Performance Evaluation and Applications Project 2023 - 2024

Concert Ticket Service – Project Type C

Simone Scevaroli - 10913296

Project Statement

Project Statement

- A concert ticketing service has to handle a large number of requests in a short amount of time
- The system can allow at most N = 1000 pending requests at a time

The service is composed by four stages:

1. Welcome message (W)
2. Seat Selection (S)
3. Payment processor (P)
4. Ticket issuing (T)

Goals of the Project

Goals

1. *Average response time* below 5 minutes
2. *Average drop probability* below 25%

using the fewest possible cores per stage

Solution

Fitting

- First of all, it was necessary to fit the given traces (using MATLAB), determing which distribution and parameters best follows the curve of durations for each stage
- To obtain all the parameters for the distributions in minutes, the durations of the traces were divided by 60*1000 (since they were in milliseconds)
- All the plots are commented in the MATLAB file. If you wish to view the plots, you need to remove the «/60000» initially added to the dataset and uncomment the desired plot

Fitting

- Looking at the plots of the 4 traces and their coefficents of variation (cv), it was possible to try only a few fits per trace

- Trace W -> Weibull or (having cv < 1) a Hypoexp / Erlang
- Trace S -> an Exponential (having cv ~ 1)
- Trace P -> an Exponential (having cv ~ 1)
- Trace I -> a Hyperexp (having cv > 1)

Fitting

- The following results were found:

1. Trace W -> **Erlang distribution**
   Params -> $k=36$, $\lambda=288008 \text{ min}^{-1}$

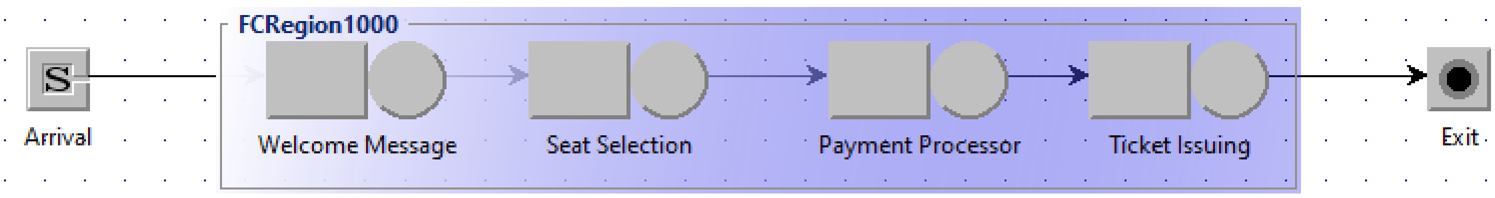
2. Trace S -> **Exponential distribution**
   Params -> $\lambda=239.124 \text{ min}^{-1}$

3. Trace P -> **Exponential distribution**
   Params -> $\lambda=1203.19 \text{ min}^{-1}$

4. Trace I -> **HyperExp distribution**
   Params -> $\lambda_1=537.042 \text{ min}^{-1}$, $\lambda_2=2149.71 \text{ min}^{-1}$, $p_1=0.203235$

Simulation using JMT

- After identifying all the necessary parameters, I used JMT to implement the following queuing network, representing the topology of the system



Simulation using JMT

- The system shown before is characterized by:
  - An **open class**, modeled as a Markov-Modulated Poisson Process (mmpp2) with λ_0=6000 req/min, λ_1=300 req/min, σ_0=1/480min and σ_1=1/9600min. It represents the arrivals
  - **Four queue stations**, each representing a stage of the system and modeled based on the distribution and parameters found during the fitting process
  - A **finite capacity region**, with a size of 1000. This ensures that no more than 1000 requests are present in the system at any given time

Performance indexes

- The following indexes were computed at each iteration to understand how to modify the systems, adding cores to stages when necessary:
  - Utilization (for all stages)
  - Source Throughput
  - System Response Time
  - System Drop Rate (Number of drops)
  - System Drop Percentage (SysDropRate/SysThroughput)

Other JMT parameters

- For each simulation I used the following JMT parameters:
  - Maximum number of samples -> 10M
  - Confidence interval -> 0.99
  - Maximum relative error -> 0.03

Running simulations

- All the results from my simulations were put into an Excel file, documenting important values for each simulation
- The Excel file is inside the folder so you can have a look at all the iterations and the results obtained
- In particular, to determine the allocation of cores for the next run, I identified bottlenecks by looking at the utilizitation of each stage: if the average utilization was close to 1, I increased the number of cores for the corresponding stage

POLITECNICO
MILANO 1863

Results

Results

- Through multiple iterations of this process, I discovered that:
  - The average response time was already below 5 minutes even with just 1 core per stage. However, the drop rate percentage exceeded 95%, which is unaccetable for a concert ticket service
  - It took **11 iterations** to reach the final configuration that minimized the number of core per stage, achieving both an average response time and an average drop rate percentage below the given threshold

Results

- The final configuration found is:
  - **1 core** for Welcome Message stage
  - **19 cores** for Seat Selection stage
  - **4 cores** for Payment processor stage
  - **4 cores** for Ticket Issuing stage

Conclusions

Conclusions

- It's important to note that, with the discovered configuration, the drop rate percentage is just slightly below 25%. If additional resources (in terms of cores/money) are available, it's better to increase the number of cores in the Seat Selection stage (the last bottleneck) by at least one. This adjustment aims to further reduce the drop rate percentage, ensuring a more secure and stable outcome
- It was clear after the fitting phase that the Seat Selection stage would frequently be the bottleneck and would require the highest number of cores, because its exponential distribution has a very low lambda compared to the other exponentials, indicating a lower number of requests processed per minute