Design and Development of M/G/1 Queuing model for Real Time Applications

N. Paranjothi[1] , Vuppala Lakshmi Narayana[2] , Dr. Kanthala Sampath Kumar[3]

[1]Assistant Professor, Department of Statistics, Annamalai University, Chidambaram, India.
[2]Research Scholar, Department of Statistics, Annamalai University, Chidambaram, Department of Humanities and Sciences, Vardhaman College of Engineering, Hyderabad, Telangana, India.
[3]Assistant Professor, Department of Applied Statistics, Telangana University, Nizamabad, India.
[1]Jothi_stat@yahoo.co.in [2]vnarayana2006@gmail.com [3]ksampath1@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract: : The questions relevant to the queuing system's performance can be addressed by considering the available information of user demand and system capacity. This system can evaluate based on different parameters such as service denial, service duration, response time, etc. Based on analytic tools of Queuing Theory, the desired evaluation of performance is not able to conduct, since the models are of specific kinds for arrival and service processes. The system performance is evaluated based on the simulation. The case of a Spotify back-end server is considered in this work that utilizes in the well-known on-line media-streaming service. A set of models is provided that demonstrates the service capabilities of the band-end server and the user demand. The proposed method is measure the performance of a system based on the simulation of behaviour towards the user input demand.

Keywords: Queuing model, M/G/1 model, Spotify server, Real time applications.

I. Introduction

A Spotify band-end site is considered that is responsible for music delivery for the Spotify customers [1-2]. In Fig.1, the Spotify back-end site's structure is illustrated. A set of storage servers includes in the site on which the music files or songs are stored. C is indicated the number of available storage servers. N is denoted the total number of songs in the system [3]. N indicates the set of songs. In a single storage server, Ni is stored which is assumed for each song. $C_i \in \{1, ..., C\}$, i = 1, 2, ..., N denotes the allocation of song N_i. The information about this system is how the songs are allocating on the various storage servers is contained in the song allocation [4] vector, C = {Ci}, where i = 1,….N.

1.2 Client Requests & Server Response

The client requests to the back-end site i.e. requests for downloading the music files, produce by the Spotify users that relevant to a large population. The Spotify manager collects a client request for a music file which forwards to a storage server immediately on which the respective music file is stored [5-6]. The arrival process of the client requests to the Spotify Manager is modelled using a Poisson time-invariant process with intensity λ requests per second.

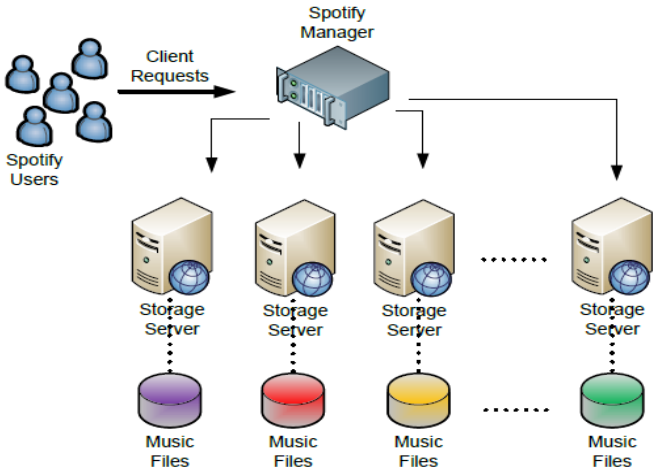

Figure 1: Architecture of the Spotify back-end server

The storage server buffer is queued-up by the requests which are served with the policy of first-come-first-served [7]. At most client requests of B, the buffers of the storage servers can hold. The client requests can't be served by a server in parallel. By relying on whether the storing of music file in the hard disk of the server or in its cache

memory, either disk or cache memory access involves in the service of a request [8]. In case of storing a file in the cache, the service time is lower which assumes reasonably.
In the cache of a server, a song is stored [9] with a probability of pC and is stored on the disk with probability 1-pC. A fixed time of T_S^C seconds, a request with service time requires for this song if it is stored on the cache. Or else, with a mean of T_S^H seconds, an exponentially distributed amount of time requires [10].

As the total time between a request arrival at the Spotify back-end server [11-12] and the time when the request initiates to serve by the corresponding storage server, the response time of a client request T_R is defined.

1.3 Song Popularity

The definition of the popularity q_i of a music file, N_i is described as the probability of a random client request that includes this specific music file, i.e.:

$$q_i = P_r\{Request\ is\ for\ song\ N_i\} \epsilon\ (0,1), \quad i = 1, \ldots\ldots, N \qquad (1)$$

We assume that the popularities of the arriving client requests are i.i.d random variables. The popularity vector, Q contains the popularities of all songs [13-14], i.e.:

$$\varrho \triangleq \{q_1, q_2, \ldots . q_N\},$$

where it, clearly, holds:

$$\sum_{i=1}^{N} q_i = 1$$

II. Literature Review

Singh L. K et al., [15] proposed an improved scheme for autonomous performance of Gateway servers under the condition of highly dynamic traffic loads. Performance metrics such as waiting time and buffer estimation can be computed by using the performance models. The possible queue models have been implemented to determine the final value of memory size and queue length. However, the proposed system can provide better track and smooth performance control in Web Server Systems based on the analyzation of simulation results.
Kuaban et al., [16] proposed the concept of correlated reneging is considered in a finite capacity multi-server queuing model with balking in health-care industry to avoid the queues or waiting lines at the hospitals, medical laboratories, and many other healthcare facilities. The numerical examples have provided to describe the effect of correlated reneging and balking on performance measures such as the probability of patient rejection, mean waiting time of patients, and the mean number of patients waiting to be serviced. So, it will be helpful to allocate the service resources and capacity of healthcare facilities in such a way that it can reduce the waiting time of patients effectively.
Horváth et al., [17] introduced the M/G/1 resampling queue and analyses based on the non-pre-emptive LIFO policy which approximates the processor sharing queue with accurate service time when the distribution of accurate service time is exponential. By analysing the simulation results, the resampling queue overestimates the mean response time of the PS queue with accurate service time for a service time distribution with increasing hazard rate. It is contrary that means the resampling queue underestimates the PS queue mean response time to the service time distribution with decreasing hazard rate.
Wu De-An et al., [18] investigated an M/G/1 queue with exhaustive service discipline and multiple vacations that means the server works with different service times instead of stopping service completely in a vacation. For the distribution of the vacation length, the Laplace-Stieltjes transform is assumed as a rational function. By using the transient solution for the queue size, the LST of the distribution function and PGF have been derived for the system time of an arbitrary customer. An M/G/1/WV behaves like an M/G/1 queue if the vacation length is very small or very large.
Charan Singh Jeet et al., [19] presents a single server queueing model by considering the arrived units in bulk with varying arrival rates in the Poisson process. The proposed system's steady-state behaviour is investigated and different performance measures have been obtained by assuming the supplementary variable technique and probability reasoning in addition to the incorporation of maximum entropy principle. The model can be utilized to deal with the real-time applications of various industries by implementing the concepts like general service time, balking, vacation, etc.
Jafarnejad Ghomi et al., [20] studied the modelling of cloud computing based on a queuing theory have been investigated and presented the results of an SMS integrated with SLR of the existing studies of AQTMCC. Additionally, it provides the need for conducting more research on AQTMCC and its future line.
Ke Jau-Chuan et al., [21] focused on studying a general retrial queue with balking and feedback in which a policy of modified vacation operates by the server. Some specific performance measures have been derived with the implementation of the supplementary variable technique in a general retrial system. Based on the analyzations on

different performance metrics, the general decomposition law is a good fit for this model which may operate in some real-time applications like e-mail system, WWW server, etc.

III. Proposed Method

Primarily, provide input for all the data such as P_C, N, B, C, T^H_S , T^C_S, and λ. The maximum time (T_{Max}) is defined for which the system will run and the minimum time (T_{Min}) will take to reach the steady state condition. For all the 5 storage servers, the response time is set to 0. The file_popularities.txt and songs_allocation.txt files are imported in Matlab.

For the given system, the primary task was to produce the client arrival request between t=0 to $t<T_{Max}$. A file with name arrival requests is created for achieving this task i.e. empty at t=0 and filled out as time increases from 0 to T_{Max} with a factor of $t+1/\lambda$. As soon as t becomes greater than T_{Max} the loop ends. The generated number of requests contain in the file at the end of the loop and the time at which the requests are produced from t=0 to $t<T_{Max}$.

The second task is to allocating the songs for the above generated requests. The probability of requesting a song from the file of file_popularities.txt to a file of cdf.txt is derived. Based on the approach of adaptive numerical integration, these probabilities can be added. Another file named as song_alloc.txt is made that has the same length like arrival request file and it doesn't have songs. A loop from 1 to length of arrival request is running and a random number between 0 to 1 generates. Additionally, determines the first accumulative probability which is greater than the random number and allocate that song in the file of song_alloc.txt.

In this work, the third task is to send the requests of a song to the servers by considering the file of song_allocation.txt. A file named server is created that includes a length with a range of 1 to the arrival request. In the file of song_allocation.txt, the songs are stored with a server number. The requested song has included in the song_alloc.txt. The requested song is matched with the stored server and the request is sent to that server.

The song requests, arrival times, and server have already existed where the request is to be sent. Two scenarios have considered while sending the request to the server. In the first case, the song is in cache memory and secondly, the song is in disk memory. A rand function generates and compares it with P_C when the rand is less than P_C. For a fixed time of 10^{-4} seconds, the request is served. Or else, it will have an exponential service time and will go to the disk memory. The difference between the arrival time and end of service time is defined as the response time. The end of service time will be = t + service time of request (T^C_S or T^H_S).

If a queue length < B than song goes to queue else drop request, a loop is running for determining the drop request. As mentioned the work description, the variance and other parameters are computed.

3.1 Queuing model

Due to the customers may have to wait in the queue before serving them by the system, the queuing system can be defined as the waiting system in Spotify back end server is shown in figure 2.

For the system, the Kendal notation will be M/G/5/100, where 100 represents the system's queue length, 5 indicates the number of servers in the system, G refers the general system (unspecified), and M represents that the arrival process is a Poisson process.

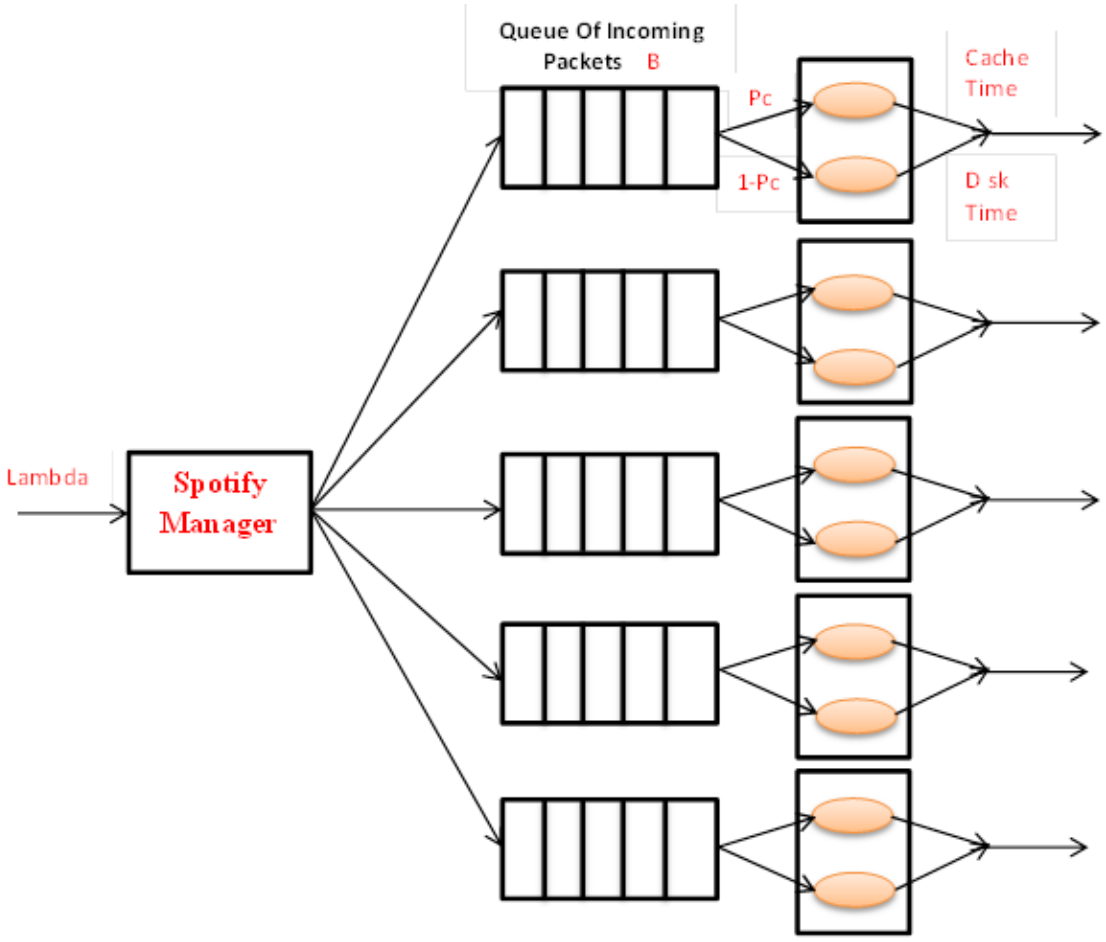

Figure 2: Queuing model in Spotify back-end system

The definition of arrival process is described as the generated number of requests by the customers. Based on the Poisson Process, the customer requests are provided. The parameter lambda (λ) whose value is to be 5000 requests per second is provided. By using the Spotify manager, the request is sent to the storage server where the song is stored.

As the client request can be served, the service time distribution is not specified with the use of two different service time intervals. In the server's cache memory, the first service time interval is a fixed time interval for the stored songs. In the hard disk, the second service time interval is an exponential time interval for the stored songs. T^C_S denotes the service cache time that has a fixed value of 10^{-4} seconds and T^H_C indicates the service disk time which includes an exponentially distributed time with 10^{-3} seconds. A song will have a probability of a service cache time which is given by P_C with a value of 0.25 and a probability of a service disk time is given by 1-P_C with a value of 0.75.

3.2 Parameter Setting

The parameter setting for the considered scenario is listed out in Table 1.

Table 1: List of parameters for the numerical solutions

Request Arrival Rate (λ)	$5000\ \text{sec}^{-1}$
Service Time (cache)(T^C_S)	$10^{-4}\ sec$
Service Time (disk) (T^H_S)	$\sim \exp(\mu)$, $1/\mu = \bar{T}^H_C = 10^{-3}$sec
Number of Servers (C)	5
Buffer Capacity (B)	100
Number of Songs (N)	10^3
Cache Availability (pC)	0.25

The popularity vector Q is contained in the file i.e. filepopularities.txt and is the popularity of each song (1000 songs). The song allocation vector C includes in the file song allocation.txt. The number of the respective server for each song i.e. 1, 2, 3, 4, or 5. Between the two files' entries, one-to-one correspondence is existed.

IV. Results and discussion

4.1 System Simulation

The above system simulates through the designing and implementing of a program. Any programming development environment is allowed to use in addition to the Matlab™. We can consider a benefit of its statistic toolbox which includes libraries for random generators and probability distributions.

It's recommended to draw a pseudo-code of the simulator before starting the programming and utilize for checking whether the simulator operates according to the requirements. In the following section, the pseudo-code delivers in addition to the answers.

For extracting the significant results statistically, the above system simulates for a period of time i.e. sufficiently long after implementing the system. For some initial warm-up period, the simulator run is allowed and the results are collected after completing this period. Accordingly, the system has been reached to the steady-state.

4.2 Performance Evaluation

4.2.1 Queuing System

The Spotify back-end server is modelled by using the queuing systems and is to be simulated. The block diagrams and the Kendall notations are provided. The distributions of service time and the arrival processes have been defined together with the parameters.

4.2.2 Response Time

The queuing systems are simulated and computed the Spotify back-end system's performance in terms of the response time. The empirical distribution (CDF) of the requests' response time is delegated to the derived server for each storage. The average response time is measured for each server.

Response time of system with P_C = 0.25

Figure 3 represents the response time at server 1. The average response time at server 1 is 0.0767.

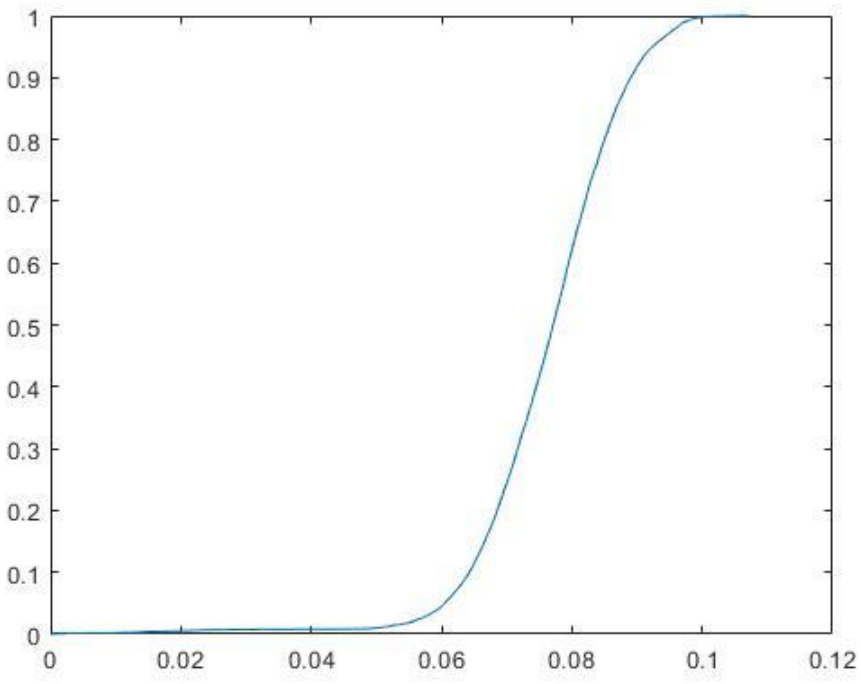

Figure 3: Response time at 1 server

Figure 4 represents the response time at server 2. The average response time at server 2 is 0.0728.

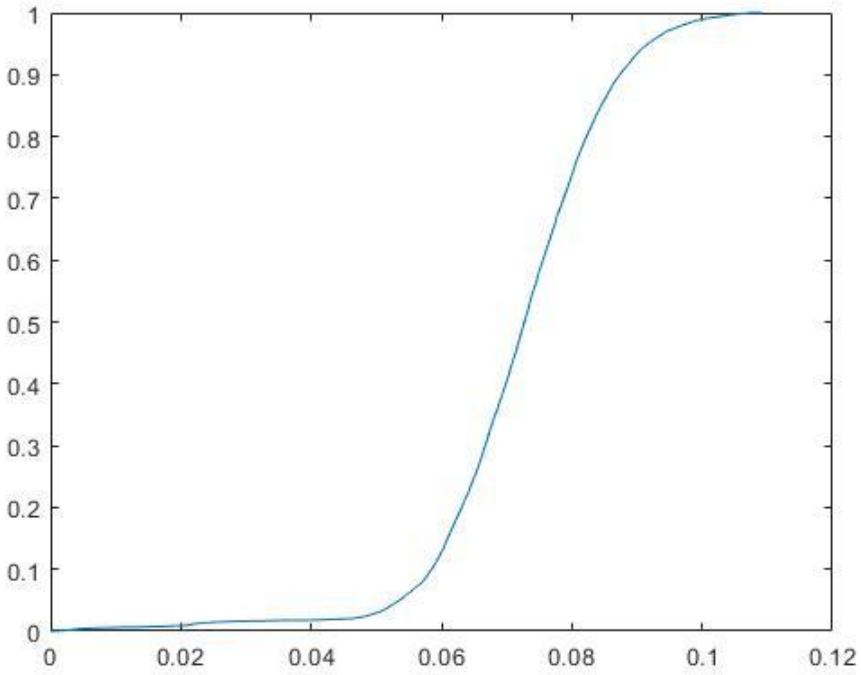

Figure 4: Response time at 2 servers

Figure 5 represents the response time at server 3. The average response time at server 3 is 0.0046.

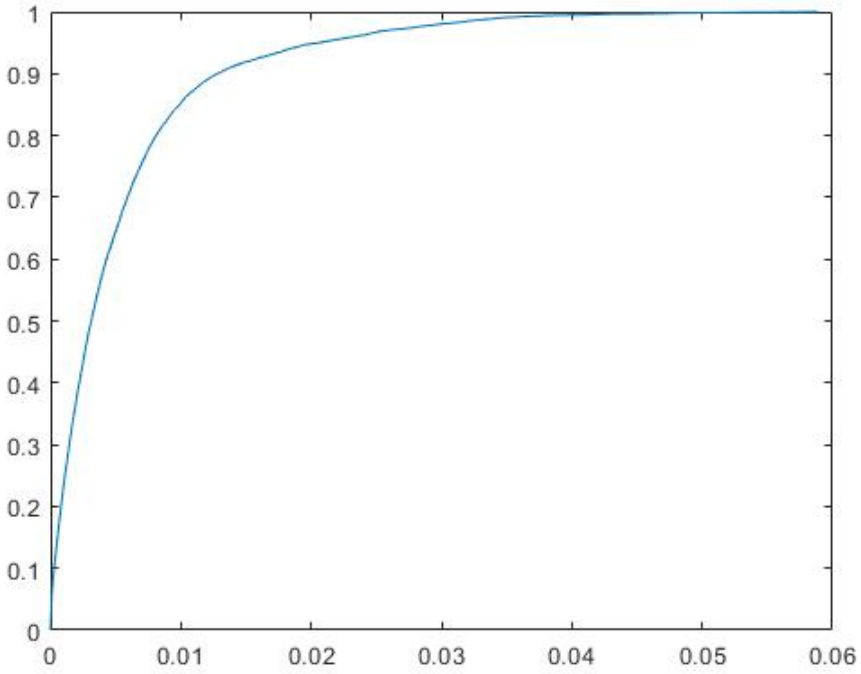

Figure 5: Response time at 3 servers

Figure 6 represents the response time at server 4. The average response time at server 4 is 0.0014.

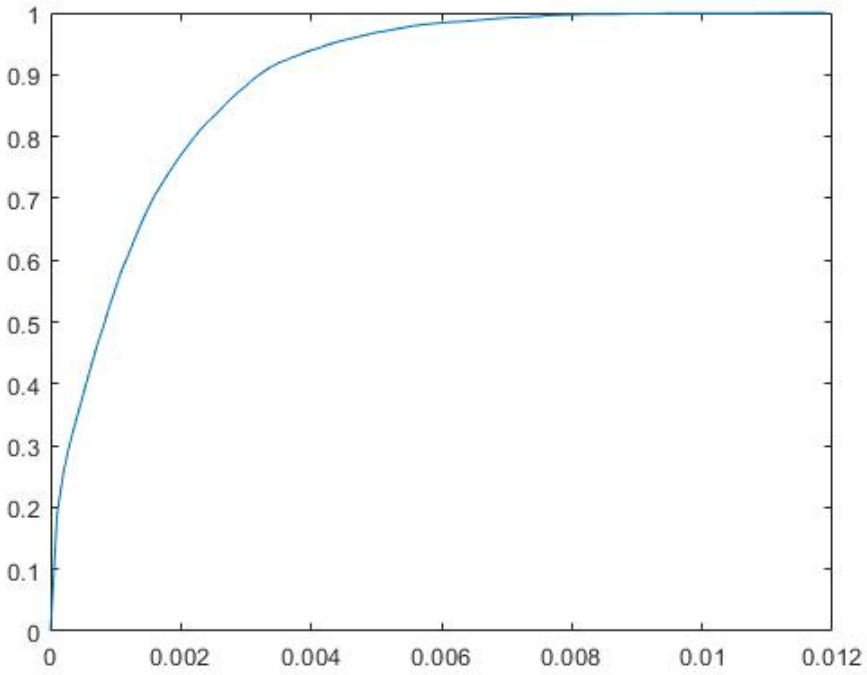

Figure 6: Response time at 4 servers

Figure 7 represents the response time at server 4. The average response time at server 5 is 0.0010.

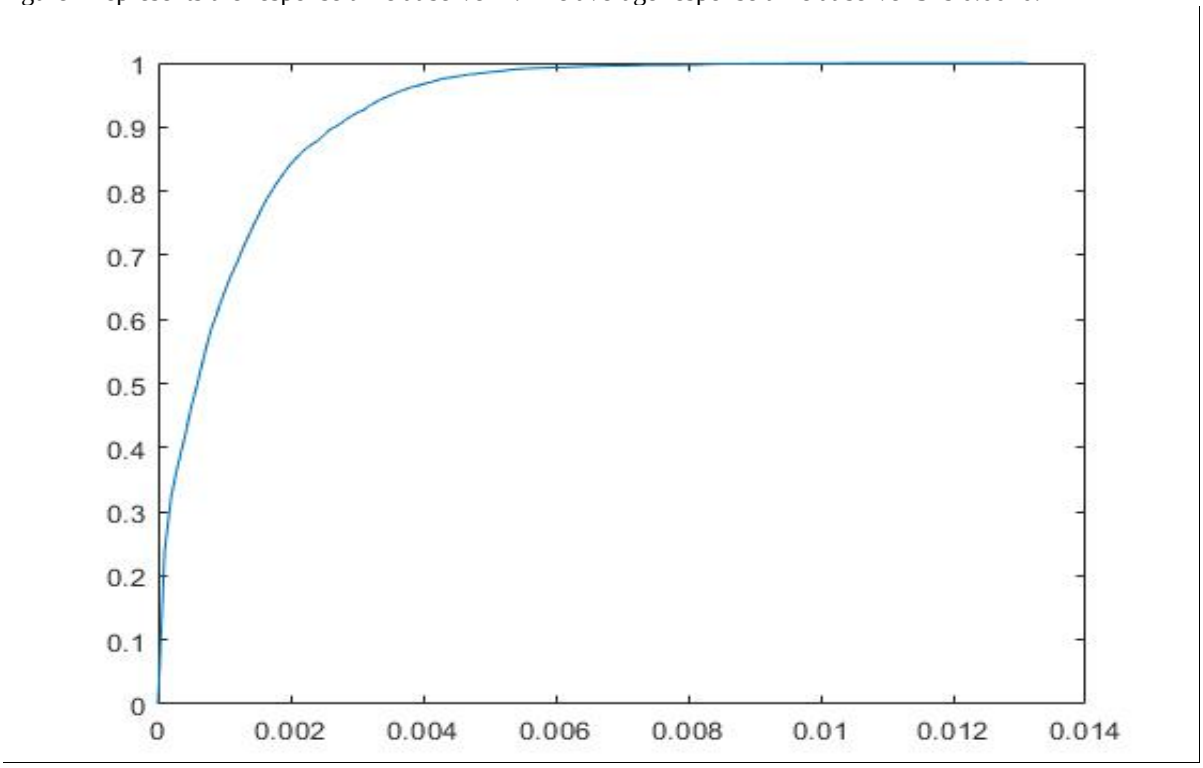

Figure 7: Response time at 5 servers

Table 2 represents average response time at different servers. If the number of servers increases the average response time to be decreased.

Table 2: Average response time at different servers

Number of servers	1	2	3	4	5
Average Response Time	0.0767	0.0728	0.0046	0.0014	0.0010

4.2.3 The Effect of Caching

The system simulates by reducing pC until it reaches to 0 or no cashing. The resulting average response time per server is compared with the original system and demonstrates the simulated results. The infinite buffer capacity approximation at the storage servers and pC = 0 are considered. Based on the known analytic results, the approximate CDF of the system response time is provided.

4.2.4 Service Denial

The Spotify back-end performance is measured for the original system in terms of service denial. That is, the storage servers are dropped the percentage of client requests as their buffer is full. For each storage server, the probability of service denial is provided and the result is evaluated and shown in table 3.

Table 3: Performance measurements at different servers

Number of Server	1	2	3	4	5
Dropped Packets (DP)	3843	1500	0	0	0
Total number of packets	14755	12610	8390	4370	2190
Probability of dropped packet	0.2604	0.1189	0	0	0

4.2.5 Load Balancing

At each storage server, the sample variance ($\hat{}\sigma^2$) of the average response times is determined:

$$\sigma^2 = \frac{1}{C}\sum_{j=1}^{C}(T_C^j - \mu T_C)^2$$

Where μT_C indicates the sample mean of the average response times by considering all storage servers and T j C denotes the measured average response time for server j:

$$\mu^{\wedge} T_C = \frac{1}{C}\sum_{K=1}^{C} T_C^K$$

At the storage servers, an allocation of a different song is suggested to reduce the variations between the response times. The variances at different servers are shown in table 4.

Table 4: Server response time at different variations

Number of Server	1	2	3	4	5
Variance	1.0178e-04	1.4600e-04	1.4653e-05	2.6036e-06	1.5914e-06

For the same client request demand (λ) and song popularity vector Q, the system is simulated by considering the proposed song allocation vector, C´. The new average response times is measured for the storage servers. These simulated results are compared with the original system and evaluated them whether the proposal was good or not.

Conclusions

In this paper, real time case study is conducted by considering the Spotify server streaming platform. In real time scenario, analytic tools of queuing theory are unable to evaluate the performance of the system. The proposed M/G/1 model is developed for real time applications and it has been evaluated in terms of service denial, service duration and response time.

References:

1. Swanson, Kate. "A Case Study on Spotify: Exploring Perceptions of the Music Streaming Service." *MEIEA Journal* 13, no. 1 (2013).
2. Voigt, Kai-Ingo, Oana Buliga, and Kathrin Michl. "Passion for Music: The Case of Spotify." In *Business Model Pioneers*, pp. 143-155. Springer, Cham, 2017.
3. Aguiar, Luis, and Joel Waldfogel. *Platforms, promotion, and product discovery: Evidence from spotify playlists*. No. w24713. National Bureau of Economic Research, 2018.
4. Ramos, Esmeralda Florez, and Knut Blind. "Data portability effects on data-driven innovation of online platforms: Analyzing Spotify." *Telecommunications Policy* 44, no. 9 (2020): 102026.
5. Schettino, Vinicius J., Regina Braga, José Maria N. David, and Marco Antônio P. Araújo. "Spotify characterization as a software ecosystem." In *Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse*, pp. 1-10. 2017.
6. van de Haar, Ilse, Cecilie Pedersen Broberg, and Ifigenia Doshoris. "How Artificial Intelligence is changing The Relationship Between The Consumer and Brand in The Music Industry." *LBMG Strategic Brand Management-Masters Paper Series* (2019).

7. Lozić, Joško, and Goran Vojković. "“Financial” Aspects of Spotify Streaming Model." In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1446-1450. IEEE.
8. Hänninen, Mikko, and Lauri Paavola. "Digital Platforms and Industry Change." In *Society as an Interaction Space*, pp. 213-226. Springer, Singapore, 2020.
9. Salameh, Abdallah, and Julian Bass. "Spotify tailoring for B2B product development." In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 61-65. IEEE, 2019.
10. Razlogova, Elena. "Provincializing Spotify: Radio, algorithms and conviviality." *radio journal: international studies in broadcast & audio media* 18, no. 1 (2020): 29-42.
11. Negus, Keith. "From creator to data: the post-record music industry and the digital conglomerates." *Media, Culture & Society* 41, no. 3 (2019): 367-384.
12. Kumar, Rakesh, and Sapana Sharma. "Transient solution of a two-heterogeneous servers' queuing system with retention of reneging customers." *Bulletin of the Malaysian Mathematical Sciences Society* 42, no. 1 (2019): 223-240.
13. Khomonenko, Anatoly D., Sergey I. Gindin, and Khalil Maad Modher. "A cloud computing model using multi-channel queuing system with cooling." In *2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM)*, pp. 103-106. IEEE, 2016.
14. Baumann, Hendrik, and Werner Sandmann. "Multi-server tandem queue with Markovian arrival process, phase-type service times, and finite buffers." *European Journal of Operational Research* 256, no. 1 (2017): 187-195.
15. Singh L. K, and Riktesh Srivastava. "Memory estimation of internet server using queuing theory: Comparative study between m/g/1, g/m/1 & g/g/1 queuing model." *a a* 2 (2007): 2.
16. Kuaban Godlove Suila, Rakesh Kumar, Bhavneet Singh Soodan, and Piotr Czekalski. "A Multi-Server Queuing Model With Balking and Correlated Reneging With Application in Health Care Management." *IEEE Access* 8 (2020): 169623-169639.
17. Horváth Illés, Rostislav Razumchik, and Miklós Telek. "The resampling M/G/1 non-preemptive LIFO queue and its application to systems with uncertain service time." *Performance Evaluation* 134 (2019): 102000.
18. Wu De-An, and Hideaki Takagi. "M/G/1 queue with multiple working vacations." *Performance Evaluation* 63, no. 7 (2006): 654-681.
19. Charan Singh Jeet, Madhu Jain, and Binay Kumar. "Analysis of MX/G/1 queueing model with balking and vacation." *International Journal of Operational Research* 19, no. 2 (2014): 154-173.
20. Jafarnejad Ghomi, Einollah, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Applying queue theory for modeling of cloud computing: A systematic review." *Concurrency and Computation: Practice and Experience* 31, no. 17 (2019): e5186.
21. Ke Jau-Chuan, and Fu-Min Chang. "Modified vacation policy for M/G/1 retrial queue with balking and feedback." *Computers & Industrial Engineering* 57, no. 1 (2009): 433-443.