Resource mapping

TwitterV2

Table of Contents

[Introduction 2](#_Toc164023213)

[User Registrations 2](#_Toc164023214)

[Tweet posts 2](#_Toc164023215)

[Load tweets 3](#_Toc164023216)

# Introduction

The purpose of this document is to get an idea of how much load TwitterV2 would be under on a second-by-second basis. Of course, the numbers are rough estimates, as it is impossible to accurately predict how many users will be active at a time and what they would be doing. The estimates are based on an average of 500 million users per day.

# User Registrations

A diagram of a user registration

Description automatically generated

# Tweet posts

Not all users tweet, most just read others’ tweets, so the calculation is made for 100 million.

A diagram of a diagram

Description automatically generated

# Load tweets

This estimation is based on each of the 500 million users loading an average of 20 tweets per day.

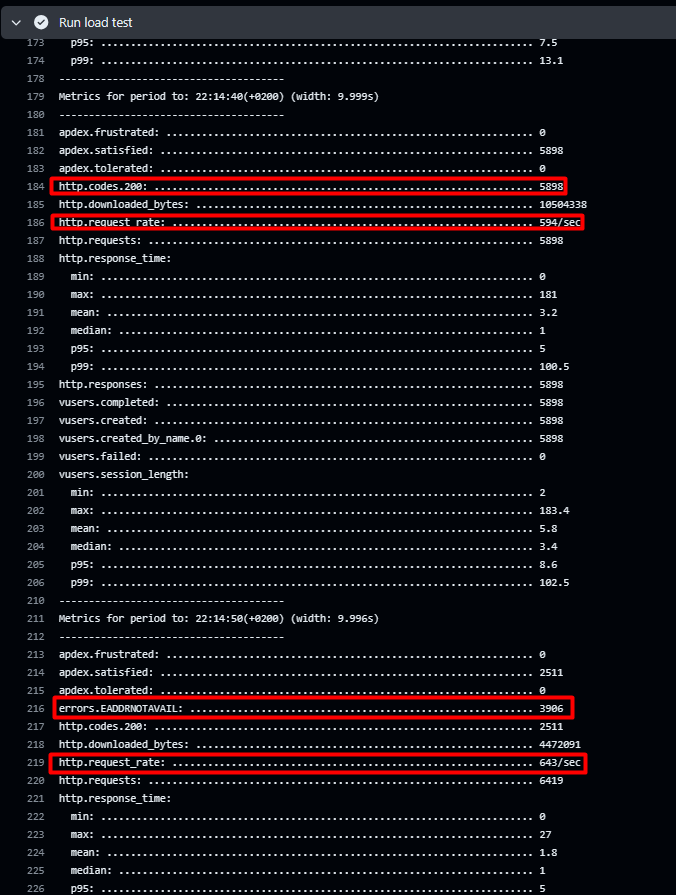
A diagram of a diagram of a tweet

Description automatically generated

# Findings from load tests

## Tweet loading

I create a load test based on the assumption that each user would be loading an average of 20 tweets per day, which started getting failed requests at around 620 requests per second:



I scaled up the tweet service’s pods to see if the bottleneck was coming from there or the database:

A screenshot of a chat

Description automatically generated

The load test afterwards yielded the same result – requests started failing at around 620 per second. This means the bottleneck is in the database.

## Users logins

User logins had a 100% success rate up to 803 requests per second. Nothing changed after scaling the number of pods, so the bottleneck is in the database here as well. Nevertheless, it could take ~150 more requests per second than the tweet loading test. This implies that users and tweets should be separated into different services due to different scaling needs.

A screenshot of a computer

Description automatically generated

# Conclusion

The tweet loading side has the most resource consumption by far out of the current functionalities. Based on this information, I will be looking into separating reads and writes for the tweet service for the sake of scaling up only what needs to be scaled. Additionally, I will be implementing Azure’s CosmosDB with Mongo into TwitterV2 to get past the database bottleneck in the load tests. Thanks to its automatic handling of fluctuating traffic demands, it will be able to handle a higher load.