Carnegie Mellon University

Recitation 9

15440 - Distributed Systems - Spring 2023

P3 Checkpoint 3

Mehal Kashyap& Ryan Santhoshkumar, March 29, 2023

Checkpoint 3: Dynamic Loads

- Arrival rates will fluctuate suddenly
  - May have large number of requests followed by quick turnaround and vice versa.
  - What happens if your VMs don't boot up in time? Will incur > unhappy clients and > VM overhead.
- Overall similar to Checkpoint 2, but speed is more critical
  - Speed referring to both detection and adaptation
- Think of how you queue requests
  - Should you sometimes drop requests preemptively to improve a larger amount of user speedup?

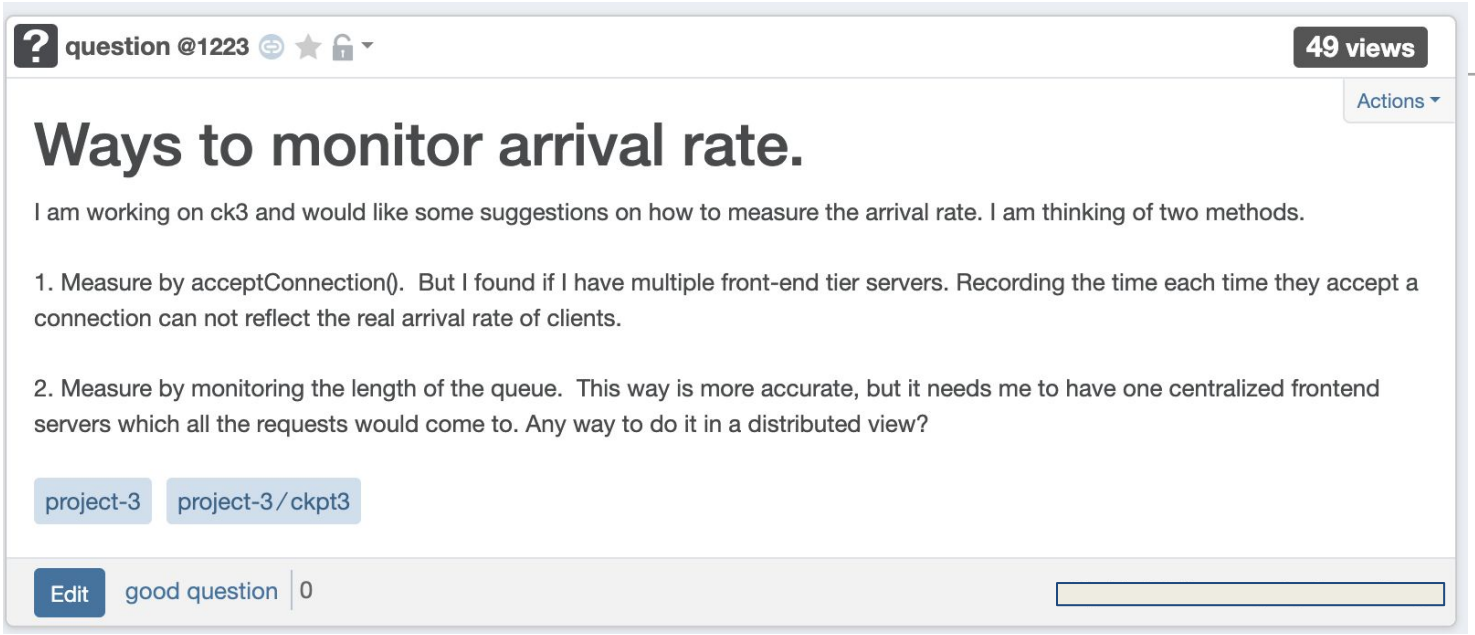
Overall Takeaways

- Different types of operations can lead to different bottlenecks
  - Separation of work into tiers
- Fixing one bottleneck can move it to a different place!
- Simply adding more resources will not always help
  - Need to target specific issues
- Being able to identify a bottleneck and being able to fix it are distinct
  - Response time can be critical

Debugging

- View for logging help:
  - https://stackoverflow.com/questions/15758685/how-to-write-logs-in-text-file-when-using-java-util-logging-logger
  - Comment out logs before submitting, they can slow you down
- Benchmark plotting different measurements can be very useful in identifying certain trends

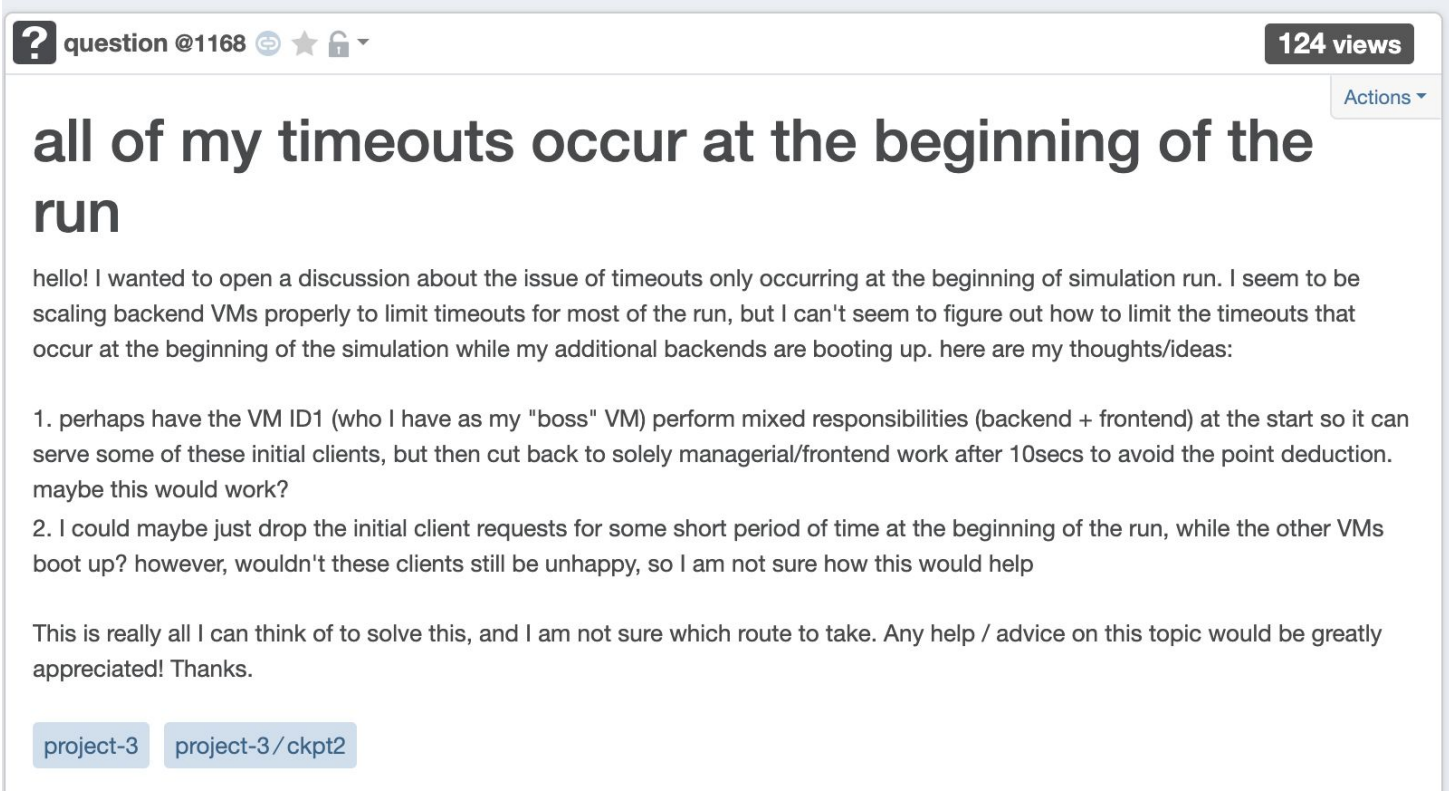
P3 FAQ


question @1223
49 views
Actions
Ways to monitor arrival rate.
I am working on ck3 and would like some suggestions on how to measure the arrival rate. I am thinking of two methods.
1. Measure by acceptConnection(). But I found if I have multiple front-end tier servers. Recording the time each time they accept a connection can not reflect the real arrival rate of clients.
2. Measure by monitoring the length of the queue. This way is more accurate, but it needs me to have one centralized frontend servers which all the requests would come to. Any way to do it in a distributed view?
project-3
project-3/ckpt3
Edit
good question
0

Answer 1

- You can use RMI to communicate between the front end servers
- RMI is much faster than any of the other functions
- Think about how to coordinate between one server–
  - EIG, whenever one server accepts a connection, it should communicate this with the main server in order to account for the differing queue length

P3 FAQ



question @1168

124 views

Actions

all of my timeouts occur at the beginning of the run

hello! I wanted to open a discussion about the issue of timeouts only occurring at the beginning of simulation run. I seem to be scaling backend VMs properly to limit timeouts for most of the run, but I can't seem to figure out how to limit the timeouts that occur at the beginning of the simulation while my additional backends are booting up. here are my thoughts/ideas:

1. perhaps have the VM ID1 (who I have as my "boss" VM) perform mixed responsibilities (backend + frontend) at the start so it can serve some of these initial clients, but then cut back to solely managerial/frontend work after 10secs to avoid the point deduction. maybe this would work?
2. I could maybe just drop the initial client requests for some short period of time at the beginning of the run, while the other VMs boot up? however, wouldn't these clients still be unhappy, so I am not sure how this would help

This is really all I can think of to solve this, and I am not sure which route to take. Any help / advice on this topic would be greatly appreciated! Thanks.

project-3 project-3/ckpt2

Answer 2

- Should drop limited number of clients at the beginning of the run if they are arriving too quickly
- As VMs are booting, you should be either dropping or processing requests
  - Can immediately boot a backend so that the master server is not processing requests for too long while the other VMs are booting

P3 FAQ

Actions

Knowing when to drop clients

Hi there,

So I did some computations to check roughly how long it takes the request processing methods to run. I want to use these times to determine whether or not it is worth serving this client or if the client should be dropped. However, I feel like it may be helpful to know when the client requests joined the queue (when the client made the actual request), but I'm not sure it's possible to access this information. Should our computation just be rough and based on when the front-end server actually accepts that connection and starts working on that request?

Thanks!

Answer 3

- If your queue length gets too long, you should drop clients
  - If your queue length is too long, you won't be able to process the clients at the end of the queue anyway-> drop those clients
  - Up to you to figure out what "too long" means

P3 FAQ

- Strange RMI issues?
  - As long as this happens at the end, you should be okay

Last Tips

- Start and **submit early**
  - There may be slowdowns in submission processing near the end of the due date, so don't expect to get efficient feedback from Autolab then
- The system is inherently non-deterministic
  - Consecutive runs will **not** always get the same score
  - This slight change can result in vastly different results
  - **We are not able to delete later submissions!**
- You can earn Bonus points for Checkpoint 3

Design Document

- Should be a concise explanation of the major design decisions in your project
- Should cover the following:
  - How you coordinate the roles of the different server instances
  - How you decide how many servers in each tier to run
  - When you decide to add or remove servers
  - Including plots is helpful and describe what you learned about scaling out

Design Document

- Can cover the following:
  - High level design ideas that were used in the project and **why** were they chosen
  - What other alternatives were considered and why were they rejected.
- Ensure that the different ideas/sub-topics are cleanly orgranised in paragraphs. Use paragraph headings, bullet points etc.
- Not more than 1-2 pages in length

Code Style - General

References

- 15-440/640 Course Page
  https://www.andrew.cmu.edu/course/15-440/
- 15-213 Style Guidelines
  https://www.cs.cmu.edu/~213/codeStyle.html

Code Style - Reminder

- No magic numbers — use macros instead
- Ensure that the code is modular
  - Use helper functions instead of repeating the same code in different places
  - Try to wrap system calls that are repeatedly used into helper functions.
- Ensure that Lines no longer than 80 characters and functions are not more than 30-40 lines of code.
- Use intuitive naming of variables and functions. (Prefer camel casing in Java).

Code Style

- Consistent code formatting (white spaces, braces, indentation)
- No dead/test code
- Proper failure condition and error checking
- Use file headers, function headers to cover purpose, description, parameters and return values
- Limit the use of debugging print statements in the final code.

Code Style - Error Handling

- Always check return values for every call
  - Ensure that you are returning the correct error code back to the clients
- Deal with the errors with proper actions (send error codes/print error messages)

All Failures are born as Byzantine

- Bit flips (have to do checksum), power failures
- Explicit engineering needed to transfer them to failfast (checksum, predicate validation in optimistic methods)
- Faith based approach accepts Byzantine, but leases are willing to get performance, but are fail fast

Autolab deciding the interarrival time between clients:

Autolab when you submit the same code twice

Questions?