An Evaluation of Distributed Concurrency Control

Paper authored by Rachael Harding et al.

Presented by Uzair Inamdar

What We'll Cover

1. Introduction
2. Protocols Being Tested
3. Test and Architectural Overview
4. Tests
    i. Contention
    ii. Update Rate
    iii. Multi-Partition Transactions
    iv. Scalability
    v. Network Speed
5. Points to Consider
6. Conclusion

Introduction

- Evaluation of performance of protocols across various tests.
- Rise of Distributed Database Management Systems (DBMS)
- Data Partitioning
- Serializability
- Deneva
    - Deployed on Amazon EC2 with 8 virtualized CPU cores and 32 GB of memory

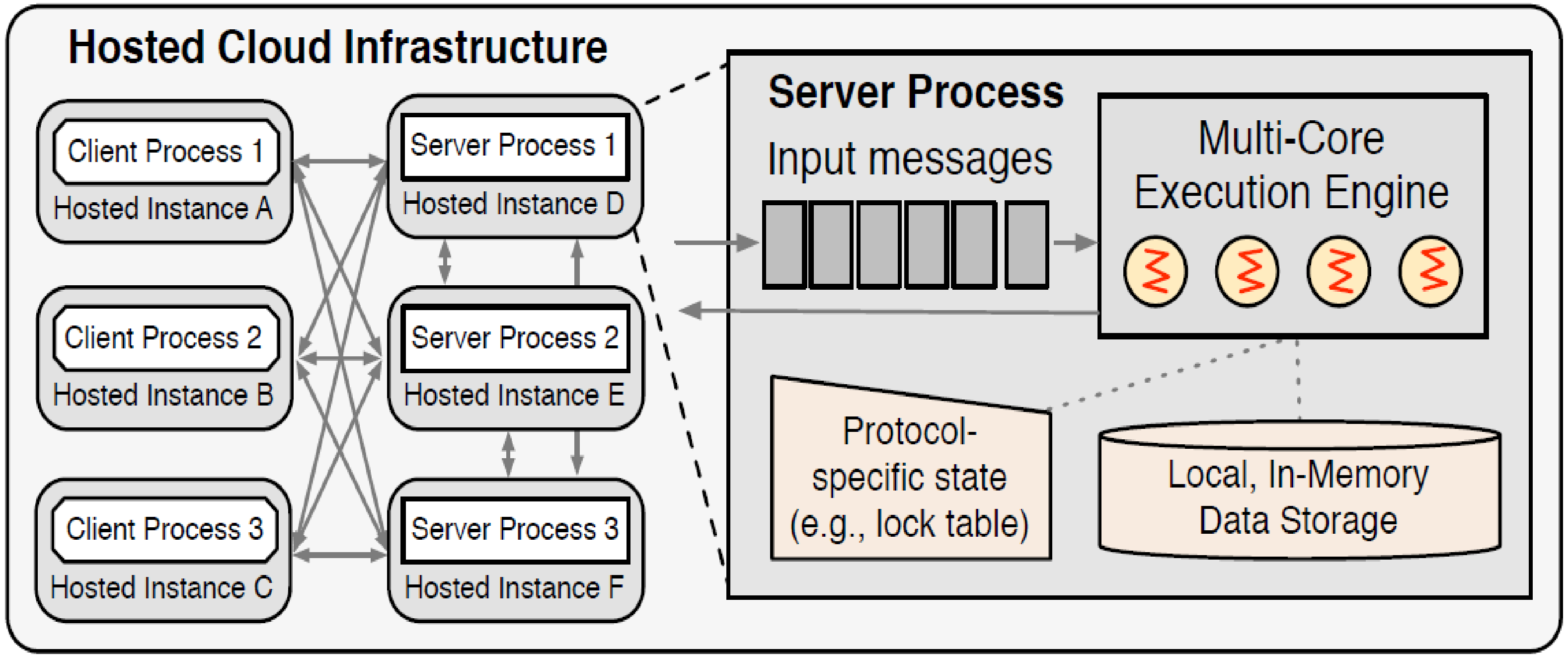
Protocols Being Tested

1. Two Phase Lock (2PL)
   a) NO_WAIT
   b) WAIT_DIE
2. Timestamp Ordering
   a) TIMESTAMP
   b) Multi-Version Concurrency Control (MVCC)
3. Optimistic Concurrency Control (OCC)
4. Deterministic (CALVIN)

Tests Overview

- Only serializable executions are analyzed.
- Online Transaction Processing are done through thread-safe sockets over TCP/IP.
- The queries are executed by 4 threads in a non-blocking manner unless a shared resource is being worked on.
- No logging, checkpoint, and recovery.
- Table partitions are preloaded on servers.
- Each server will carry 10,000 open client connection.
- When a transaction aborts, it restarts after a penalized period.

Architectural Overview



- Each server can host more than one partition, but each partition exists only on one server.
- Does not provide replication or fault tolerance.

TESTS

Contention

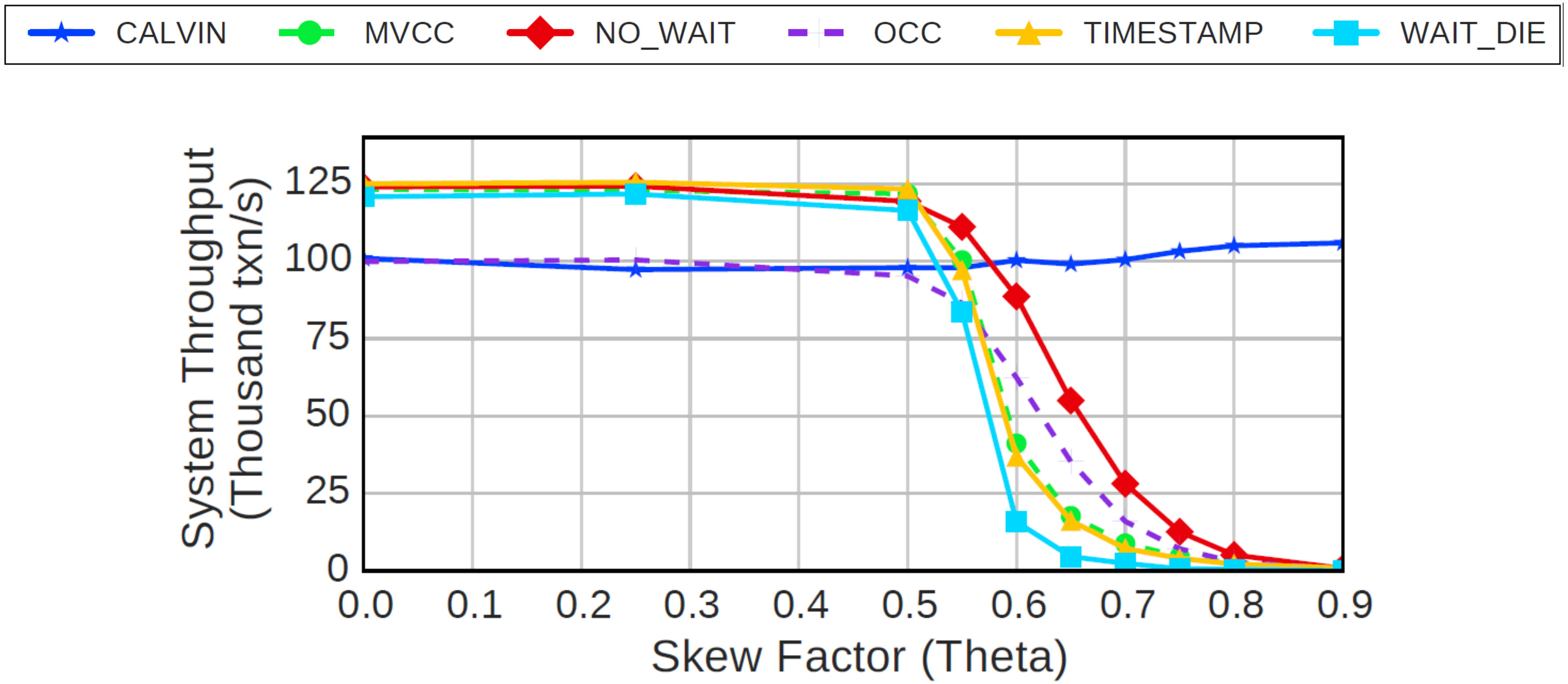


Figure 2: Contention – The measured throughput of the protocols on 16 servers when varying the skew factor in the YCSB workload.

1. Contention

- CALVIN is initially bottlenecked because of thee single threaded schedulers, but it is the only protocol to maintain good performance despite high skew.
- This is because locks are released much quicker once the read data is made available.
- OCC performs badly under low contention due to the overheads of copy and validation even when chances of modifications are the least.
- However, at higher levels the benefit of tolerating more conflicts and thus avoiding unnecessary aborts outweighs these overheads.
- The rest have a steep drop because of excessive wait times caused by data locks.

Update Rate

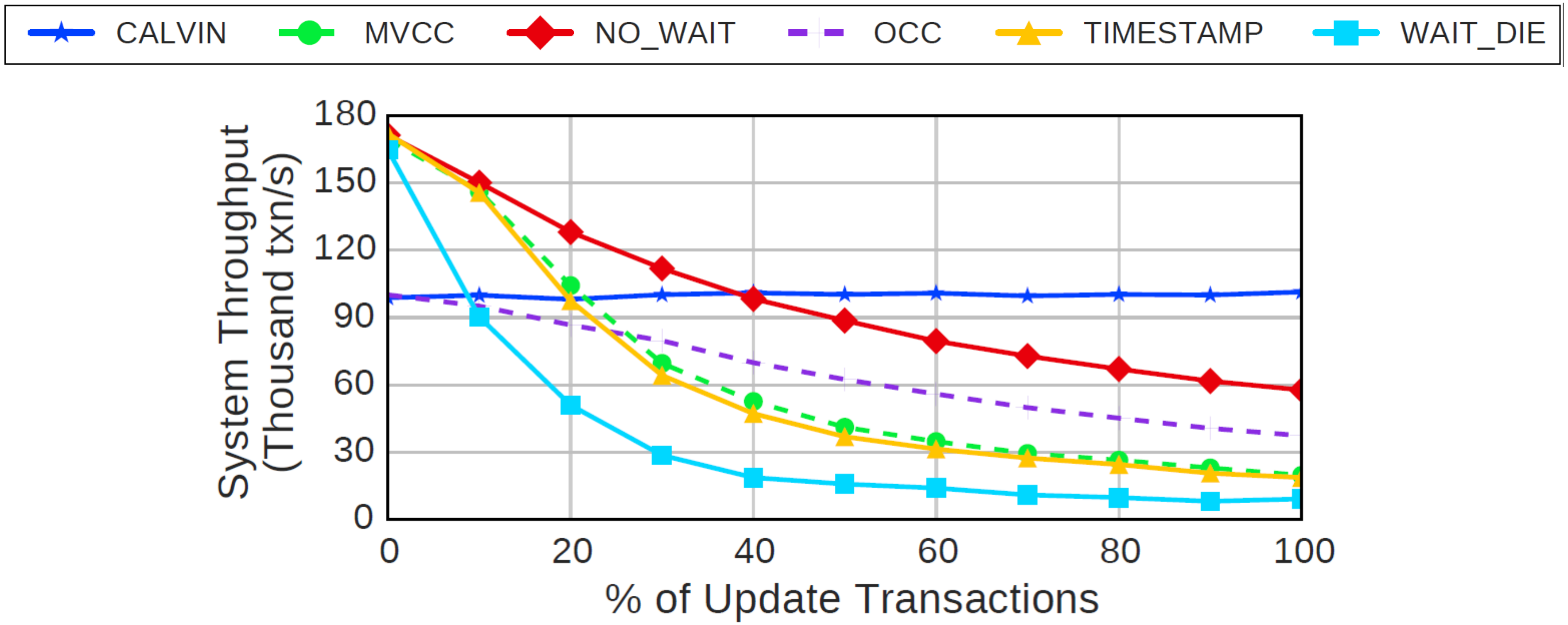


Figure 3: Update Rate – The measured throughput of the protocols on 16 servers when varying the number of update transactions (5 reads / 5 updates) versus read-only transactions (10 reads) in the workload mixture for YCSB with medium contention (*theta*=0.6).

2. Update Rate

- As the update% goes up, WAIT_DIE drops drastically and more transaction get queued into wait state as hot records are locked by various transactions.
- These transactions waste a lot of resource because they non-deterministically bypass the queues and prolong the natural flow of transactions. Even after all this waiting, they grow old and are aborted.
- NO_WAIT on the other hand is not affected because there is no waiting. The transactions abort straight away if the lock is not available.
- TIMESTAMP and MVCC suffer from the wait time caused by locks that are held by active transactions.
- OCC suffers from lower rates initially due to unnecessary validation checks, but at higher update% it is relatively faster as locks are not acquired.
- CALVIN's advantage and disadvantage from contention carry over to update rates.

Multi-Partition Transactions

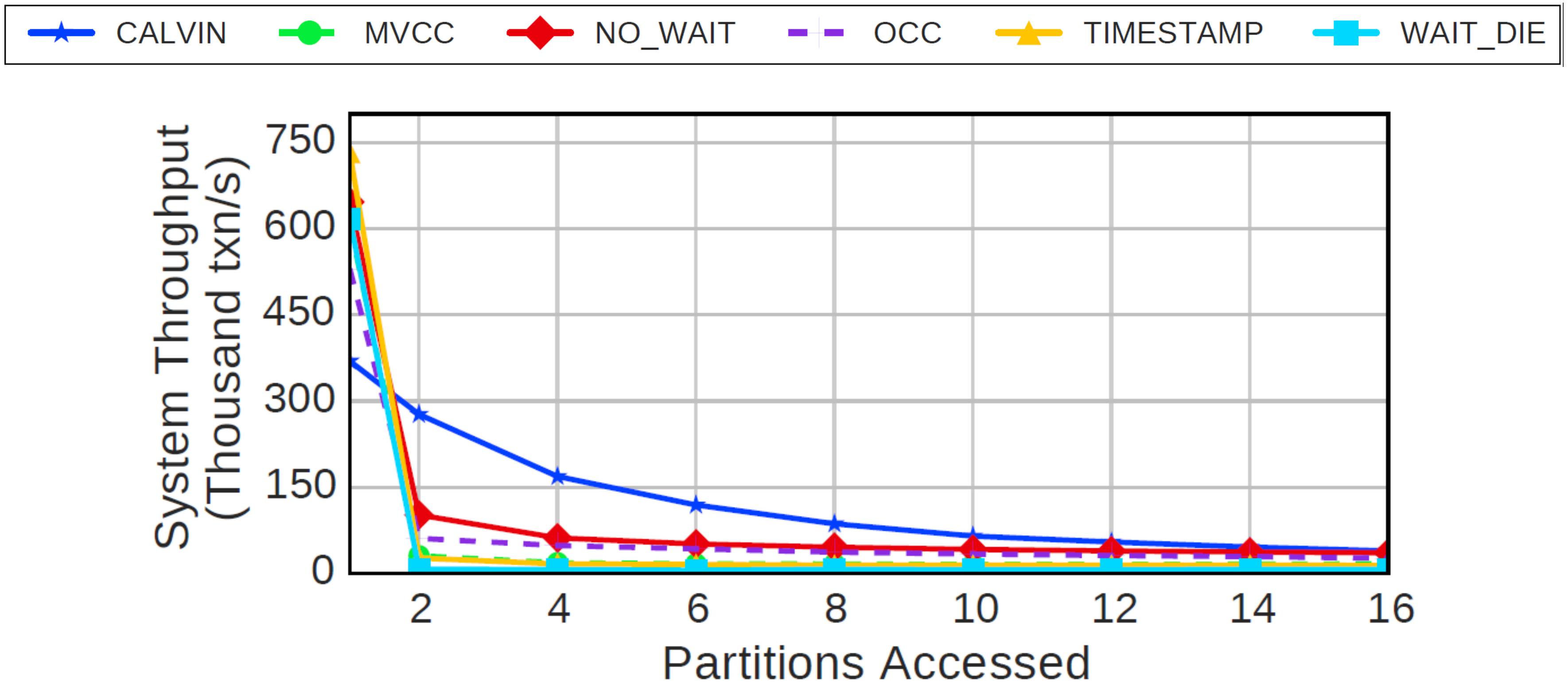


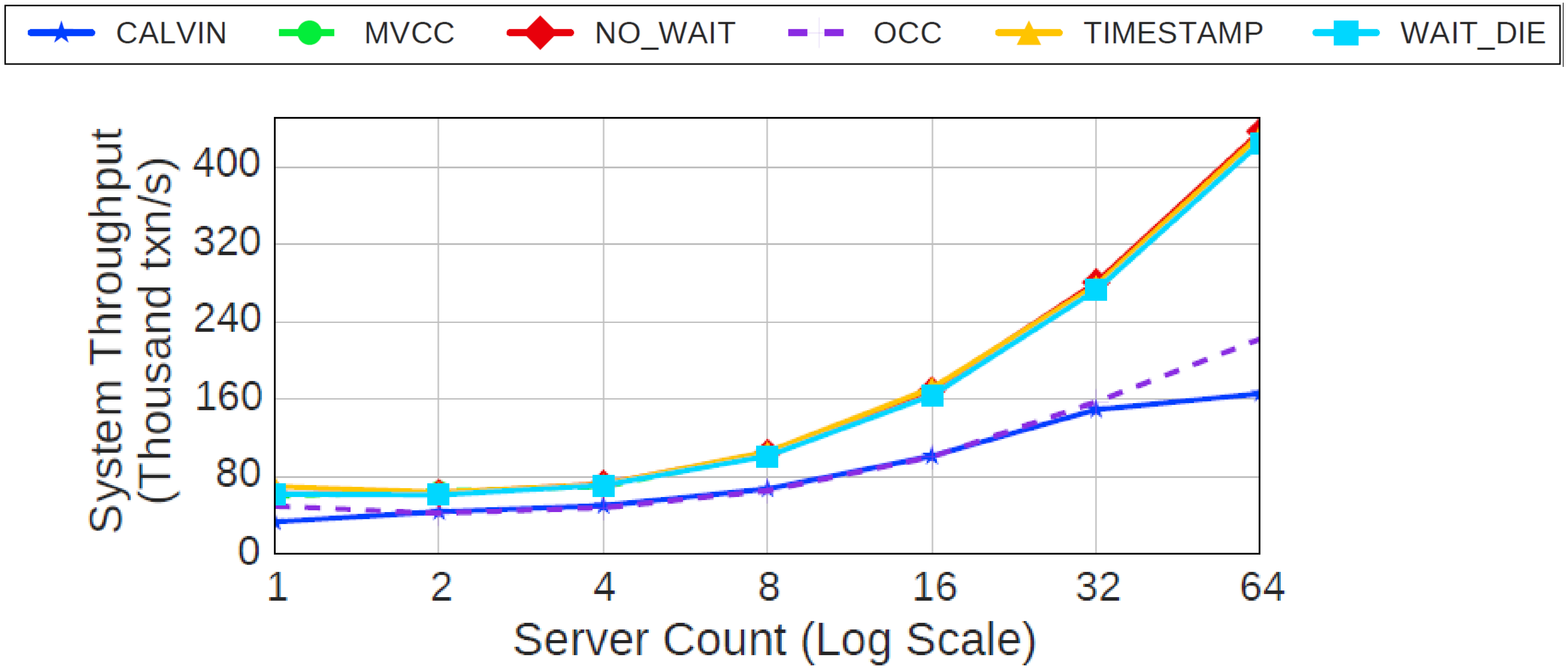
Figure 4: Multi-Partition Transactions – Throughput with a varying number of partitions accessed by each YCSB transaction.

3. Multi-Partition Transactions

- Pretty much all protocols plummet equally except CALVIN.
- Part of the reason is because CALVIN does not make use of 2 Phase Commit protocol (2PC).
- There is also the overhead caused by multiple requests and responses between different server messages.
- CALVIN on the other hand synchronizes its schedulers every 5ms and hence the transactions are automatically forwarded to the target partitions.

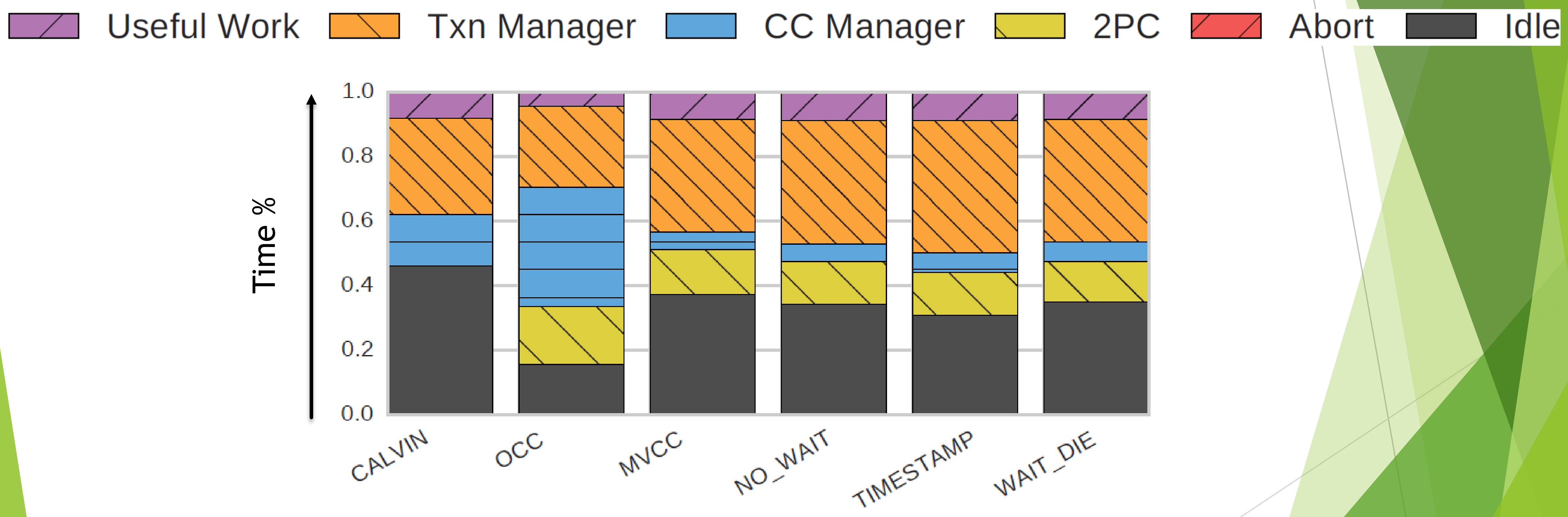
4. Scalability

A) Read-only Workload



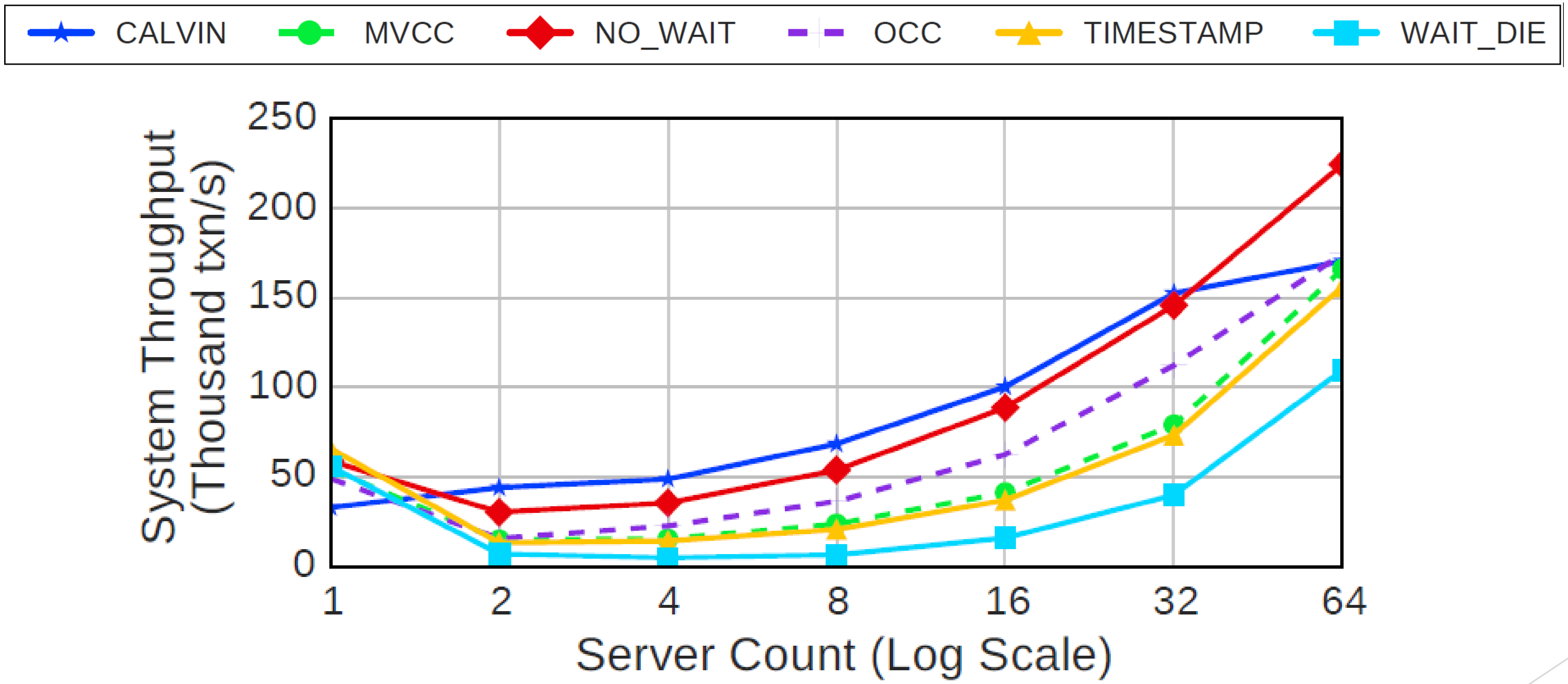
(a) Read-Only (No Contention)

Time % Graph



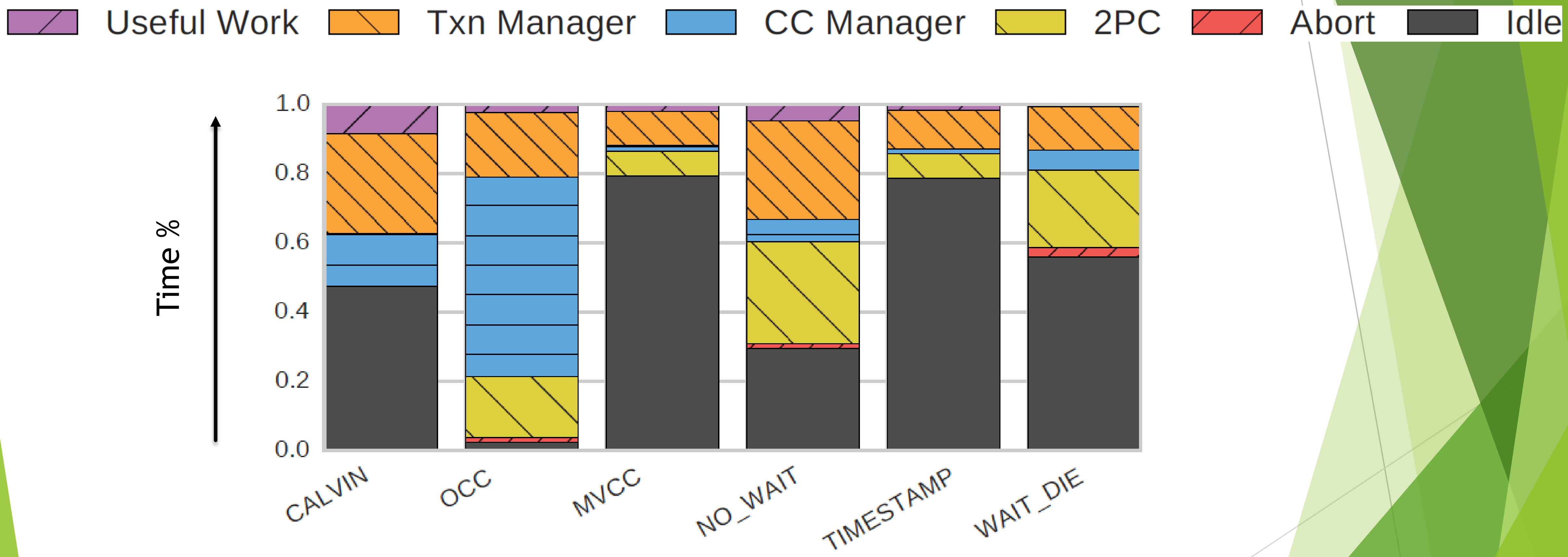
(a) Read-Only (No Contention)

B) Read-Write (Medium Contention)



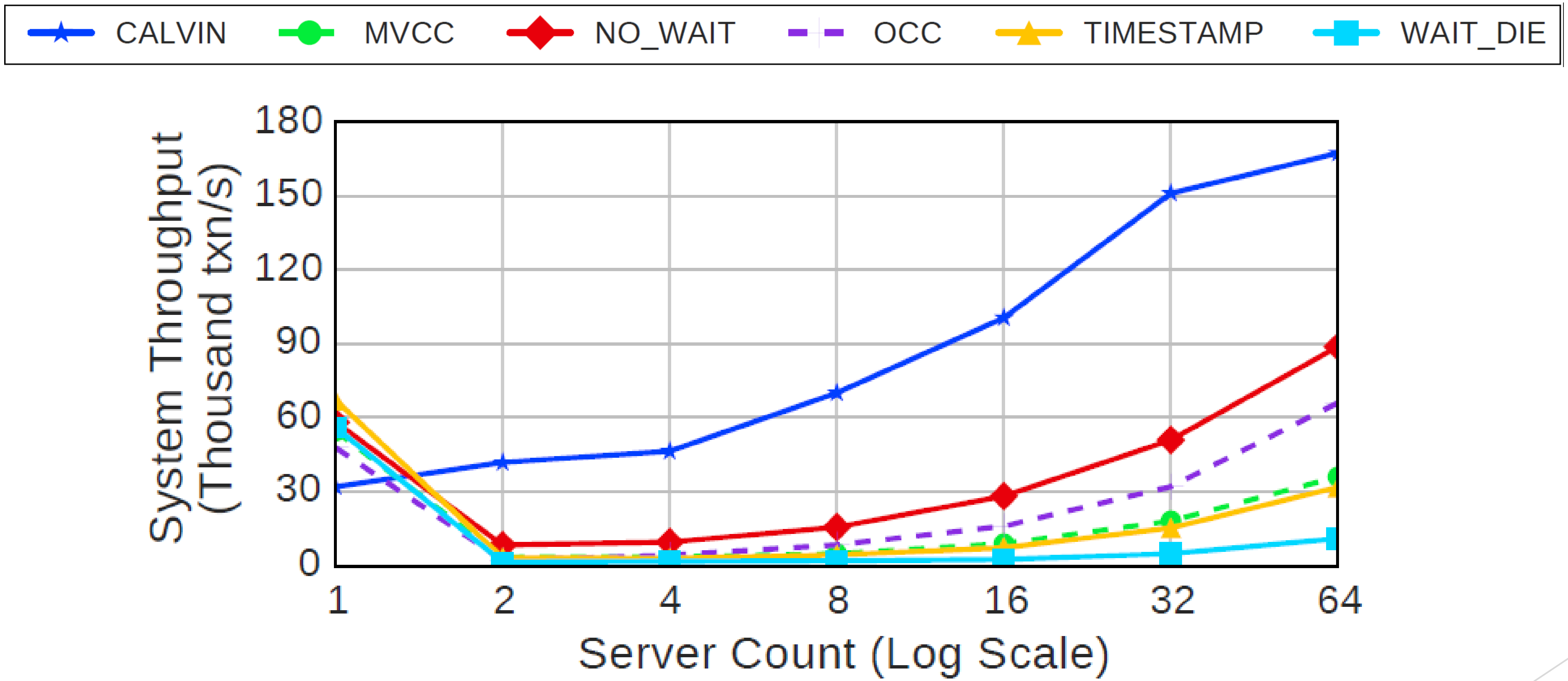
(b) Read-Write (Medium Contention)

Time % Graph



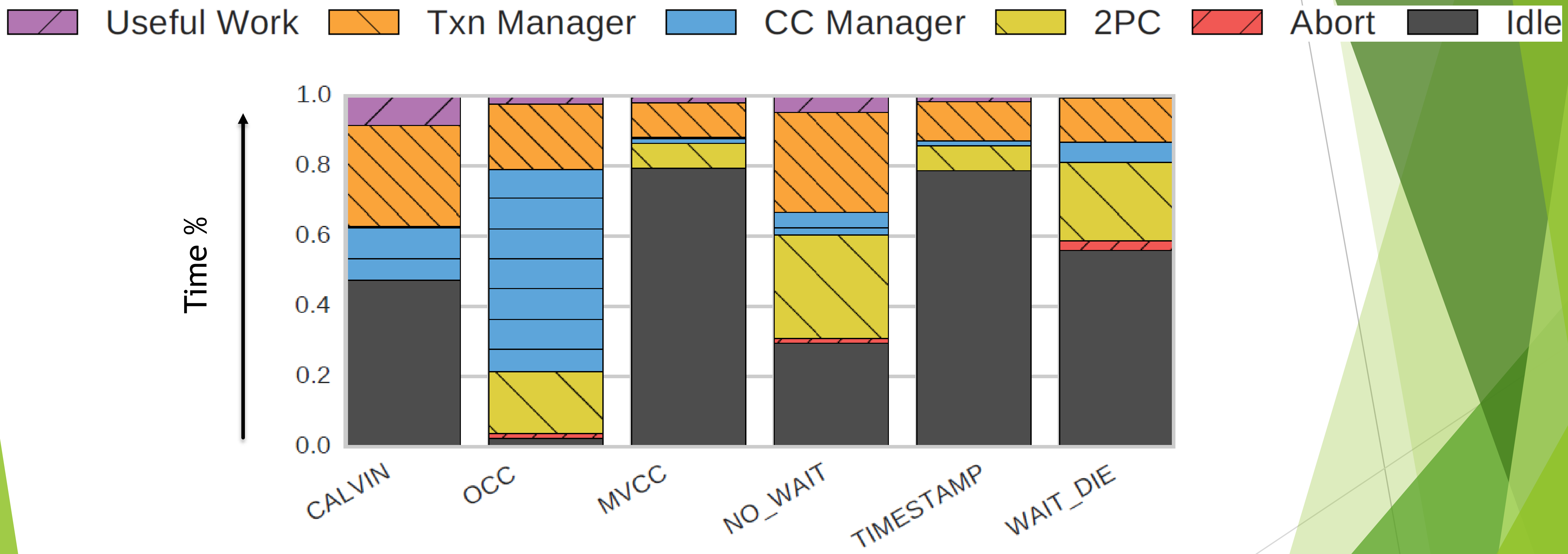
(b) Read-Write (Medium Contention)

C) Read-Write (High Contention)



(c) Read-Write (High contention)

Time % Graph



(b) Read-Write (Medium Contention)

5. Network Speed

- Protocols that use 2PC strategy experienced the most loss in throughput as the commits required delivery of transactional messages.
- CALVIN does not need to exchange any messages between servers between its read and write phase. Hence, it performs the best.
- WAIT_DIE sees the worst performance as not only are the messages being lagged, the lagging is causing a lot of the waiting transactions to age and abort. The high abort rate in the main reason for the significant loss in performance.
- TIMESTAMP and MVCC also suffer loss in performance because of delay in transactional messages, but don't suffer as bad a rate for aborts.

Network Speed

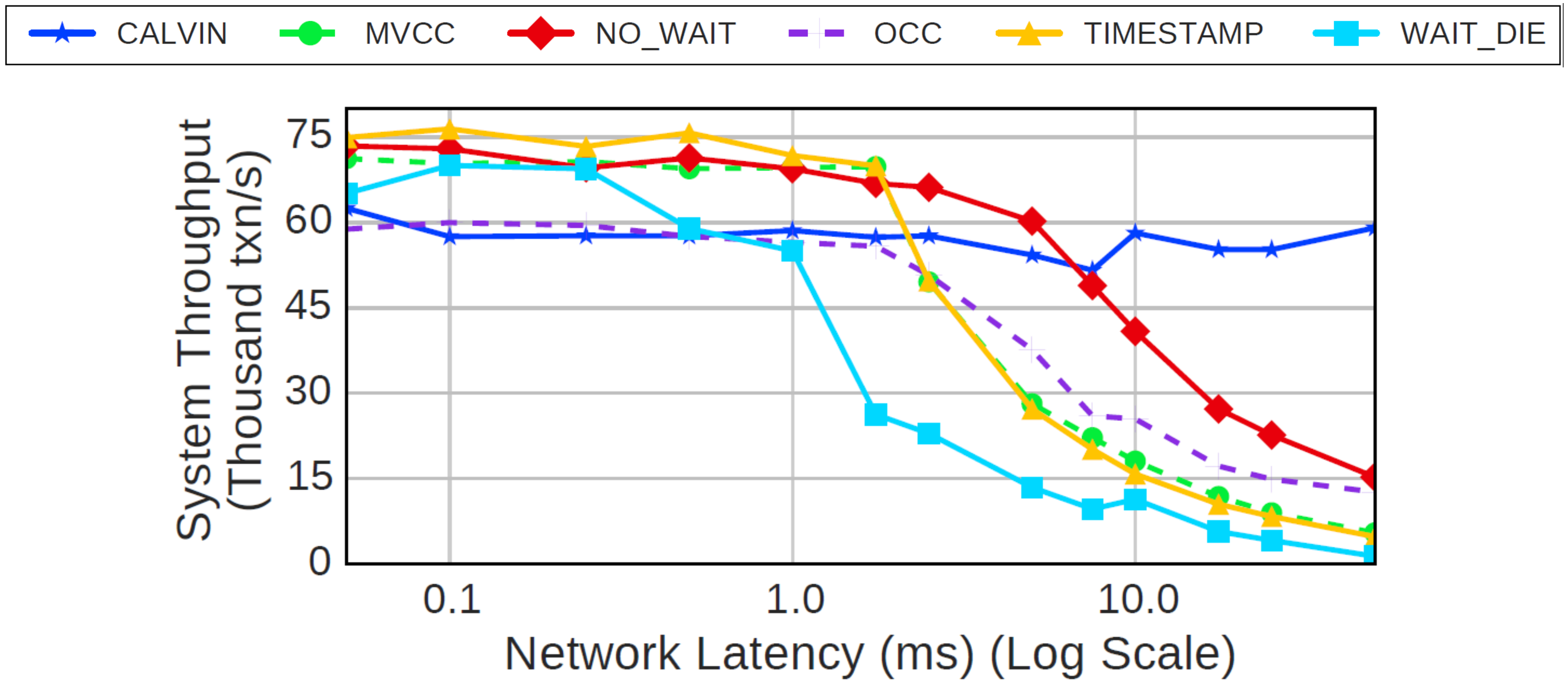


Figure 9: Network Speed – The sustained throughput measured for the concurrency protocols for YCSB with artificial network delays.

Points to Consider

- Effect of platform constraints on protocols
  - Number of requests allowed
  - The back-off penalty
  - Number of Servers/Partitions
  - Type of Transactions
- Testing on Transaction Processing Performance Council - Type C (TPC-C)

Conclusion

- 2PL performs poorly under high contention due to aborts.
- Timestamp-ordered concurrency control does not perform well under high contention due to buffering.
- Optimistic concurrency control has validation overhead.
- Deterministic protocol maintains performance across a range of adverse load and data skew but has limited performance due to transaction scheduling.
- There exists a serious scalability problem, especially when the partitions do not exist in a single data center.
- Possible Solution

Thank You