AM115 Final Project

Ream Gebrekidan

April 26th, 2023

Github Repo

0.1 Abstract

I aim to create a model that reflects the interactions between a distributed system providing some digital service and clients making requests to said system. This model should allow companies to glean insights into how to best setup their distributed systems to quickly resolve client requests. Through implementing a stochastic model I have been able to demonstrate how a company could optimize their distributed system, highlighting the importance of processing power and replication in a system. Furthermore, I demonstrate how through replication a system can become highly reliable, even when the individual parts of said system are not highly reliable.

1 Introduction

1.1 The Problem

In the modern day, companies are relying more and more on providing digital services to their clients. This is most evident in companies such as Google and Amazon, which rely on providing consistent digital services in order to earn billions of dollars in revenue. This trend extends to companies of all sizes however. As such, providing consistent server uptimes to allow customers to consistently access their digital services is paramount. Even brief downtimes can cost these corporations thousands to millions of dollars in revenue loss [3]. Thus it is a benefit for companies to maximize their digital service uptimes and the speed with which they can handle client requests.

To improve both the speed and reliability of the digital services they provide, a company will often use a distributed system. By having multiple servers that can flexibly coordinate with each other to handle requests, companies reap various benefits. This includes ensuring that a server is always available to fulfil client requests quickly. As companies design their distributed systems they should keep these goals in mind to maximize their user experience and revenue. This model is aimed at helping companies better analyze how their distributed system design will impact the experience of their users.

1.2 Mathematical Background - Stochastic Models

To analyze this problem I've implemented a stochastic model. This problem lends itself well to a stochastic model because of the dynamics involved. At any point in time there is a random number of users making requests to a company's servers. Similarly, there exists a random probability that any given server in a company's system is functioning.

1.3 Motivations

In this model, I hope to allow companies to predict the experience of their users depending on the amount of money they are willing to spend and the characterization of the servers they buy. This would serve as a powerful starting point for a company deciding how they want to set up or alter how they provide their digital services.

2 Model Design

2.1 Variables

The model defines:
$S \rightarrow$ The set of servers in a system
$s(c, r, p) \rightarrow$ A server with c cores (The number of requests the server can handle at any given time), reliability of $r \in [0, 1)$ (the proportion of time the server is up), and p, the number of requests the server is currently processing.
$C(r, p) \rightarrow$ a cost function for a server with reliability r and c cores.
$R(t) \sim N$, The number of requests the distributed system receives at any given time, distributed normally with mean/variance that changes depending on the time of day.

2.2 Stochastic Simulation

This simulation works by stepping through time during a day and performing the following steps: 1) Sampling from $R(t)$ to find how many clients are making a request at that time and adding those requests to a queue, 2) Iterating through the set of servers S and assigning requests from the queue based on the capacity of the server and whether or not the server is up at any given time, 3) Having servers resolve those requests and then recording the time it took for each request to be resolved.

3 Analysis of the Model

3.1 Initial Conditions

To test the behaviour of this model I began by defining a distribution of client requests. To simulate real-world conditions where there is a random amount of users at a given time, but there exist times during which more people tend to access a digit service during a particular time of day, I define the distribution of clients to as follows Where $t \rightarrow$ time of day (military time in hours):

$$C(t) \sim \begin{cases} N(100, 10) & \text{for } 7 < t < 9, 17 < t < 19 \\ N(300, 10) & \text{for all other } t \end{cases}$$

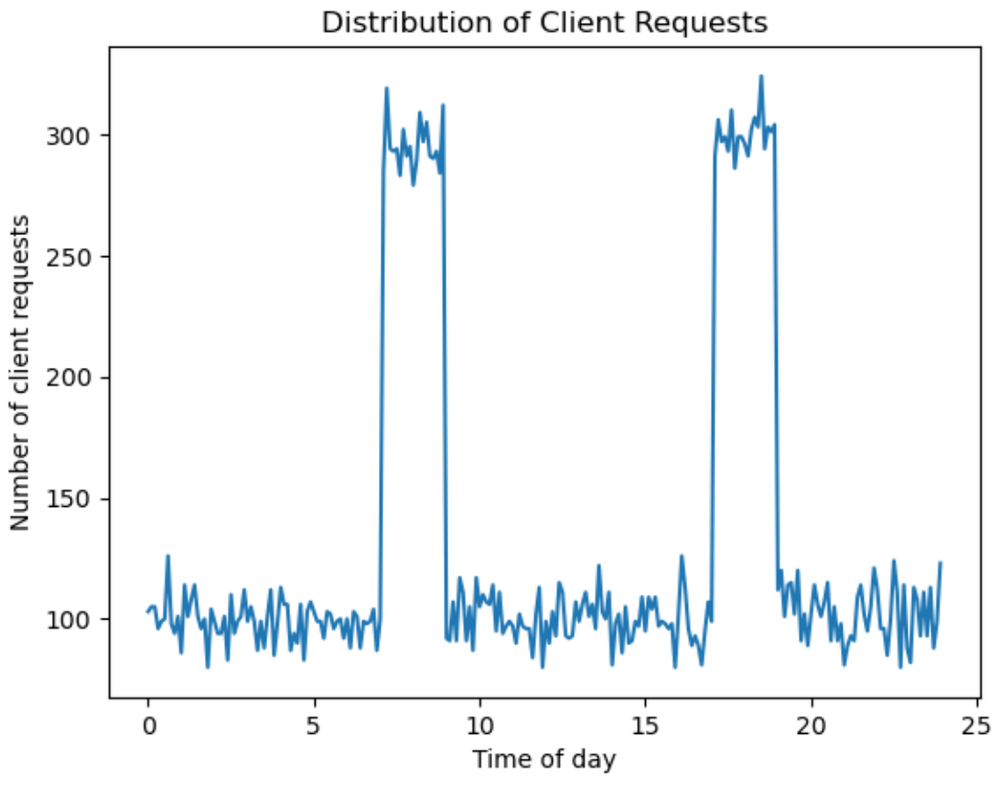

Figure 1: Sample distribution of client requests throughout a given day

From there, I defined a cost function that scales exponentially based on the reliability of the server and linearly based on its processing power (adding onto some base cost), creating a cost function comparable to the current market [1]. This resulted in the following cost distribution:

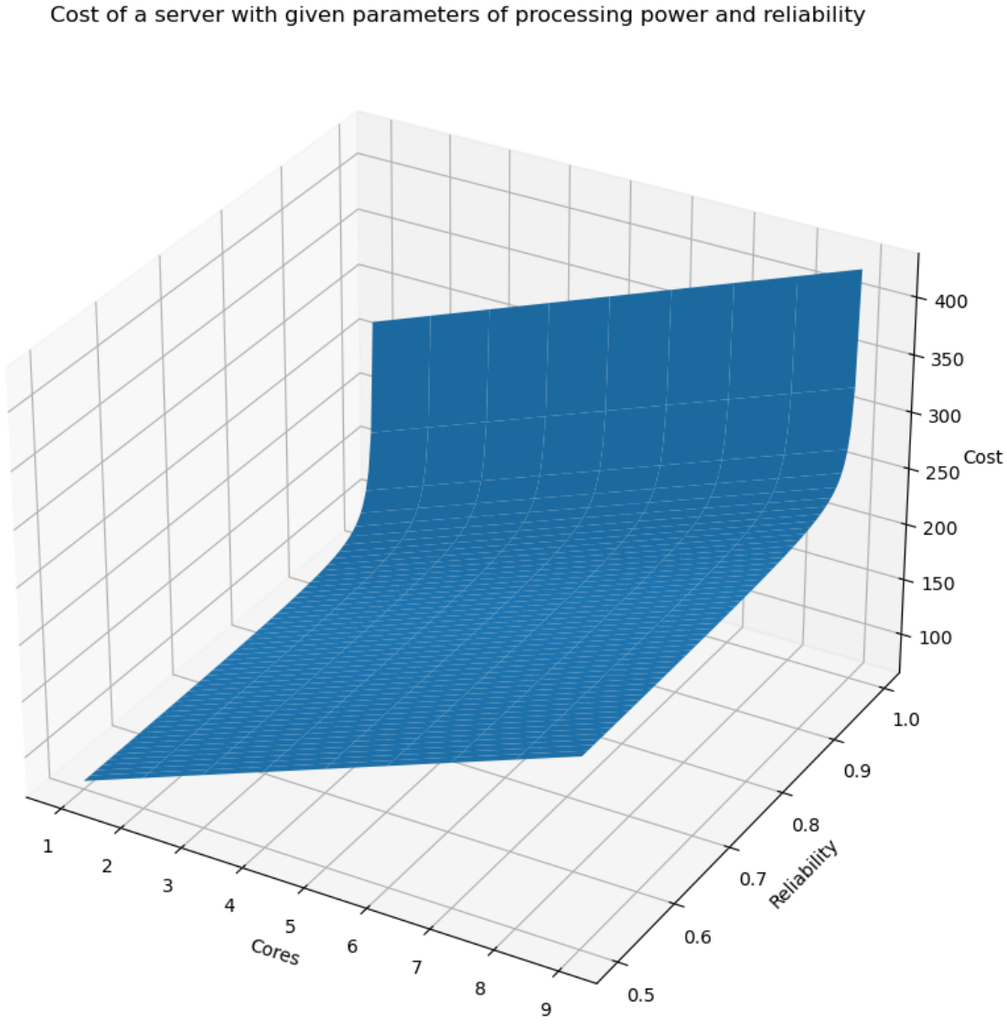

Figure 2: Cost of a server versus parameters of processing power and reliability

Following this, I created an initial server configuration of a set of 10 servers, each with the ability to handle 10 requests at a time and an uptime ratio of 0.99. The results of this simulation can be seen in Figure 3.

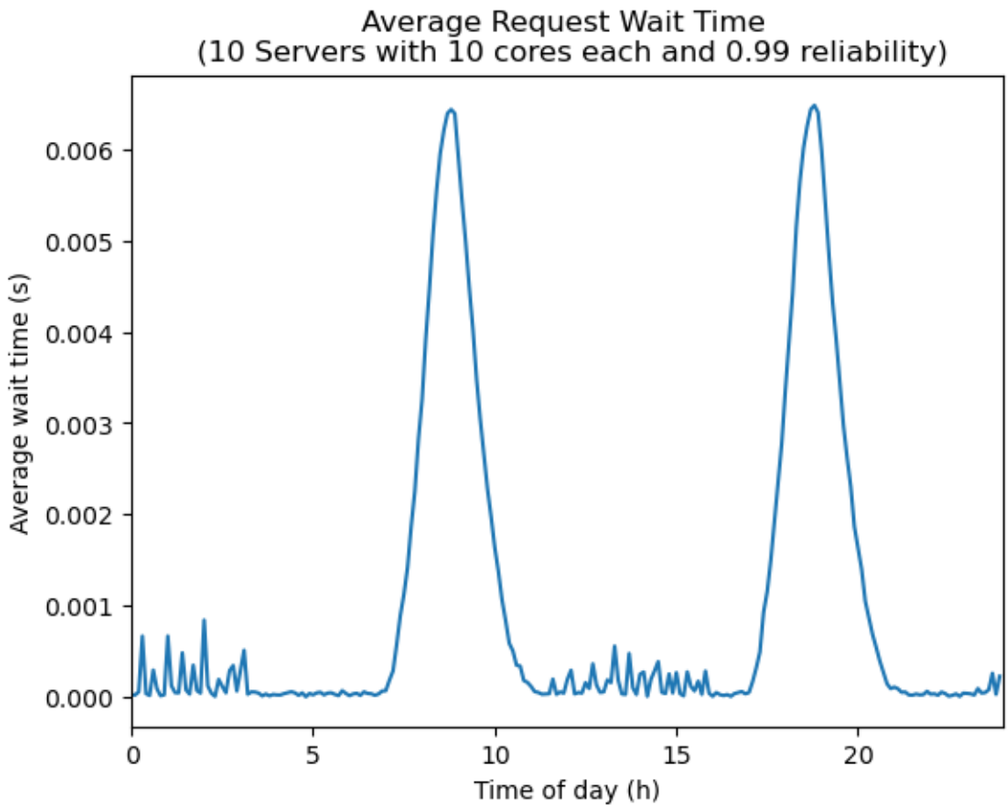

Figure 3: Average amount of time it takes to resolve a request initiated at a given time of day

In this figure, we see that during regular/low usage times requests are fulfilled nearly instantly whereas during peak usage times requests are slower to fulfil. This shows how, during the peak times of usage, the amount of requests exceeds the capacity of the distributed system and the amount of queued requests grows. Based on this observation, a company may opt to either accept this delay in fulfilling requests or may seek to obtain a more powerful distributed system.

3.2 Model Sensitivity

Following this initial model, I decided to investigate how changing the properties of the servers that make up the system affects the system's performance. To that end, I set a moderate budget of $10,000 [1]. I then tested three approaches to system design: 1) Buying the most reliable server 2) Buying the most powerful server 3) Buying a high number of inexpensive servers (maximizing each quality based on the cost function $C()$). The results of these experiments can be seen in Figure 4

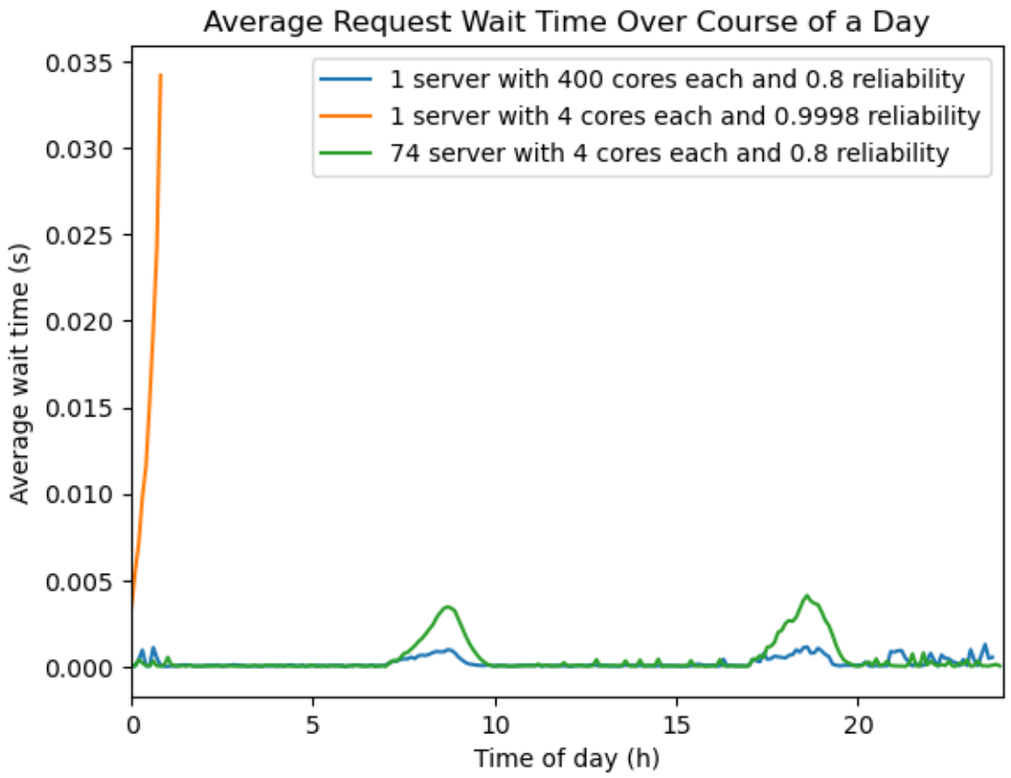

Figure 4: Average amount of time it takes to resolve a request initiated at a given time of day for different server configurations

Here we see that the network comprised of the most reliable server (orange) quickly cannot keep up with the incoming requests and is rapidly overwhelmed by requests after just a couple of hours in the day. In contrast, both the high-power server (blue) and the highly replicated system (green) are able to keep up with the incoming client requests throughout the day. However, there are several key differences between these two server setups. We see that the singular, powerful server with 400 cores is able to more rapidly complete requests during peak client request times. This makes sense as when running, this system can handle up to 400 requests at a given time whereas when fully running, the highly replicated system configuration can only handle 296 requests at any given time. That being said, there are also several peaks in response time that are noticeable in the single server system that are not present in the highly replicated system. This makes sense as it is more likely for the single server to go down and not be able to handle any requests than it is for the entire set of 74 servers to fail simultaneously, meaning that the highly replicated system is always processing some requests, whereas there are times when no requests are being fulfilled by the single server system.

4 Discussion

4.1 Model Behavior

Here we have an adaptable stochastic model which provides insight into user wait times for digital services based on a company's distributed system design. We find that wait times increase when the amount of client requests exceeds the capacity of the distributed system as expected but tend to remain consistent and low otherwise. Furthermore, we see the effects of not having enough replication in a distributed system, leading to sharp spikes in wait times for users whenever a server crashes.

4.2 Misc. New Insights

This model teaches us several things about the design of a distributed system. First and foremost, it demonstrates the importance of having a high overall capacity and independent replication in a system. Additionally, it shows that in a highly replicated system, the relia-

bility of each individual server is not as important. When these conditions are met, there is minimal impact when any server crashes and overall a minimum wait time for client requests.

4.3 Results in Context

This model can serve as a powerful starting point for companies looking to setup their own distributed system to fulfill the digital services they seek to provide. This is demonstrated in section 3.2, wherein by setting a budget and having an approximate idea of how clients will interact with a service (estimate the distribution of client requests), a company could optimize their system. The example demonstrates the importance of balancing the number of servers with the quality of each server (measured by each server's reliability and processing power). By having a highly replicated system that has sufficient capacity, a company can reply to client requests consistently and quickly.

4.4 Shortcomings and Future Embellishments

This being said, this model has several limitations. Firstly, this model assumes uniformity in both servers and requests. In a real distributed system, each server is not identical to the next but rather only certain servers can respond to certain requests. Additionally, while if well coded all requests are similarly small in size, in the real world certain requests require more resources than others. While these nuances were abstracted away when formulating this model, it could be interesting to include these nuances into the model and see how they affect the results. Additionally, this model does not explicitly include possible bottlenecks that arise in a distributed system (such as from a load balancer that distributes requests throughout a distributed system) and slow down the overall response speed of a system. While this is implicitly included by looping through the servers to distribute requests, this still ignores other possible bottlenecks that could arise. It would be interesting to see how introducing bottlenecks into model of the system affects the results.
Taking these shortcomings into account, it would be interesting to introduce the following elements into the model: 1) Allow for more complex distributed system designs. This includes modeling systems that are not fully replicated and including dependency in failures 2) Incorporate penalties for failures of the entire system. In the real world whole system failures can incur severe penalties such as loss of information and loss of customers. 3) Compare this model to the efficacy and cost of cloud services. Many companies opt to use services such as Amazon Web Services (AWS) to create their distributed system rather than creating their own distributed systems from scratch. It would be interesting to try and model at which point creating a custom system would make more sense than using something like AWS.

5 Summary

I have created a stochastic model which simulates the interactions between a distributed system and client requests. This model defines a cost function that defines the cost of a particular distributed system. Furthermore, it characterizes a distributed system in terms of the number of servers, the processing power of those servers, and the reliability of each of those servers. Together, this allows someone to model how well a given distributed system design can quickly respond to requests, providing a stepping stone for anyone looking to create a distributed system to fulfill digital services.

References

[1] Abacus, *How Much Does a Server Cost for a Large Business?*. 2021. https://goabacus.com/how-much-server-cost-large-business/

[2] Miller, Kaycee, *What Happened with the Amazon S3 Server Outage that Broke the Internet*. 2017. https://www.rentecdirect.com/blog/temporarily-unavailable-what-it-really-means-for-your-business/

[3] StatusCake Team, *The Most Expensive Website Downtime Periods in History*. 2020. https://www.statuscake.com/blog/the-most-expensive-website-downtime-periods-in-history/