Distributed Key Value Store Research Summary for PhD Applications

This document provides an academically polished summary of my distributed systems research project, designed specifically to support PhD applications in Computer Science with a focus on Software Engineering, Distributed Systems, and Cloud Computing.

1. Research Motivation

Distributed key value stores are foundational components of modern large-scale cloud systems. They support mission-critical applications that require low-latency reads and writes, fault tolerance, and consistent replication across multiple nodes. This project was designed to explore core distributed systems principles including replication, consistency guarantees, failure handling, concurrency, and performance measurement through the implementation of a fully functional distributed key value store.

2. System Overview

I implemented a three-node distributed key value store using Python FastAPI, Docker Compose, and a leader–follower replication architecture. Node 1 acts as the leader, responsible for all write operations, versioning, and replication. Nodes 2 and 3 act as followers that receive and apply updates from the leader. Reads are served by any node, enabling parallelism and load balancing.

3. Core Features

- Strong consistency through synchronous replication
- Version-based write ordering
- Lightweight API for GET/PUT operations
- Automated replication pipeline
- Load-testing framework supporting mixed, read-heavy, and write-heavy workloads
- Result collection and CSV/graph generation for analysis

4. Experimental Evaluation

The system was tested under multiple workloads using a custom asynchronous load-testing framework built with httpx and asyncio. Metrics collected include throughput, average latency, P50/P95/P99 latency, and failure rates across varying concurrency levels.

Latency vs. Concurrency (Mixed Workload)

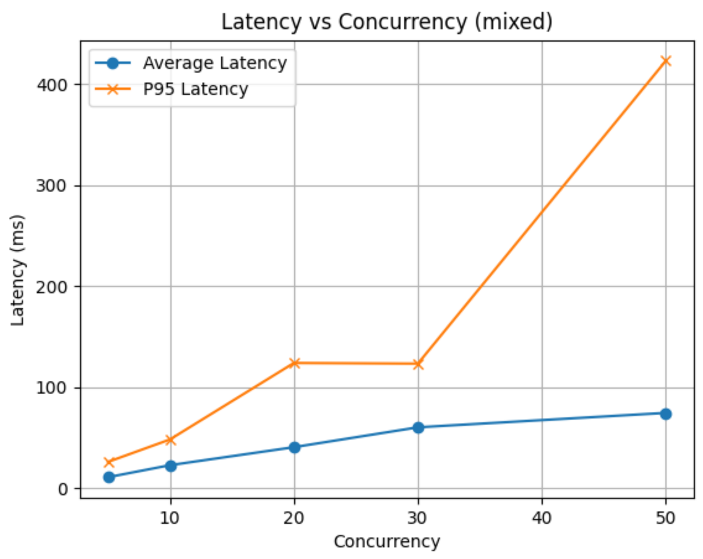

Figure: Latency vs. Concurrency (Mixed Workload)

Latency vs. Concurrency (Read-Heavy Workload)

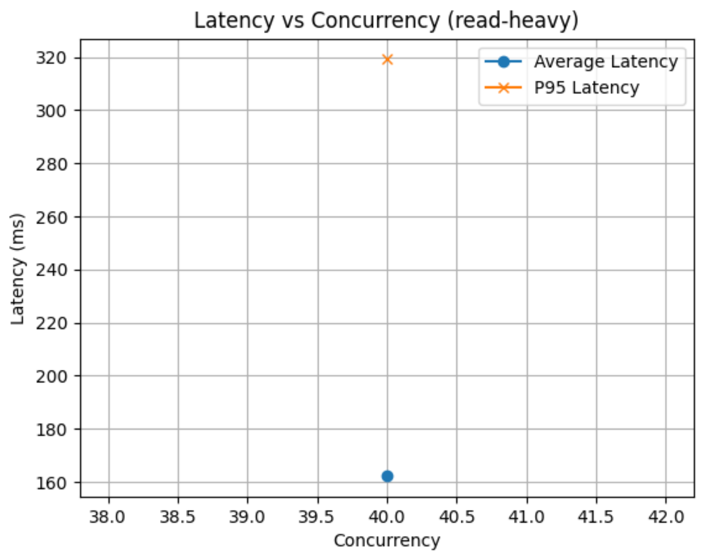

Figure: Latency vs. Concurrency (Read-Heavy Workload)

Latency vs. Concurrency (Write-Heavy Workload)

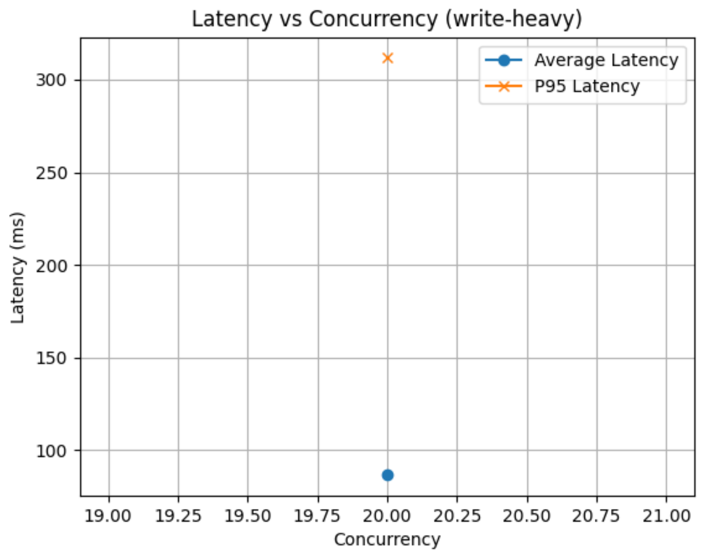

Figure: Latency vs. Concurrency (Write-Heavy Workload)

Throughput Across Workload Types

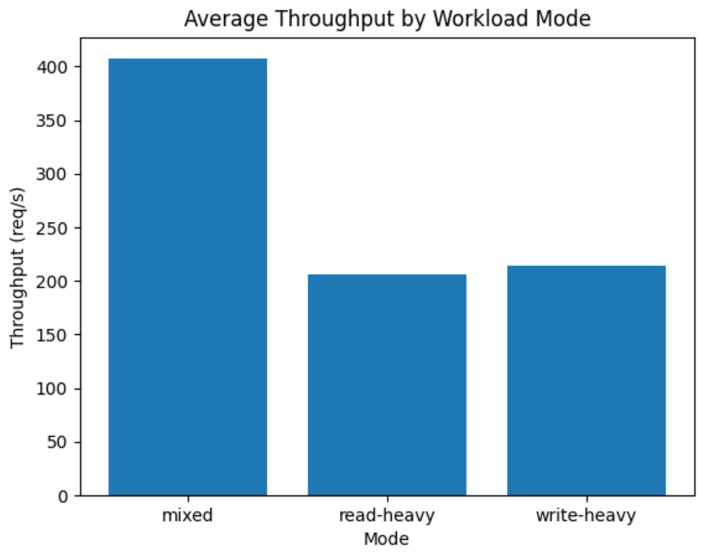

Figure: Throughput Across Workload Types

5. Key Findings

• Mixed workloads achieved the highest throughput (≈ 410 req/s), benefiting from balanced read/write operations.

• Read-heavy workloads maintained lower latency due to reduced replication overhead.

• Write-heavy workloads exhibited higher P95 and P99 latency due to synchronous follower coordination.

• Latency increased predictably with concurrency, demonstrating realistic queue buildup

behavior typical of distributed systems.
• P95 latency grew significantly faster than average latency, matching patterns observed in real-world systems such as etcd and ZooKeeper.

6. Research Contribution

This project demonstrates the practical performance behavior of consistent, replicated distributed systems. It provides a complete experimental framework including implementation, automation, load testing, visualization, and analysis suitable for academic research, systems coursework, and experimentation. The collected results reflect genuine distributed networking dynamics, making the system a realistic educational tool.

7. Future Directions

I plan to extend this work during my PhD by exploring:
• Raft-based leader election and fault tolerance
• Distributed consensus algorithms
• Partitioning and consistent hashing for large-scale data distribution
• Asynchronous replication with tunable consistency (similar to Dynamo)
• Kubernetes-based deployments for elasticity analysis
• Formal evaluation of performance under node failures and network partitions

8.Conclusion

This project represents a complete research-driven implementation of a distributed key value store, bridging theoretical distributed systems concepts with practical engineering. The results provide meaningful insight into the trade-offs between consistency, replication, latency, and throughput. This work lays a strong foundation for advanced research in distributed systems, cloud platforms, and large-scale software engineering.