Performance Evaluation of Lock-Free Distributed Linked Lists

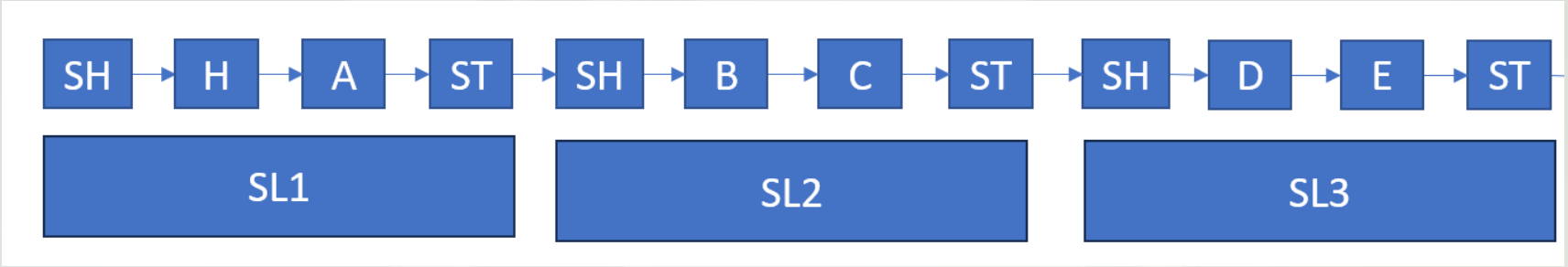
Raaghav Ravishankar

Doctoral Student

Computer Science and Engineering

Introducing Distributed Linked Lists

- List example (A chain of sublists):



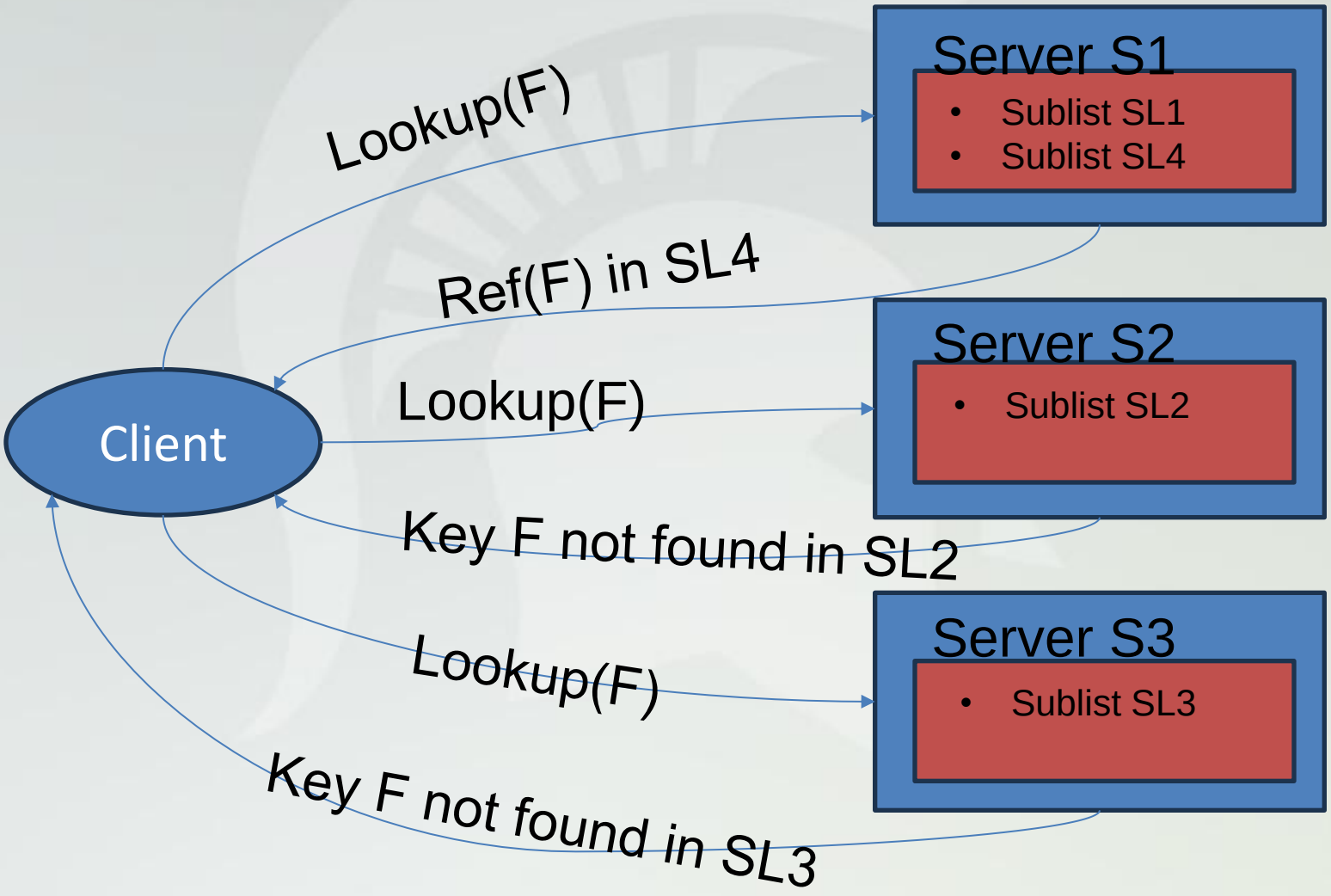
Here SH, ST are subheads and subtails of a sublist.

SL1, SL2, SL3 are thus sublists that together make up the list.

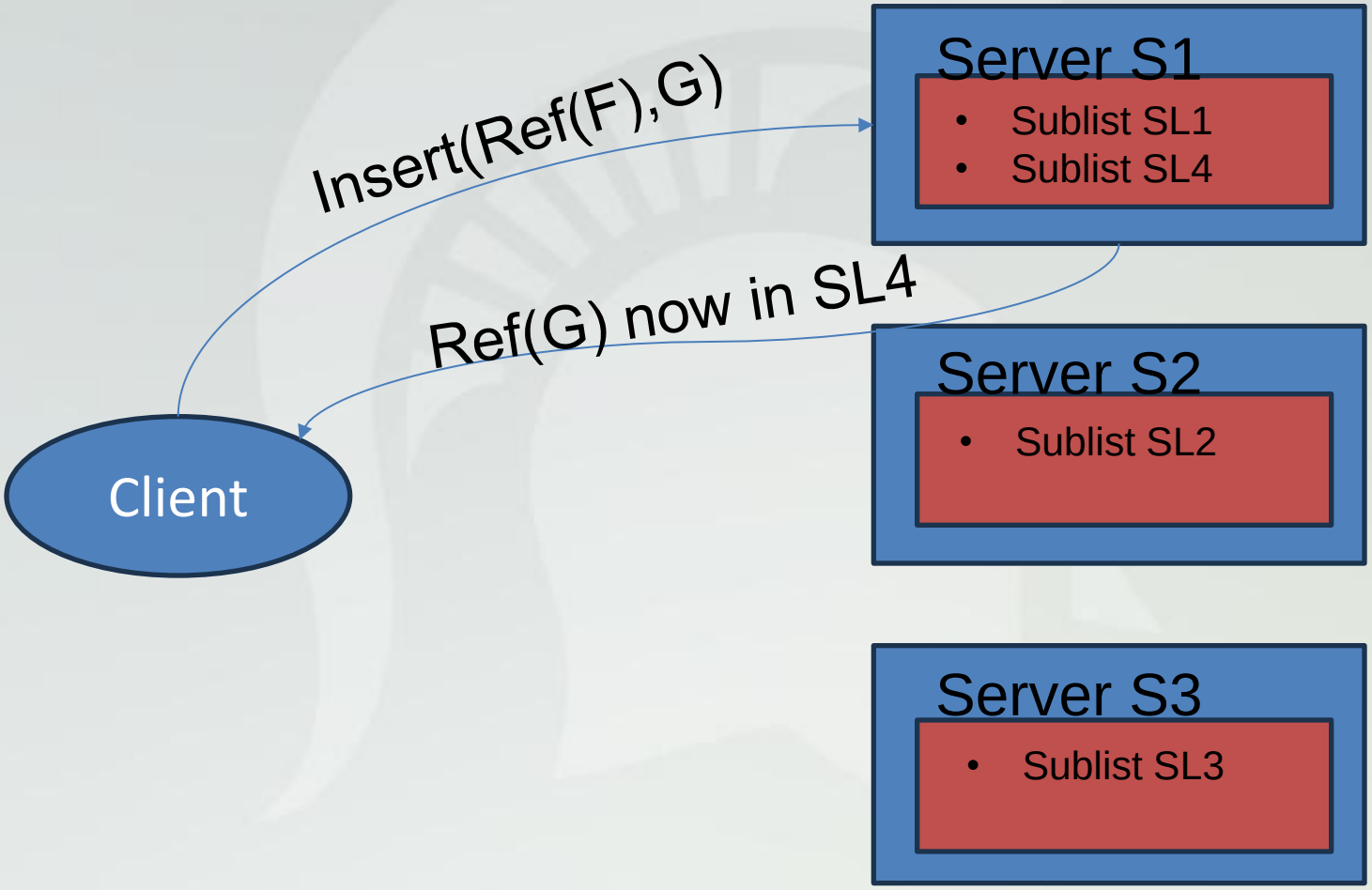
System Model

- Client-Server Model
- Client requests one or more machines for the following operations:
  - Lookup(key)
  - InsertAfter(prevNode, newKey)
  - Delete(node)
  - Next(node)
- **Servers are capable of asynchronously load balancing the list across the machines.**

Architecture for Lookup



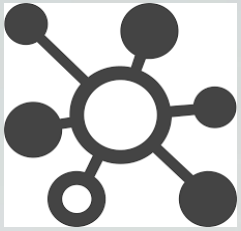
Architecture for InsertAfter, Delete, Next



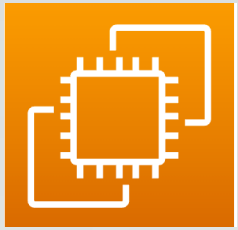
Salient claims that are evaluated in this fellowship project

- **Claim 1:** The asynchronous load balancer has very low overhead on the client operation throughput.
  - Observed by measuring the max throughput with and without load balancing.
- **Claim 2:** The implementation of the distributed list is horizontally scalable as the number of servers grow.
  - Observed by measuring the max throughput as the number of server machines of the same configuration is increased.

Main components for the Experiments



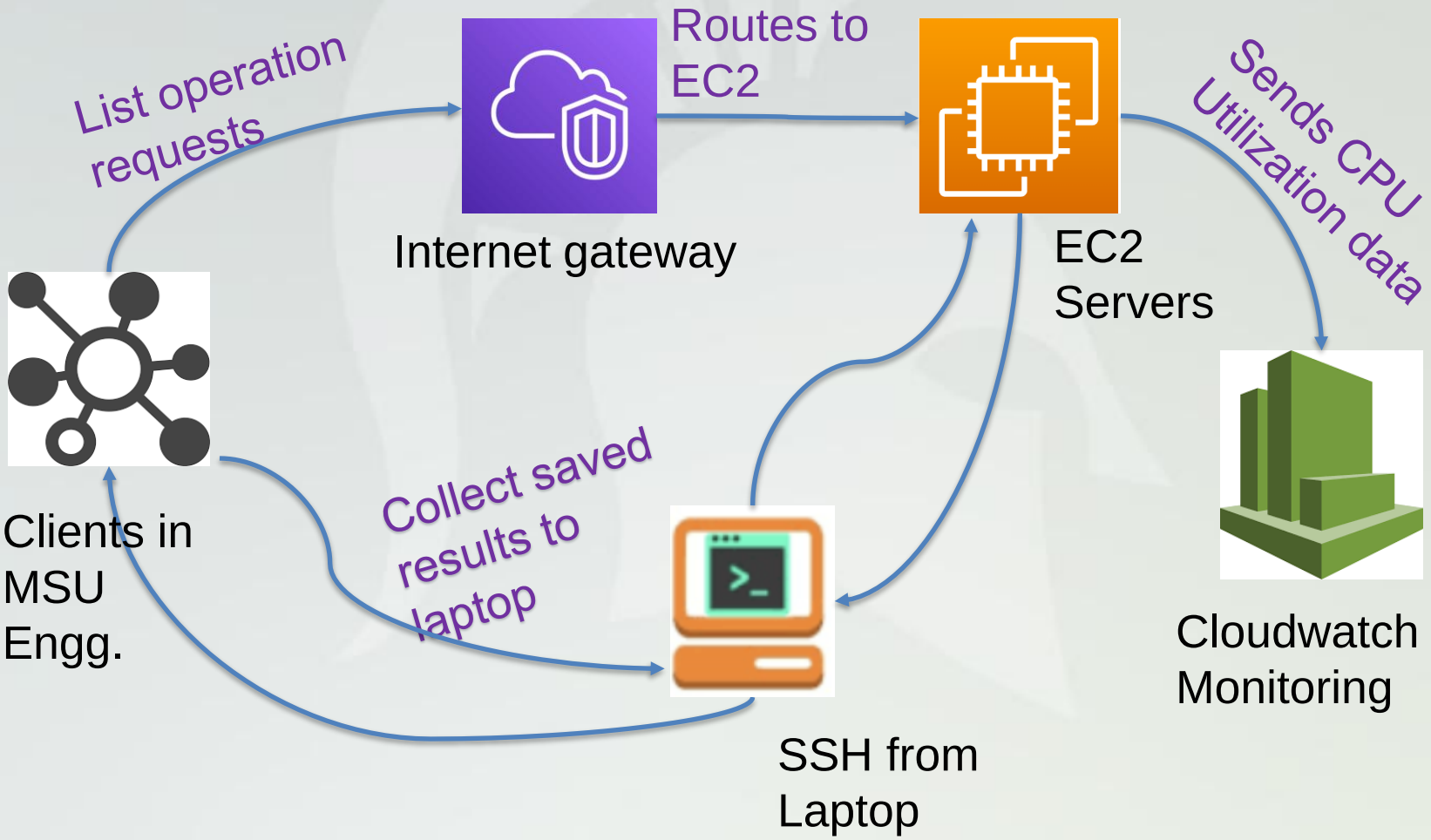
Computing Cluster from MSU Engineering resources are used to launch over as many as 1000 clients.



AWS EC2 instances are procured as the servers of the distributed linked list.

Amazon VPC is used to enable connection between the above two.

Overview of the Architecture



Rationale behind the decisions

- EC2 provides multiple machines of the same hardware, which helps measure performance scaling without worrying about difference in hardware.
- AWS network is very stable, giving consistent ping from MSU clusters to the VPC, hence providing reproducible, stable results.
- Cloudwatch helps monitor CPU utilization, which is essential to maximize in order to observe maximum throughput.

Distributed List Settings

- Executions of the same configuration is done with and without load balancing for comparison.
- Load balancing is performed too frequently, to measure the worst-case performance.
- Clients here always target operations on the sublist that is being moved between servers.

AWS Settings

- Number of servers is set to 2,4,6,8 during the experiments.
- Each server is a c7i.4xlarge compute optimized EC2 instance.
- Necessary ports are unblocked in the firewall through VPC for server-server communication.
- An internet gateway is enabled for client-server communication
- Cloudwatch is used to monitor CPU Utilization.

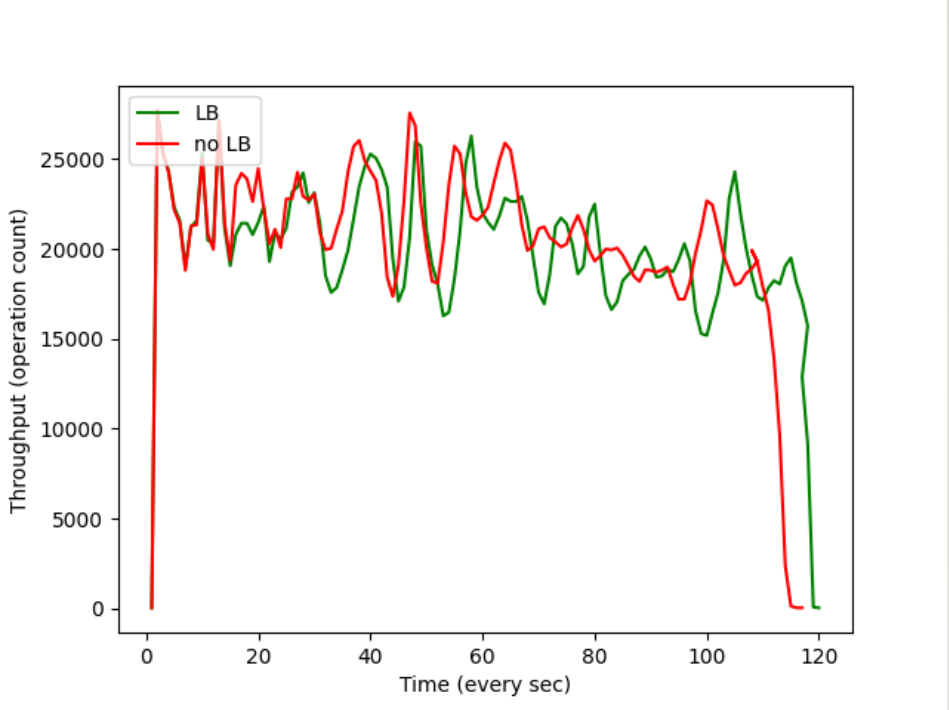
Challenges

- My server executables required a specific version of linux or higher. I had to switch to Ubuntu 24.04 for being able to run them.
- EC2 instances needed warmup iterations after booting them, before executing the experiments.
- My initial parameters for the distributed list surprisingly did not achieve maximum CPU utilization. I spent a few days fine tuning them.

Experiment 1: Low overhead

We observe similar variation in throughput and latencies overtime

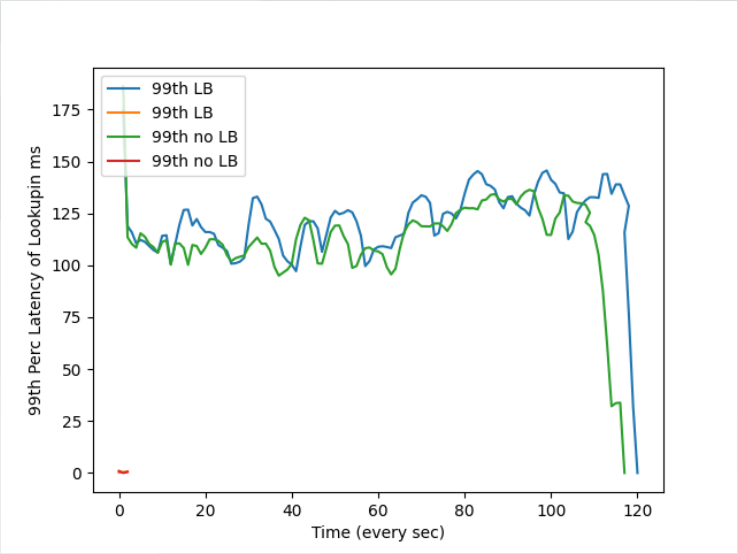
Avg. throughput every second

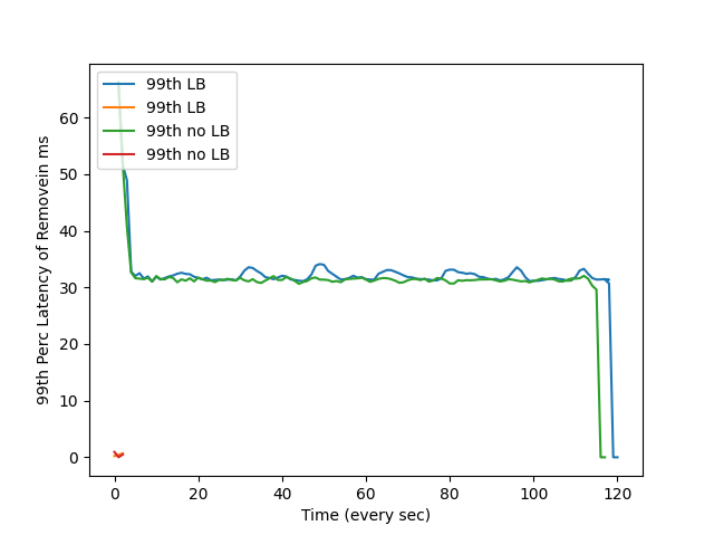


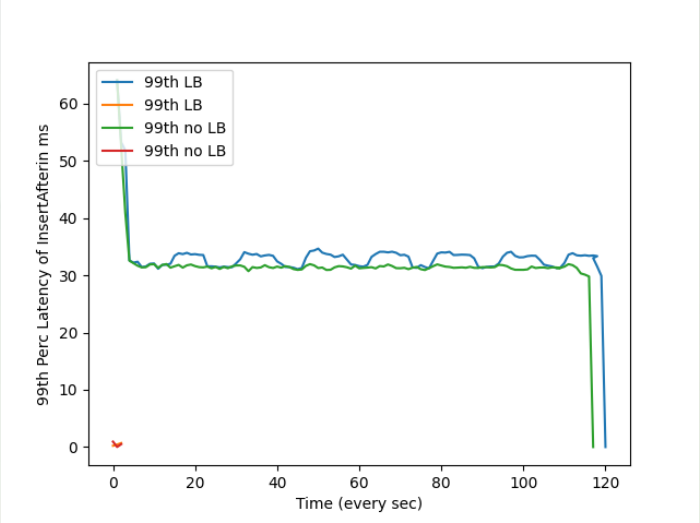
Experiment 1: Low overhead

P99 Latencies every second:

The small periodic spikes correspond to load balancing activity

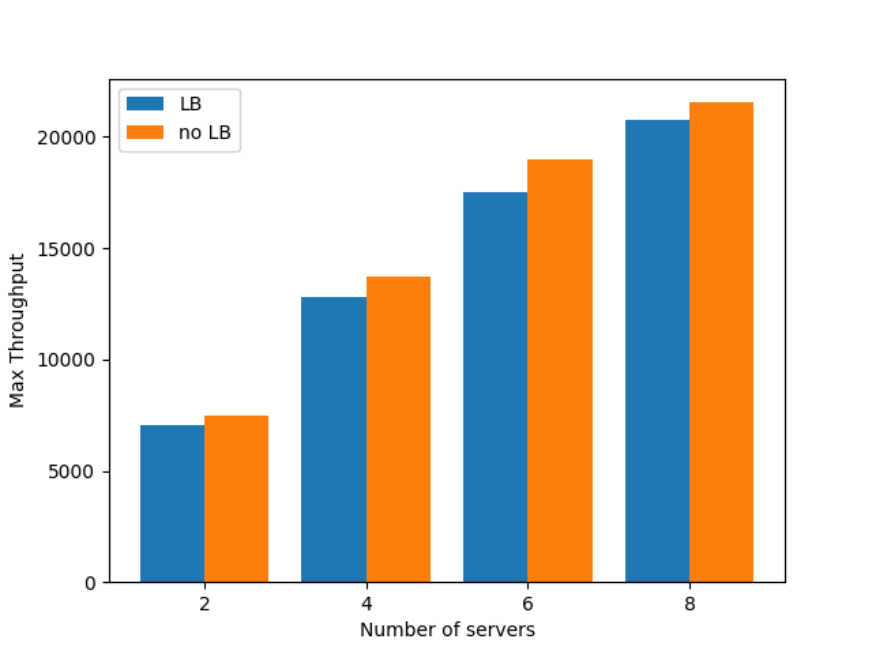






Experiment 2 : Horizontal scalability

- Number of clients is varied until maximum stable average throughput is observed.



Even in the worst case, the throughput loss < 10%

Success!

- **Claim1:** The asynchronous load balancer has very low overhead on the client operation throughput.
- **Claim 2:** The implementation of the distributed list is horizontally scalable as the number of servers grow.

Take-homes from the AWS Project

- While I had my own lessons to learn on Routing Tables and VPC, AWS has excellent documentation and community support for all configuration setups that I needed to do.
- I was able to reproduce my results on subsequent executions, proving stability of the results produced from AWS.

Future Interests

- Due to result reproducibility and ability to acquire various instance configurations on demand, cloud computing serves as an excellent tool for my research experiments.
- I have two other algorithms in the field of distributed that I will need to evaluate empirically, and having access to cloud resources would be invaluable.

Questions?