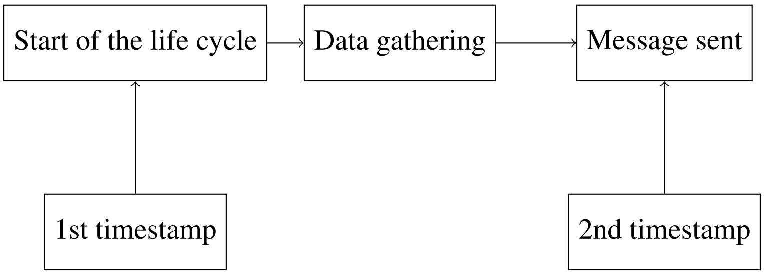
Figur 3: Individual partitioned data broker queue.



Figur 4: Individual partitioned data consumer queue.

In the case study, data collection happens at the entry and exit of each of our 3 steps, on the first step data the producer script reads one row from the heart data collected, with a frequency of 1 cycle per second (1 Hz), establishing the life cycle of the data system of new customers entering the data pipeline.

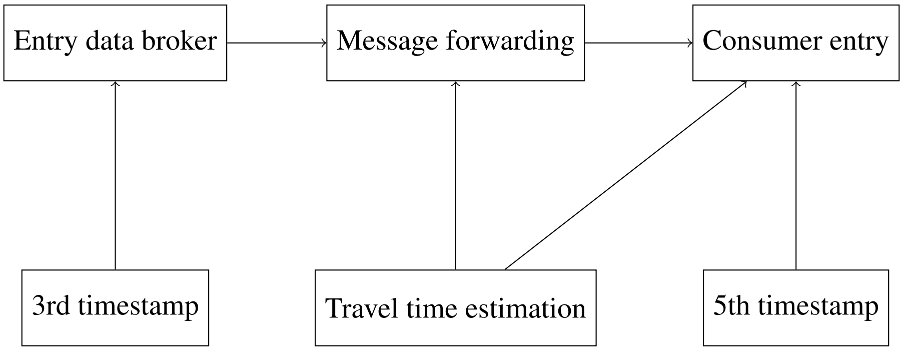
In this producer script is record two important metrics through timestamps, the first timestamp at the start of the of each new producer cycle, to register the start time of the both the entire data system life cycle and the first step of the data producer for each new customer entering the data system. The second timestamp is recorded as the producer script is ready to send the heart data of the current customer to the data broker, essentially capturing the end of the first step of data processing and providing the metrics to analyse the processing of the first step of the queue or analyse it has a smaller individual queue.



Figur 5: Illustration of the 1st step of the queue

Through this method of recording the performance metrics we can register the these timestamps with precision down to the millisecond, allowing us the as precise as possible using only software based time recording tools.

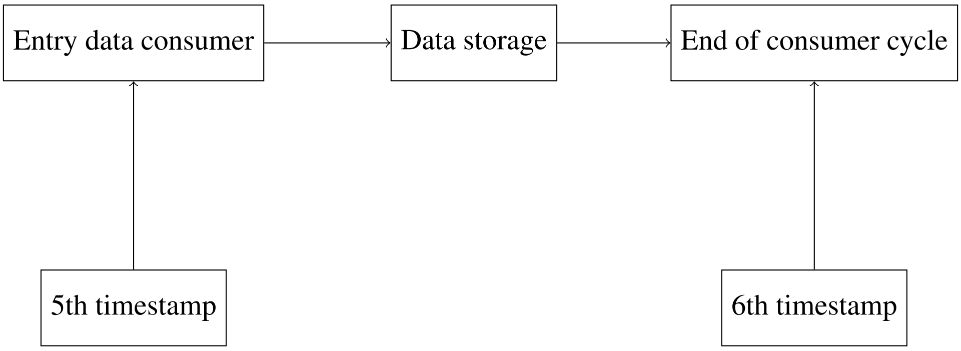
Proceeding to the second step of the pipeline, the data broker, the approach to data collection is less accurate due to the technical inability of tracking and recording in real time, the full data journey during this step without the usage of physical recording hardware. The entry of each of the customers in the data broker is recorded through the Apache Kafka service of receiving messages from the topic registered during the producer script. This timestamp allow us to calculate the travel time between the producer server and the data broker server. Since the Apache Kafka does not support a service to record the time the data is forwarded to the consumer, this timestamp is approximated by using the timestamp of entry at the 3rd step and subtracting it the time of travel between the broker server and the consumer server.



Figur 6: Illustration of the 2nd step of the queue

As you can see from the figure 6, the 4th timestamp is then estimated after deducting the travel time from the 5th timestamp, allowing for an approximation of the then processing time during the data broker.

Lastly, on the third and final step, there is a consumer script, this script subscribes to the topic created in Apache Kafka to receive all messages that are available on the broker, following a first-come-first-server type of policy when receiving each customer, recording the remaining 2 timestamps and recording both the heart data and the performance metric data do the database. This 3rd step 1st timestamp is recorded at the entry of the data to the consumer script has previously established on the last step, the script then proceeds to record all the current heart data and performance metric timestamps into the database, with the exception of the final timestamp, that is only recorded right after the register of all the available data and metrics of the customer. This last timestamp is recorded and written on the database as soon as the the other data and metadata was written, ending the life cycle of the customer.



Figur 7: Illustration of the 3rd step of the queue

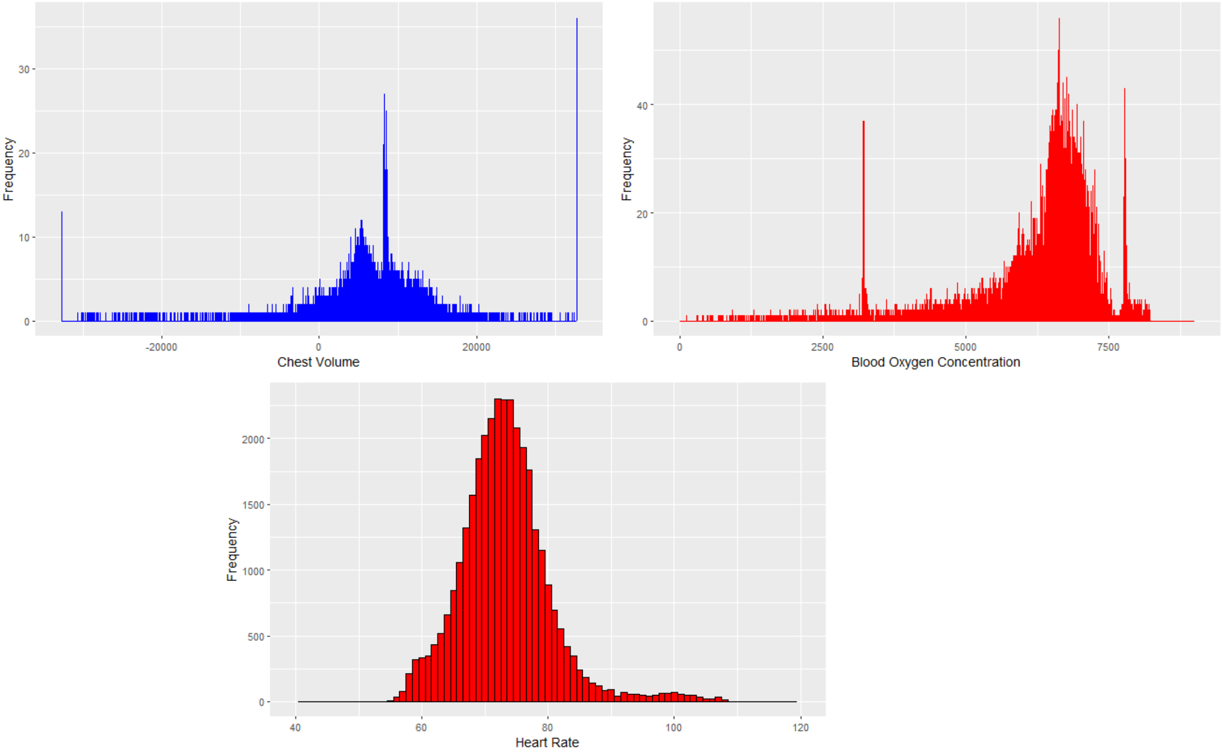
As we can see from the figure the 6th timestamp is recorded as the consumer has reached the end of its life cycle, which is then able to identify the correct timestamp to update using the customer unique id.

Now that all the metrics have been stored we can quickly convert the data from it's SQL format to a data frame to further wrangle to the same units of measurement and calculate the processing time of each step, and the travel time between the remaining 1st and 2nd steps.

6.2 Data Analysis

For the data analysis, I followed a quantitative approach to data analysis.

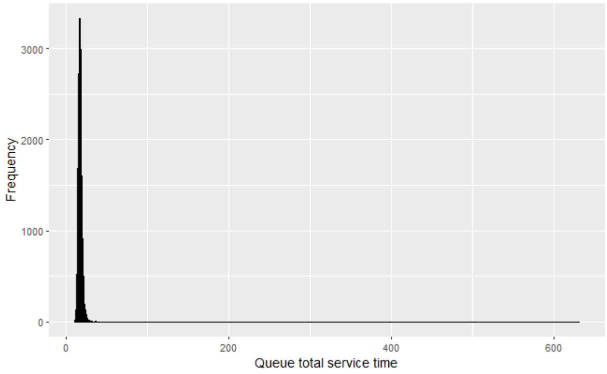
I have first started by performing a brief exploratory analysis on the data set to explore if all data is indeed similar and therefore conforms with the deterministic assumption that processing time is both predetermined and constant, eliminating the need to estimate it.



Figur 8: histogram of the 3 data set variables.

As we can see from the histograms of the chest volume, blood oxygen concentration and heart rate, there seems to be no extreme clusters of data that would discourage the deterministic assumption of all data having a similar size and therefore an identical processing and service time.

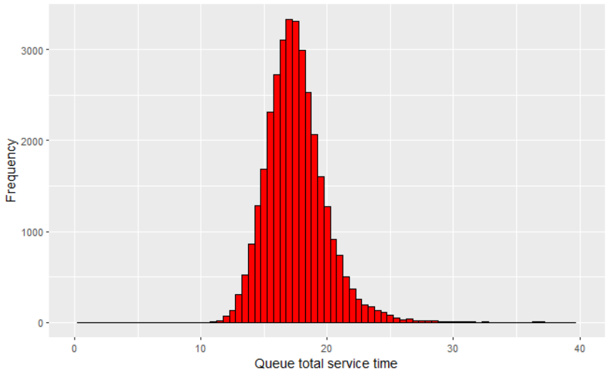
Lastly, checking the distribution of the service time of the queue to ensure the quality of the data used for parameter estimation.



Figur 9: histogram of the service time with outliers included.

As we can see from the first histogram in figure 9 there are total service times that are outrageously big compared to the average service time, with single values going over 600 seconds of total service time, this tremendously bigger service times most likely are representative of service failure and although it can be considered as outliers, it is also important to understand it's validity as representative of real world service outages.

As such I considered two cases for analysis, one that contains all the data captured and another removing the data of the top 0,5% that better represent the average non disturbed performance of the data pipeline, with a total of 170 data points out of the 34000 as it can be seen in figure 10.



Figur 10: histogram of the service time without outliers.

With this assumption we can now start by preparing to estimate the parameters for our queueing theory model.

The estimation paradigm chosen was the classical sampling approach as in this case study there are both endogenous and exogenous processes being sampled and the number of samples might be fixed to the number of rows in the data set used to generate the sampling for the case study [ANT21d].

We first start by using these metrics to calculate if this data system was able to accommodate the requests during run time by calculating the traffic intensity ρ and verify if the data queue is stochastically stable and ensure most of our assumptions hold true. The main requirement for traffic intensity, defined as:

$$\rho = \lambda \cdot m, \quad 0 < \rho < 1 \tag{1}$$

As it is shown in [ANT21e].

The currently sample size is fixed to 3400, as this is the number of entries of collected data set, this is also fixed as the number of total customers, as such we can define the number of entries as

$$X = (X_1, X_2, X_3, ..., X_{3400})$$

The primary method of estimating the traffic intensity ρ using the Classical sampling approach paradigm is the Maximum Likelihood Estimation (MLE).

The log-likelihood function is expressed as:

$$L(\rho; x) = -n\rho + \sum_{i=1}^{n} x_i \log(\rho) - \sum_{i=1}^{n} \log(x_i!) \tag{2}$$

As seen in [SK16b].

First we need estimate the $\hat{\rho}$, that we need to estimate ρ using maximum likelihood estimation.

The estimator $\hat{\rho}$ is calculated as follows:

$$\hat{\rho} = \frac{\hat{\lambda}}{\hat{\mu}} \tag{3}$$

As the study case has a large samples number of samples we are able to estimate λ and μ using the state of the art parameter estimation methods for the queueing primitives (inter-arrival and service times) [ANT21d]

Let inter-arrival time be defined as

$$A = (A_1, A_2, A_3, ..., A_{3400})$$

The parameter λ can be estimated as follows:

$$\hat{\lambda} = \frac{n_a}{\sum_{i=1}^{n_a} A_i} \tag{4}$$

Plugging in the values for the inter-arrival time to get an $\hat{\lambda}$ estimator:

$$\hat{\lambda} = \frac{3400}{\sum_{i=1}^{3400} A_i} \approx 0,9885 \tag{5}$$

Let service time be defined as

$$S = (S_1, S_2, S_3, ..., S_{3400})$$

The parameter μ can be estimated as follows:

$$\hat{\mu} = \frac{n_s}{\sum_{i=1}^{n_s} S_i} \tag{6}$$

Now we plug in the 17,3 as our predetermined measured service time to our calculate the μ and then estimate $\hat{\rho}$:

$$\hat{\mu} = \frac{1}{17,3117} \approx 0,0578 \tag{7}$$

$$\hat{\rho} = \frac{\hat{\lambda}}{\hat{\mu}} = \frac{0,9885}{0,0578} \approx 17,1021 \tag{8}$$

Now with estimator $\hat{\rho}$ and the inter arrival time A we are able to maximise the likelihood to estimate the traffic intensity of the data pipeline, using equation 2.

$$\begin{aligned} L(\rho;x) &= -n\rho + \sum_{i=1}^{n} x_i \log(\rho) - \sum_{i=1}^{n} \log(x_i!) \Leftrightarrow \\ \Leftrightarrow L(\rho;A) &= -3400 \times 17,1021 + \sum_{i=1}^{3400} A_i \log(17,1021) - \sum_{i=1}^{3400} \log(A_i!) \end{aligned} \tag{9}$$

After resolving this equation we get an estimated $\rho \approx 1.0116$ with a standard error (SD) $\approx 0,0055$

Now comparing to the filtered outlier data we resolve the equation:

$$\begin{aligned} L(\rho;x) &= -n\rho + \sum_{i=1}^{n} x_i \log(\rho) - \sum_{i=1}^{n} \log(x_i!) \Leftrightarrow \\ \Leftrightarrow L(\rho;A) &= -3380 \times 17,2464 + \sum_{i=1}^{3380} A_i \log(17,2464) - \sum_{i=1}^{3380} \log(A_i!) \end{aligned} \tag{10}$$

And when fully resolved we get a new $\rho \approx 0,9836$ and SD ≈ 0.0055, so as we can see when comparing the data without outliers with the original data, there is a tiny difference on the numeric value of traffic intensity but this change has massive implications, since by utilising the filtered data I get a $\rho < 1$ and as such the queue is in a steady state.

However analysing the estimated values of ρ for the complete data case we can see that it is not only borderline on the critical value $\rho = 1$, it is actually above the critical value, which as serious implications to both the model but also the estimation of parameters.

The 2 main implications. Firstly it is evident that if there is not enough capacity nor service speed in the current system to serve all incoming traffic at the current rate and as time increases, the queue continues to grow without a bound.

Secondly stochastic stability is a key assumption of the M/D/1 model and the estimation methods, and as such this implies that an M/D/1 model is not the most adequate when evaluating the entire data pipeline as one simple queue with the current traffic intensity [Nak05].

Furthermore, with the insight that the arrival rate remains constant during the present and future record time, with the current traffic intensity the system, the system is in a perpetual transient time and as such steady state estimations results are not accurately portray the system behaviour, rendering the current [KLD12].

For data systems that are overloaded with a traffic intensity $\rho > 1$ an M/M/1 model is more adequate due to its bigger flexibility and less strict assumptions. [ANT21f]

Due to the high complexity of performing a transient analysis I instead continued the analysis using exclusively the data with the removed outliers, since stochastic stability is required for the estimation of parameters and operational laws to hold true.

After finishing the analysis of the data pipeline in it's near maximum capacity, I performed sensitivity analysis, decreasing traffic intensity by decreasing the arrival rate of new customers to allow to derive insight of a less busy data pipeline.

Now lets star the analysis on the $\rho = 0,9835$ to identify the patterns of a system reaching it's overloaded and better understand the current system state as this is the current measured state of the system without any failure of service reducing serving efficiency.

To investigate the system I used three main performance metrics, average number of customers in the system, which are calculated through the Little's law, device utilisation which is equivalent to traffic intensity on the context of a single server queue and device throughput, calculated using the utilisation law, as they are the most appropriate to derive insight from a single server queue [HB14b].

I start by applying Little's Law to calculate the average number of customers on the overloaded system, using equation 11 and understand what is the systems maximum user capacity without resorting to ever so increasing waiting time.

Let the average number of customer be defined as:

$$\begin{aligned} L &= \lambda \times W \\ &or \\ E[N] &= \lambda \times E[T] \end{aligned} \tag{11}$$

While the first equation is the most commonly notation to refer to Little's law, some author, such as Mor Harchol-Balter use exclusively the second notation in their works [HB14b].

With a calculated E[T] $\approx 17,89$, we can plug in the Little's Law equation to calculate the average number of customers(E[N]) in the system:

$$E[N] = \lambda.E[T] = 0,98 \times 17,89 = 17,5322 \tag{12}$$

What we can derive from this average number of customers is that the data pipeline is only capable of sustaining the service of 17 customers, and depending on the possibility of higher traffic intensities, there is a need to plan for either queue capacity or waiting capacity for this bottleneck.

I can now move to calculating the throughput of the device using the utilisation law which denotes that:

$$\rho i = Xi \times E[S] \tag{13}$$

Where ρi is the device utilisation, that in this case of a single server M/D/1 queue, is synonymous with the traffic intensity, due to both measuring how busy the server or device is in relation to any incoming customer traffic. Xi is the throughput of the device and E[S] is the mean service time.

Resolving the equation to throughput and plugging in the current device utilisation and average service time:

$$\begin{aligned} \rho i = Xi \times E[S] \Leftrightarrow Xi = \frac{\rho i}{E[S]} \\ \Leftrightarrow Xi = \frac{0,98}{17,89} = 0,0548 \end{aligned} \tag{14}$$

As we can see from the equation above the throughput of the data pipeline is 0,0548 which is a very low throughput value and suggest the possibility of bottleneck present in the system and could be a highly effective optimisation opportunity. To help identify this bottleneck we partitioned the queue later in this chapter.

6.2.1 Sensitivity analysis

Now that we have done the analysis on the current data I started with the sensitivity analysis and compare each of the results with the new variable results.

I started by choosing a new value for ρ based on M/M/1 simulation exercises from Performance modelling and design of computer systems by professor Mor Harchol-Balter, of $\rho = 0,5$ since we have already analysed the state of the queue when $\rho = 1^-$ from the actual data pipeline state. This sensitivity analysis helps to more accurately analyse the data pipeline during a state of lower activity and compare its behaviour to the trends present during the system's near full capacity performance. [HB14d].

Starting with $\rho = 0,5$, we assume this values was achieved in the analysis of a less busy period of the data queue and as such, service time (μ) remains constant thus decreasing only arrival rateλ by approximately half as well, as we can see from the simple application of equation 3, we get a new $\lambda \approx 0,5$.

Now proceeding with the analysing by the two previous operational laws, Little's law and utilisation law, to calculate the average number of customers on the queue and the throughput of the queue on these new conditions.

Starting with Little's Law, I can calculate the average number of customers on the queue, using this new theoretical lambda and by calculating the average time that each customer spends in the queue.

With a calculated E[T] $\approx 17,89$, as the service time remains constant, we can plug in the Little's Law equation to calculate E[N]:

$$\begin{gathered} E[N] = \lambda \times E[T] \Leftrightarrow \\ \Leftrightarrow E[N] = 0{,}5 \times 17{,}89 = 8{,}945 \end{gathered} \tag{15}$$

As we can see, when the system on it's less busy state of $\rho = 0,5$, the system only has an average of almost 9 customers each passing second. Due to the service rate (μ) being very close to 1, a change in ρ leads to an almost directly proportional change in λ and thus changes in the arrival rate are highly impactful of the system utilisation.

Applying a second operational law, the utilisation law, I can now calculate the throughput of the system, since for a single server data system, utilisation (ρi is equal to the traffic intensity ρ.

$$\begin{gathered} \rho i = Xi.E[S] \Leftrightarrow Xi = \frac{\rho i}{E[S]} \\ \Leftrightarrow Xi = \frac{0{,}5}{17{,}89} \approx 0{,}0279 \end{gathered} \tag{16}$$

As we can see from the equation above, the throughput is almost halved in this sensitivity analysis case, showcasing just how inefficient this data pipeline can be when running at lower capacity. This low performance is a strong index of a bottleneck present on the data queue.

These findings lead us to believe, the most optimal changes would be increasing the performance of the queue on the bottleneck zone. This lead me do a partition of the data queue, allowing me to better identify the performance of each step and derive insights on the current and future computing resource allocation on this data pipeline.

6.2.2 Partitioning of the data queue

Due to the multiple step nature of our current data system queue, we are able to break down this queues into 3 different queues, considering each of the previous steps as its own individual queue. This partition of the original data queue allows to drive insight on what is major source of the bottleneck on the original data system, allowing to focus the resources on the main congestion area without needing to scale the entire system.

Starting from the new producer data queue, we can now define this new data queue as an M/D/1 due to continuing to have identical customer sizes and thus identical service time.

On this producer queue, the arrival rate is the same as the initial queue, which we had already estimated in equation 5 where we arrived at $\hat{\lambda} \approx 0,9885$

Due to the deterministic property of the service time, simply measuring is enough to get S = 0,02 instead of needing to estimating this parameter, as previously done the parent queue.

Plugging it on the new $\hat{\mu}$ equation we get:

$$\hat{\mu} = \frac{1}{S} = \frac{1}{0,02} = 50 \tag{17}$$

And now calculating the traffic intensity $\hat{\rho}$ that we require for the Maximum Likelihood estimation:

$$\hat{\rho} = \frac{\hat{\lambda}}{\hat{\mu}} = \frac{0,9885}{50} \approx 0,0198 \tag{18}$$

Now following the previously established equation 2 for the maximum likelihood estimation I estimated the traffic intensity of the producer queue

$$\begin{gathered} L(\rho;x) = -n\rho + \sum_{i=1}^{n} x_i \log(\rho) - \sum_{i=1}^{n} \log(x_i!) \Leftrightarrow \\ \Leftrightarrow L(\rho;A) = -3400 \times 0,0198 + \sum_{i=1}^{3400} A_i \log(0,0198) - \sum_{i=1}^{3400} \log(A_i!) \end{gathered} \tag{19}$$

We get an estimated $\rho \approx 0.9866$

With the $\hat{\rho} < 1$ we can determine that this system is stochastically stable, meaning that under the regular conditions observed of service time and inter arrival time, as time approaches infinite, the queue length and waiting time converge to the limiting distributions [ANT21e].

Now that I confirmed this data queue is in a steady state, I considered other relevant metrics for single server systems, such as response time, Sojourn Time (T), waiting time (Tq), number of customers on the system (N), throughput (X) and utilisation (U) [HB14b].

Applying a second operational law, the utilisation law, I can now calculate the throughput of the system, since for a single server data system, utilisation (ρi is equal to the traffic intensity ρ.

$$\rho i = Xi.E[S] \Leftrightarrow Xi = \frac{\rho i}{E[S]}$$

$$\Leftrightarrow Xi = \frac{0,5}{0.2} \approx 2,5 \tag{20}$$

We can plug in the Little's Law equation to calculate the average number of customers(E[N]) in the system:

$$E[N] = \lambda.E[T] = 0,98 \times 0,02 = 0.0196 \tag{21}$$

Now moving on the to second step of the partitioned queue, the broker queue, the data analysis was not possible because this queue fails one of the assumptions of the M/D/1 model that is the FCFS policy. This has the implications that the data that is needed to calculate lambda using equation 5 is not possible has we have multiple instances of negative Service times, which is not reliable data to do the maximum likelihood estimation since it is not possible to get the natural logarithm of a negative number.

Lastly, moving on the to third step of the partitioned queue, the consumer queue, the data analysis was again not possible because this queue continues to fails one of the assumptions of the M/D/1 model that is the FCFS policy. This again has the implications that the data that is needed to calculate lambda using equation 5 is not possible has we have multiple instances of negative Service times, which is not reliable data to do the maximum likelihood estimation since it is not possible to get the natural logarithm of a negative number. With this it will not be possibly to fully identify the bottleneck of the queue but only exclude that the producer queue is not responsible.

6.3 Results

Parameter	Study Case Data	Sensitivity Analysis Case ($\rho = 0.5$)	Producer Queue
Service Time	$S = 17.3$ seconds	$S = 17.3$ seconds	$S = 0,2$ seconds
Traffic Intensity (ρ)	$\rho = 0.9885$	$\rho \approx 0,5$	$\hat{\rho} \approx 0.9885$
Throughput (X)	$Xi = 0,0548$	$Xi = 0,0279$	$Xi = 2,5$
Average Customers in Queue	$E[N] = 17,5322$	$E[N] = 8,945$	$E[N] = 0.0196$

Tabell 1: Estimated Parameter Values for Different Cases

As we can see from the table 1 the queue in its current state is almost overloaded, borderline the critical value of $\rho = 1$ and does require an upgrade in either performance or throughput to continue service customers without risk of entering in an unsteady state. The performance of the of the data queue is also very poor with a maximum throughput of 0,0548, however from the producer queue we have established that the source of the main queue's bottleneck is not on the producer queue due to it's very low service time comparing to the parent queue and high throughput.

6.4 Future improvements

Given more time dedicated to this project, I would expand the analysis on 2 main topics, firstly increasing the complexity of the analysis by completing the transient analysis and further performing sensitivity transient analysis to understand the behaviour of the overloaded data system. Secondly I would increase the complexity and realism of the model by increasing the number of server, allowing for parallel processing and including a finite queue. Thirdly I would relax the deterministic assumptions and use a more general model

The main focus of the increase in analysis complexity would be through sensitivity analysis, increase the number of parameters used in this analysis and the choice of parameters that would allow transient analysis of the original data to be compared to other transient states, such as a more overloaded queue of $\rho = 1,5$. This would allow to derive deeper insights on the behaviour of a more dynamic scenario, improve predictions for real world occurrences and better inform future decision making and planning for optimisation, problem solving of potential system issues and computational resource management.

As some of the current M/D/1 and M/D/k model limitations are not compatible with mandatory policies of a more realistic data system, such as balking and renegading policies and customer priority service, for a more realistic analysis it is necessary to relax some of the queue initial assumption to a model closer to an M/G/M/c, where the customers distribution can follow any general distribution, with multiple parallel servers and a finite queue.

Secondly, as the current data pipeline follows a single server and is running a single one threaded script it only capable of serving one customer at the time. Increasing the number of servers increases the complexity on how busy are each server, as their utilisation does not equate to the traffic intensity anymore, as the system is now capable of parallel processing. This allows to derive insights on the scalability of the data system, predominantly on the bottleneck zone. Furthermore the limitation of a finite queue can give insight on the importance of reduced waiting times and how customers maybe be willing to leave the queue in long queues [CSS16].

7 Summary and discussion

To summarise, from the analysis proposed I have managed to fulfil the requirement of analysing the data pipeline and deriving useful insights for future improvement of the pipeline, although not to the full degree wished when proposing the study case, as it not possible to analysis all partitioned queues, due to the combination of case study having very small processing intervals that were possible to accurately record using the available data capturing methods which deal to dirty data in both the broker queue and the consumer.

The key findings were that the data pipeline is dangerously near the critical value for overloading and entering an transient state, that the data pipeline is suffering from a bottleneck, from either the data broker or the data consumer steps, successfully excluding the processor step.

The methodology starting from the choice of the classical sampling approach as the estimation paradigm, the sensitivity analysis and the partition of the queue were correct choices, but due to the captured data it was simply not possible to increase the depth of this analysis.

The study case proposed was a realistic scenario but the extra complexity of a more realistic scenario would allow to better demonstrate the utility of the type of analysis chosen on this dissertation.

For further research, this case study would benefit for transient analysis, which would derive on a state of the queue I could not analyse and would increase the possibilities of parameter testing for the sensitivity analysis, further increasing the depth of the analysis.

Referenser

[ACL18] Tyler Akidau, Slava Chernyak, and Reuven Lax. *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*. O'Reilly Media, Inc., 1st edition, 2018.

[ANT21a] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):40, 2021.

[ANT21b] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):42–55, 2021.

[ANT21c] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):55–74, 2021.

[ANT21d] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):55–57, 2021.

[ANT21e] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):45, 2021.

[ANT21f] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97(1–2):58, 2021.

[CSS16] Anna Conte, Marco Scarsini, and Oktay Sürücü. The impact of time limitation: Insights from a queueing experiment. *Judgment and Decision Making*, 11(3):260–274, 2016.

[CVDY20] P. Chandrasekhar, V. S. Vaidyanathan, T. M. Durairajan, and V. S. Yadavalli. Classical and bayes estimation in the m|d|1 queueing system. *Communications in Statistics - Theory and Methods*, 50(22):5411–5421, 2020.

[ENKK$^+$10] Attila Egri-Nagy, Emőd Kovács, Gergely Kovásznai, Gábor Kusper, and Tibor Tómács. 8 th international conference on applied informatics. page 13, 2010.

[GAG$^+$00] Ary Goldberger, Luis Amaral, Leon Glass, Jeffrey Hausdorff, Plamen Ch. Ivanov, Roger Mark, ..., and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[HB14a] Mor Harchol-Balter. *Performance modeling and design of computer systems: Queueing theory in action*, page 236. Cambridge University Press, 2014.

[HB14b] Mor Harchol-Balter. *Performance modeling and design of computer systems: Queueing theory in action*, pages 14–15,17–19. Cambridge University Press, 2014.

[HB14c] Mor Harchol-Balter. *Performance modeling and design of computer systems: Queueing theory in action*, pages 483–485,499–501. Cambridge University Press, 2014.

[HB14d] Mor Harchol-Balter. *Performance modeling and design of computer systems: Queueing theory in action*, page 246. Cambridge University Press, 2014.

[IBM] IBM. What is a data pipeline.

[Kin09a] J. F. Kingman. The first erlang century—and the next. *Queueing Systems*, 63(1–4):3–4, 2009.

[Kin09b] J. F. Kingman. The first erlang century—and the next. *Queueing Systems*, 63(1–4):4–7, 2009.

[KLD12] William H. Kaczynski, Lawrence M. Leemis, and John H. Drew. Transient queueing analysis. *INFORMS Journal on Computing*, 24(1):10–11, 2012.

[Nak05] Kenji Nakagawa. On the series expansion for the stationary probabilities of an m/d/1 queue. *Journal of the Operations Research Society of Japan*, 48(2):111, 2005.

[New82] Gordon Frank Newell. *Applications of queueing theory*, page 3–4. Chapman &amp; Hall, 1982.

[OLBO15] Peter O'Donovan, Kevin Leahy, Ken Bruton, and Dominic TJ O'Sullivan. An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. *Journal of big data*, 2(1):1–26, 2015.

[PH96] David A. Patterson and John L. Hennessy. *Computer Architecture: A quantitative approach*. Kaufmann, 1996.

[RGO$^+$93] Daniel R. Rigney, Ary L. Goldberger, William C. Ocasio, Yoshihiro Ichimaru, George B. Moody, and Roger G. Mark. Multi-channel physiological data: description and analysis. In Andreas S. Weigend and Neil A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 105–129. Addison-Wesley, 1993.

[SK16a] V. Srinivas and B. K. Kale. Ml and umvu estimation in the m/d/1 queuing system. *Communications in Statistics - Theory and Methods*, 45(19):5826–5834, 2016.

[SK16b] V. Srinivas and B. K. Kale. Ml and umvu estimation in the m/d/1 queuing system. *Communications in Statistics - Theory and Methods*, 45(19):5827–5828, 2016.

[Tho12] Nick T. Thomopoulos. *Fundamentals of queuing systems: Statistical methods for analyzing queuing models*. Springer, 2012.

[WKS$^+$15] Guozhang Wang, Joel Koshy, Sriram Subramanian, Kartik Paramasivam, Mammad Zadeh, Neha Narkhede, Jun Rao, Jay Kreps, and Joe Stein. Building a replicated logging system with apache kafka. *Proceedings of the VLDB Endowment*, 8(12):1654–1655, 2015.

[WZ18] Qiangqiang Wang and Bin Zhang. Analysis of a busy period queuing system with balking, reneging and motivating. *Applied Mathematical Modelling*, 64:480–488, 2018.