EP2200 Course Project – 2016
Project II - Simulating a Spotify Server

Submitted by: Gaurav Puri (ITA)
gauravp@kth.se

Submitted to: Viktoria Fodor
vfodor@kth.se

The Pseudo Code: First of all we input all the data provided to us, such as λ, T_S^C, T_S^H, C, B, N, P_C. After this we defined the maximum time (T_{Max}) for which the system will run and the minimum time (T_{Min}) it will take to reach the steady state. We also specified that response time for all the 5 storage servers at the start is 0. Now we import the songs_allocation.txt and file_popularities.txt files in Matlab which were provided with this project.

Task one in hand was to generate the client arrival request for this given system between time t=0 to t<T_{Max}. To achieve this, we created a file with name arrival requests which is empty at t=0 and gets filled as time increases from 0 to T_{Max} with a factor of t+1/λ. As soon as t becomes greater than T_{Max} the loop ends. At the end of the loop, the file contains the number of requests generated and the time at which the requests where generated from t=0 to t<T_{Max}.

Task two in hand is to assign the songs to the requests generated above. For this we write the probability a song is requested from the file_popularities.txt file to a file named cdf.txt. Than we add these probabilities by adaptive numerical integration approach. Now we make another file named song_alloc.txt which has the same length as arrival request file and currently have no songs in it. Now we run a loop from 1 to length of arrival request and generate a random number between 0 to 1 and find which is the first accumulative probability greater than the random number and assign that song in the song_alloc.txt file.

Task three is to send the song requests to the servers based on the song_allocation.txt file provided with this project. We create a file named server whose length is 1 to length of arrival request. Now we have song_allocation.txt which has the server number where the songs are stored and song_alloc.txt where we have the requested song. We simply now match the requested song with the server it is stored on and send the request to that server.

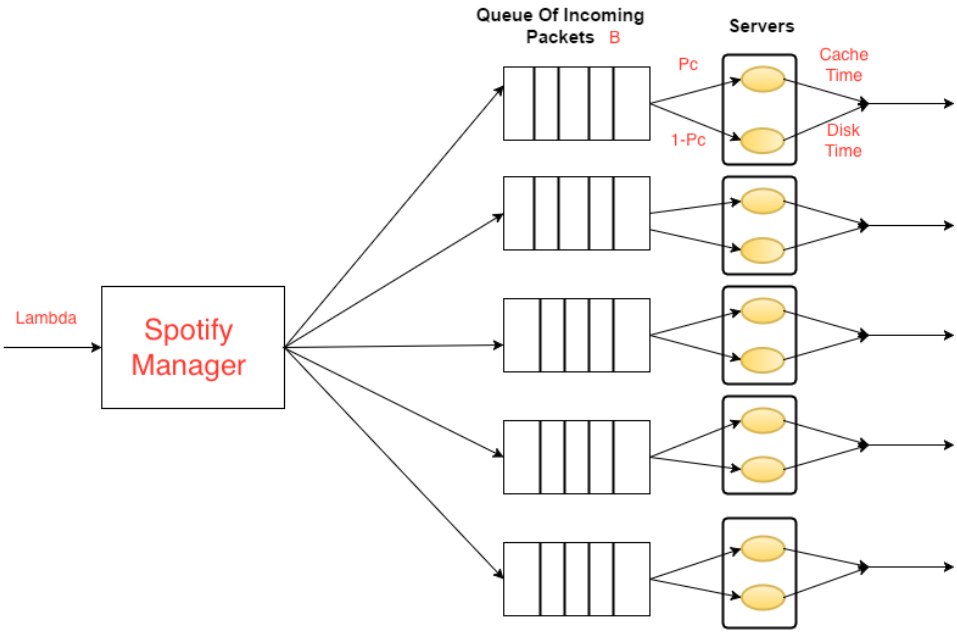
Now by this time we already have the arrival times, song requests and server where the request is to be sent. Now when we send the request to the server it have two scenarios firstly, if the song is in cache memory or secondly, if the song is in disk memory. We again generate a rand function and compare it with P_C, if the rand is less than P_C the request is served with a fixed time of 10^{-4} seconds else it will go to the disk memory and will have an exponential service time. The end of service time will be = t + service time of request (T_S^C or T_S^H). Response time is the difference between end of service time and arrival time.

To calculate drop request we run a loop where we mention that if length of queue < B than song goes to queue else drop request. We calculate variance and other parameters as mentioned in the project description as now we have the system functioning.

1. Queuing System

The queuing system in Spotify back end server can be defined as the waiting system because the customers may have to wait in the queue before the system can serve them.

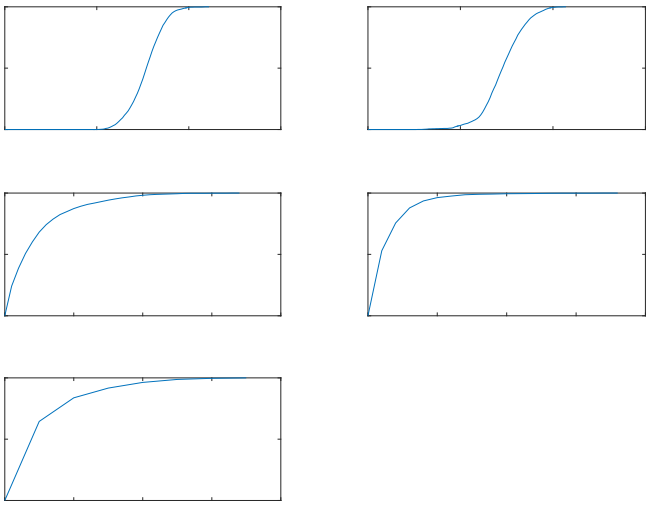
The Kendal notation for our system will be M/G/5/100, where M represents that the arrival process is a Poisson process, G represents that the system is general (unspecified), 5 represents the number of servers in our system and 100 represents the queue length of our system.



The arrival process in our case is a defined by the number of requests generated by the customers. The customer requests come according to the Poisson Process given by the parameter lambda (λ) whose value is given to be **5000** requests per second. The Spotify manager further sends the request to the storage server where the song is stored.

The service time distribution is unspecified in our case as the client request can be served with the help of two different service time interval. The first service time interval is a fixed time interval for the songs stored in the cache memory of the server whereas, the second is an exponential time interval for the songs stored in the hard disk. The service cache time is denoted by $\boldsymbol{T_s^c}$ which has a fixed value of $\mathbf{10^{-4}}$ seconds and the service disk time is denoted by $\boldsymbol{T_C^H}$ which has an exponentially distributed time with a value of $\mathbf{10^{-3}}$ seconds. The Probability that the song will have a service cache time is given by $\boldsymbol{P_C}$ which has a value of **0.25** and the probability that the song will have a service disk time is given by probability **1 - $\boldsymbol{P_C}$** which has a value of **0.75**.

2. **Response time of system with $\boldsymbol{P_C}$** = 0.25



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

3. **The Effect of Caching**

As you can see from the table below the average response time of the storage servers is increasing with the decrease in cache probability. Major increase in response time can be noticed for servers 1, 2 and 3 with minimal increase in servers 4 and 5. Overall we can say that the performance of storage servers has decreased with the decrease of cache probability. The servers are giving their worst performance when cache probability is zero and the system has infinite buffer.

	Average Response Time with decreasing values of P_C				
Number of Server	P_C = 0.25	P_C = 0.15	P_C = 0.05	P_C = 0.00	P_C = 0.00 with infinite buffer
1	0.0735	0.0845	0.0960	0.1001	3.8833
2	0.0732	0.0859	0.0946	0.0972	3.0862
3	0.0040	0.0062	0.0229	0.0396	0.1476
4	0.0015	0.0015	0.0018	0.0020	0.0020
5	0.0010	0.0012	0.0012	0.0014	0.0014

4. **Service Denial**

Here the total number of packets is the sum of the requests which were successfully served by the storage system and dropped packets by the storage system.

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 packet dropped	0.2604	0.1189	0	0	0

5. **Load Balancing**

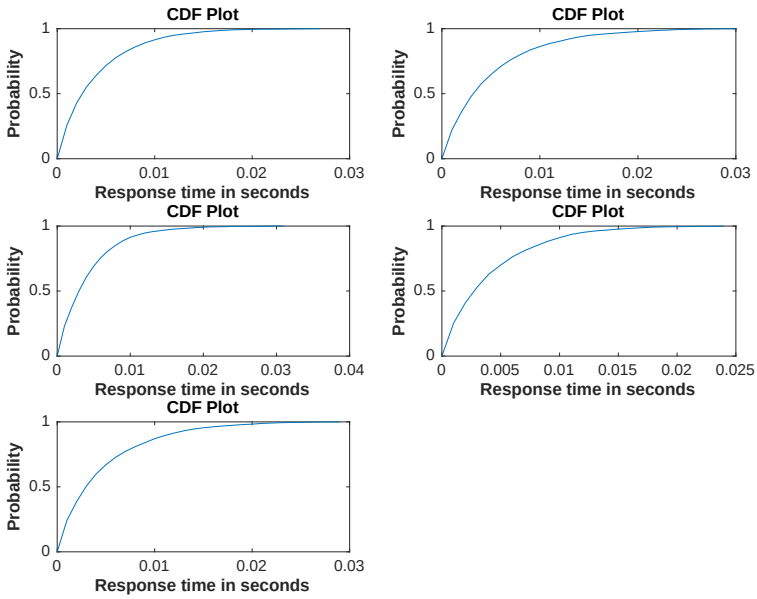
Sample variance for each storage server is as shown below:

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

The reason we see different average response times for the storage servers is due to the song distribution among the servers along with the distribution of songs on basis of popularity is also very uneven. From our findings in the above section it can be observed that most of the requests are served by server 1 and server 2. This is because server 1 and 2 have the most songs mapped along with high song popularity. Whereas, servers 4 and 5 have the least number of requests with less songs mapped along with low song popularity.

We can improve the system by allocating the high popularity songs to all the 5 storage servers equally along with equal number of song allocation.

Please find the CDF for new allocation system below:



Number of Server	1	2	3	4	5
Average Response Time	0.0039	0.0048	0.0042	0.0040	0.0046

So now with this new allocation system and song popularity distribution the response time for all 5 storage servers is approximately the same.