

Distributed Database with YugabyteDB and CockroachDB

**Advanced Database
INFO-H-415**

Sony Shrestha

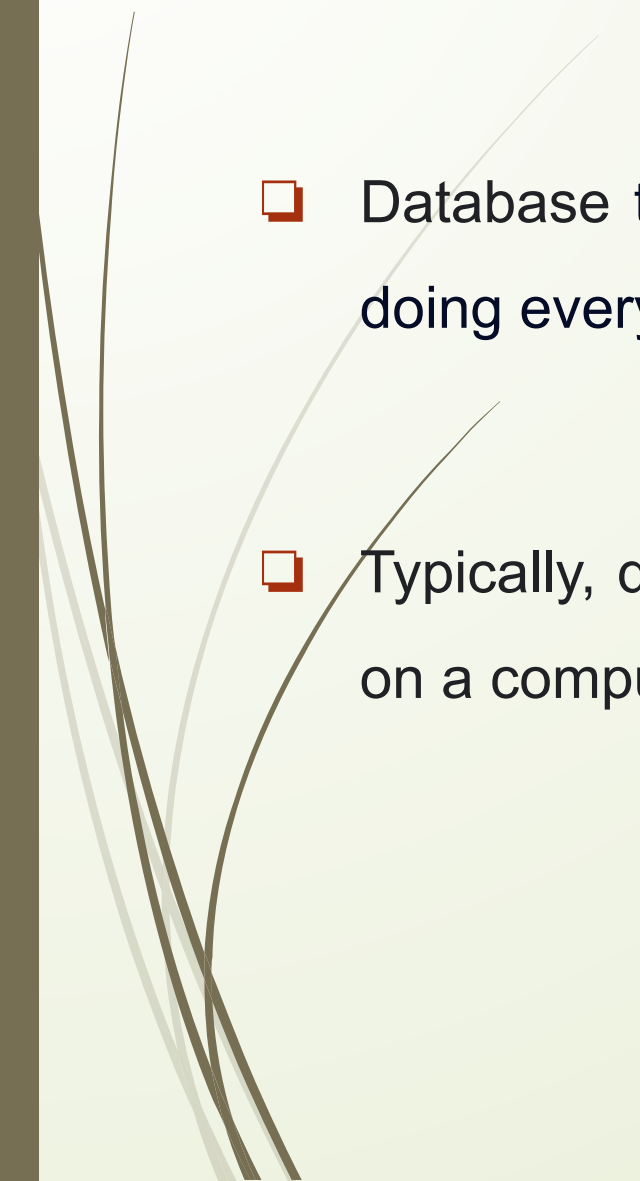
Aayush Paudel

Shofiyyah Nadhiroh

MD Kamrul Islam



Distributed Database

- ❑ Database that runs and stores data across multiple computers, as opposed to doing everything on a single machine.
 - ❑ Typically, distributed databases operate on two or more interconnected servers on a computer network.
- 



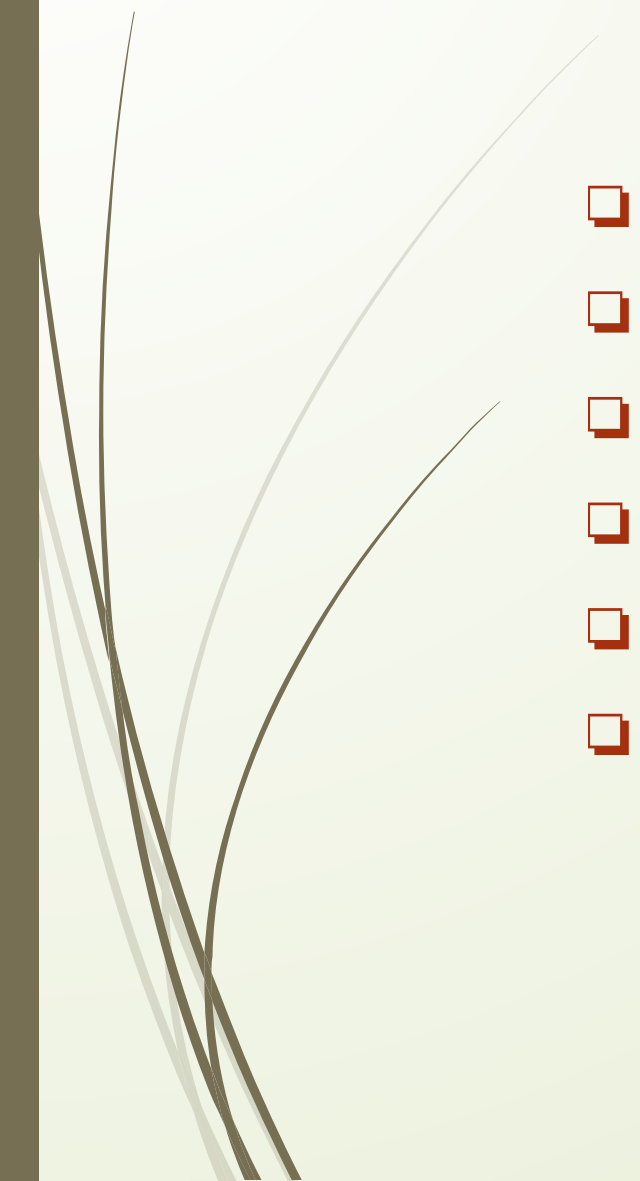
Features of Distributed Database



- ❑ Data partitioning
- ❑ Data replication
- ❑ Distributed query processing
- ❑ Distributed transaction processing
- ❑ Consensus protocols
- ❑ Distributed locking mechanisms



Advantages of Distributed Database

- 
- ❑ Scalability
 - ❑ Availability
 - ❑ Fault Tolerance
 - ❑ Performance
 - ❑ Replication
 - ❑ Distributed Query Processing



yugabyteDB

YugabyteDB

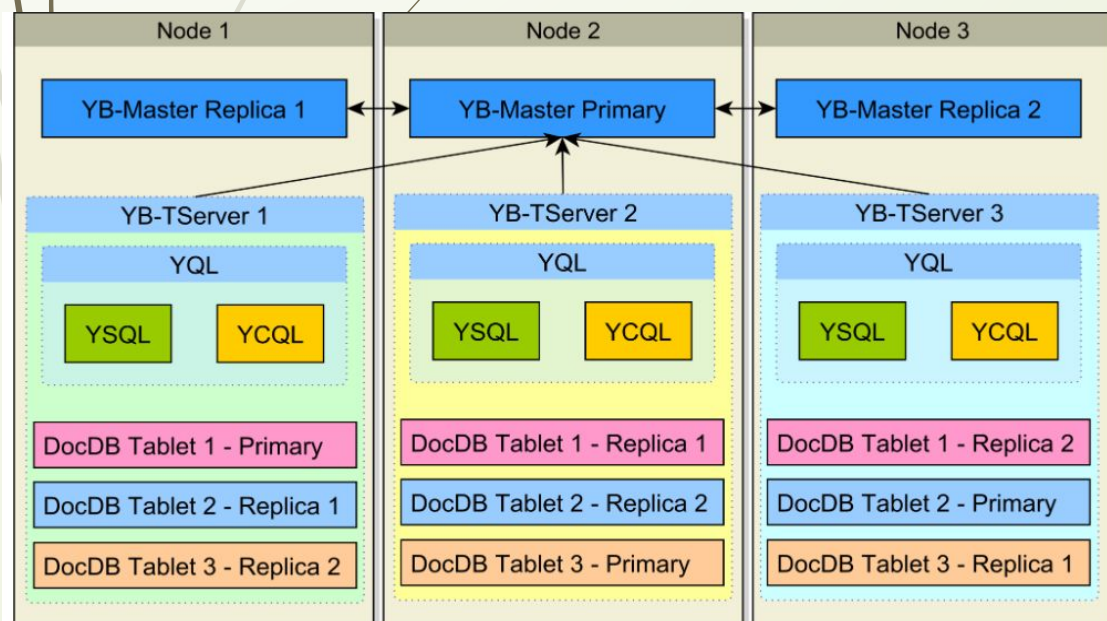
- ❑ Distributed SQL Database - SQL interface and transactional capabilities
RDBMS + scalability and resilience of NoSQL databases
- ❑ PostgreSQL compatible
- ❑ Open source
- ❑ Can run across hybrid and multi-cloud environments
- ❑ With automatic sharding, distribution and load balancing, it is possible to scale out and scale back in with zero disruption in minutes
- ❑ Suitable for mission critical applications (Online Banking, Patient Management, Billing Systems, Air Traffic Control)

YugabyteDB

- ❑ 2016: Former Facebook engineers (scaling the Facebook user database)
- ❑ Vision: open-source, high-performance, distributed SQL database
- ❑ 2019: Open Sourced under the Apache 2.0 license.
- ❑ 2020: Yugabyte Cloud, a fully managed service for YugabyteDB.
- ❑ 2021: YSQL API for PostgreSQL compatibility

Architecture (Universe)

- Inspired by Google Spanner and Facebook's RocksDB
- Auto-sharding(default) , so data is scattered (tablets) across multiple nodes
- In order to allow all database nodes to accept writes, one node becomes the Leader (Primary) of a given table record while the other nodes become Followers (Replicas) of that table row



YB-Master service

- Metadata + Mapping (table rows and the nodes that store the Leader Tablet and the Replica Tablets that store the entry for that particular table row)
- One YB-Master Leader (Raft Consensus), two YB-Master Followers
- Fourth Node, cache the metadata locally
- Load Balancing, Table Creation and Schema Changes, Background Ops, Metadata Query Response

YB-TServer

- Each database node stores a YB-TServer service - query processing, replication and storage
- Components:
 - the YQL query engine
 - the DocDB storage engine - ACID

Architecture



YQL Query Engine

The YQL query engine provides two APIs

- YSQL – SQL query engine based on PostgreSQL
- YCQL – a semi-relational API based on Cassandra Query Language

DocDB Storage Engine

- Distributed document storage called DocDB (document-oriented fork of RocksDB) which stores the table records as documents in a LSM (Log Structured Merge) tree structure. (Write Optimized) [MemTable and SortedStringTables], Flushes and Compaction

Write Operation

1. Query Language Processing

2. Forwarding to the tablet leader

3. Preparing the batch for replication

4. Raft Replication

5. Applying update to storage engine on the leader

6. Responding to the client

5. Applying update to storage engine on followers



CockroachDB

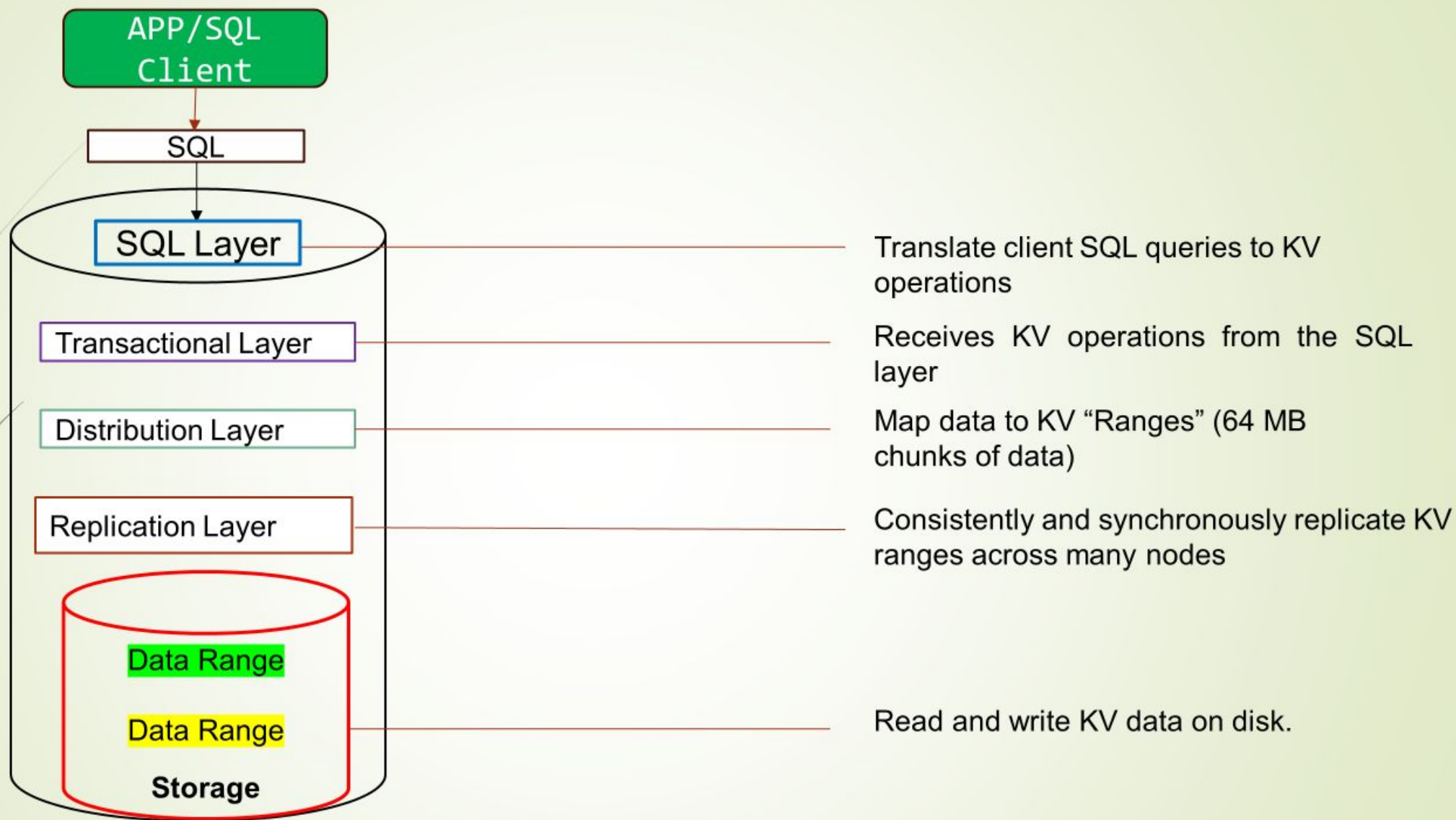
CockroachDB

- ❑ CockroachDB is an open-source distributed relational database system
- ❑ Designed for high scalability and resilience
- ❑ Ensure strong consistency and ACID guarantees by using distributed transaction
- ❑ Named after the cockroach's resilience to failure, reflecting the system's ability to withstand disruptions.

Features of CockroachDB

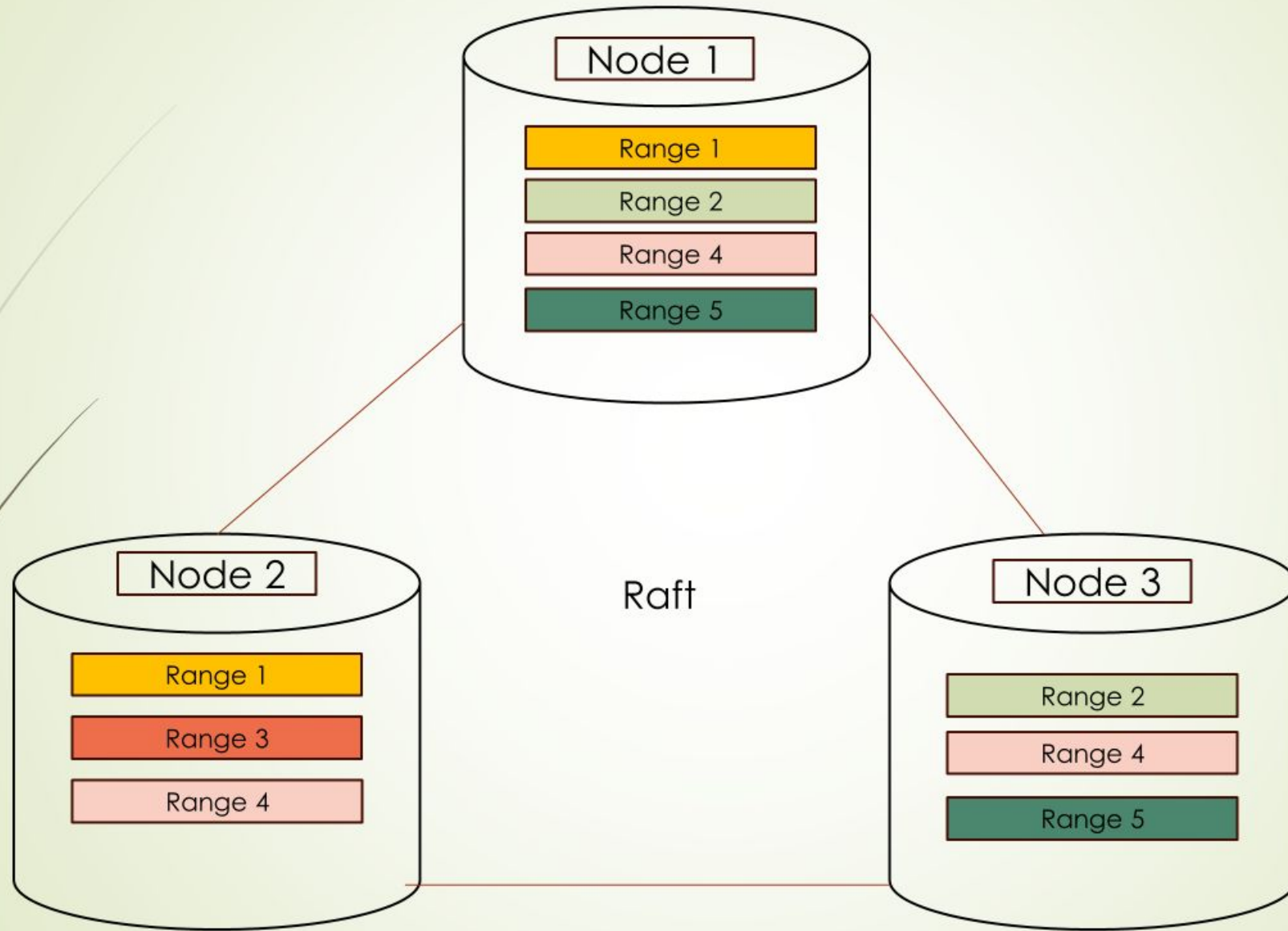
- ❑ Serializable Isolation
- ❑ Geo-replication
- ❑ Multi-cloud
- ❑ Advanced admin tool

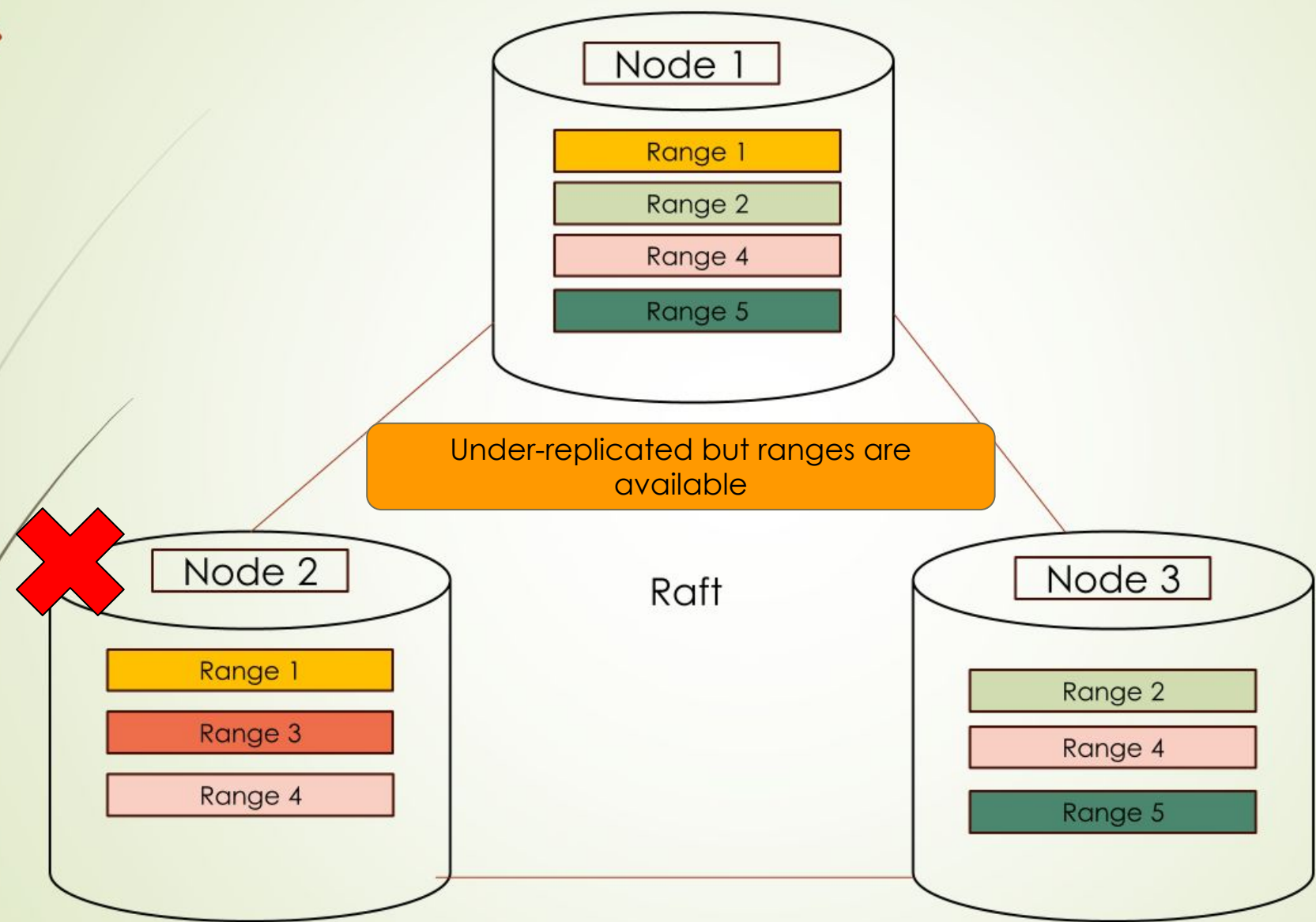
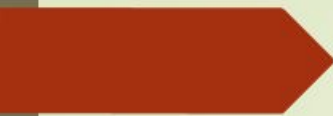
CockroachDB Layers

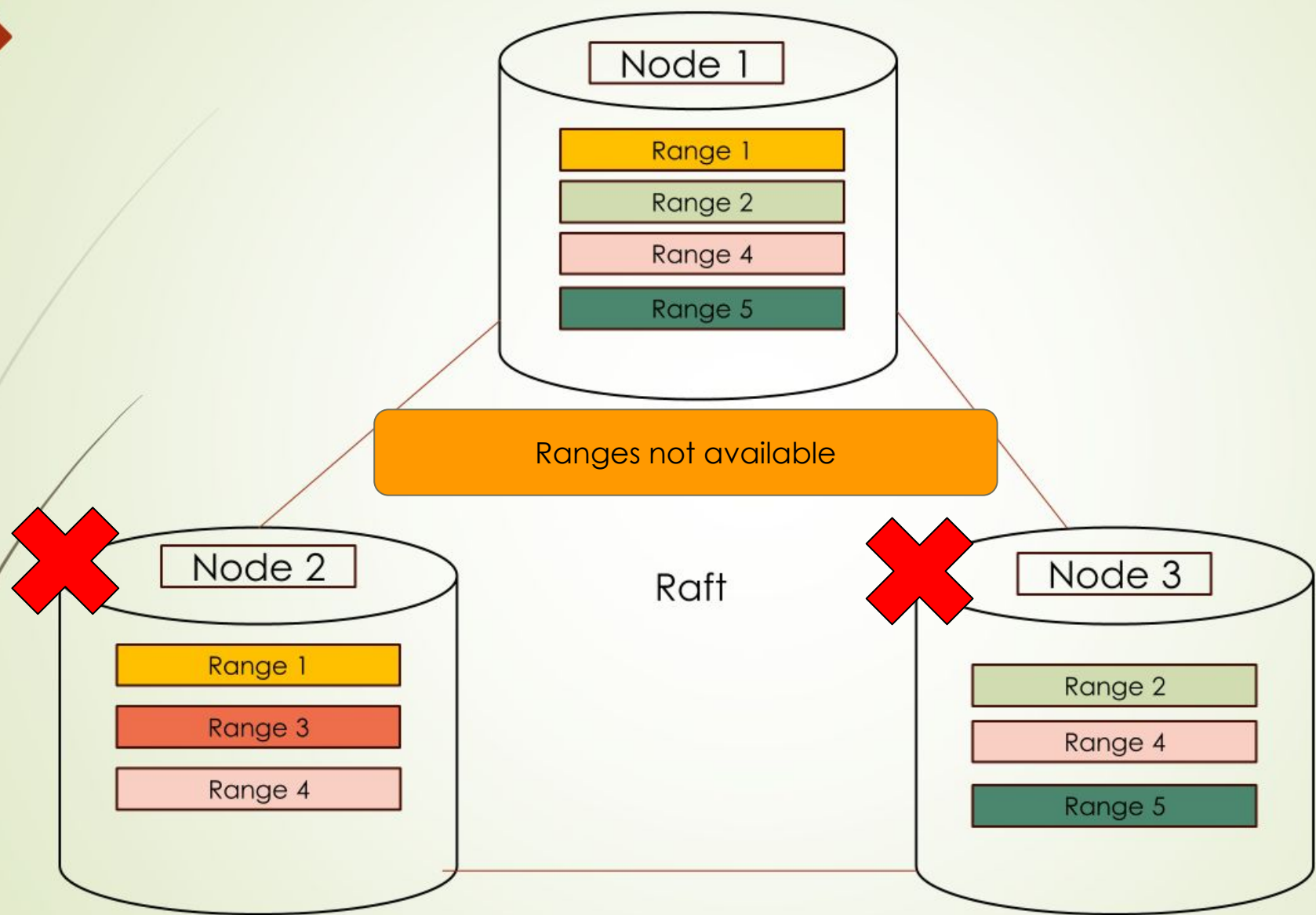


Cluster Architecture

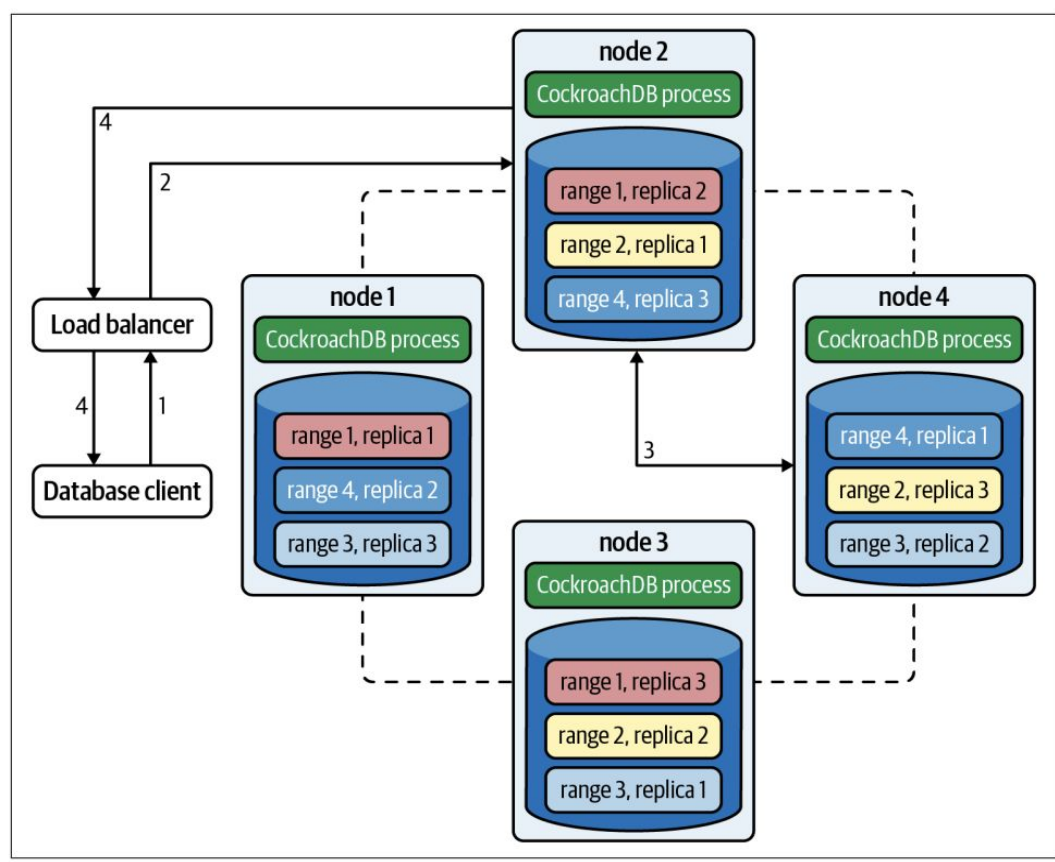
- ❑ CockroachDB requires a “quorum”
- ❑ It uses Raft Consensus protocol
- ❑ It distributes data among nodes
- ❑ Replicates data into multiple nodes
- ❑ Fault tolerance
 - ❑ It can survive = $(\text{Replication factor} - 1) / \text{number of node failures}$







Key cluster Architecture



- Shared-nothing architecture ensures high availability and scalability
- Load balancers distribute connections for improved performance
- Leaseholder nodes manage data access and replication for consistency
- Key range-based data distribution enhances fault tolerance



Yahoo Cloud Serving Benchmark (YCSB)

Benchmarking with YCSB



Workload A	Update heavy workload	50 % read 50% write (not necessarily the same records)
Workload B	Read mostly workload	95 % read 5 % write (not necessarily the same records)
Workload C	Read only	100% Read
Workload D	Read latest workload	5% insert 95 % read
Workload E	Short ranges	95 % scan 5 % insert
Workload F	Read-modify-write	Forces read of the records to be modified and updated

- ❑ Two phases
 - ❑ Load Phase
 - ❑ Run Phase
- ❑ Can parameterize
 - ❑ Record Count
 - ❑ Operation Count
- ❑ Benchmark Result
 - ❑ Runtime
 - ❑ Throughput [ops/sec]
 - ❑ Latency

Implementation

1. Installation of YugabyteDB and CockroachDB



2. Multi-node setup

3. Installation of YCSB

4. Configure db properties

5. Configure workload

6. Benchmarking

```

for wl in worloads:
    for rec in record_counts:
        for op in operation_counts:
            for lr in load_or_run:
                if lr=='load':
                    connection = connect_to_database(db_name)
                    if connection:
                        try:
                            truncate_table(connection,table_name)
                        except psycopg2.Error as err:
                            print('connection failed')
                        finally:
                            connection.close()

                if db_name=='Y':
                    command = """./bin/ycsb.sh {load_or_run} jdbc
                        -s -P ./yugabyte-binding/yugabyte.properties
                        -P workloads/workload{workload_name} -p recordcount={recordcount_name}
                        -p operationcount={operationcount_name}
                    """.format(
                        workload_name=wl,
                        recordcount_name=rec,
                        operationcount_name=op,
                        load_or_run=lr
                    )

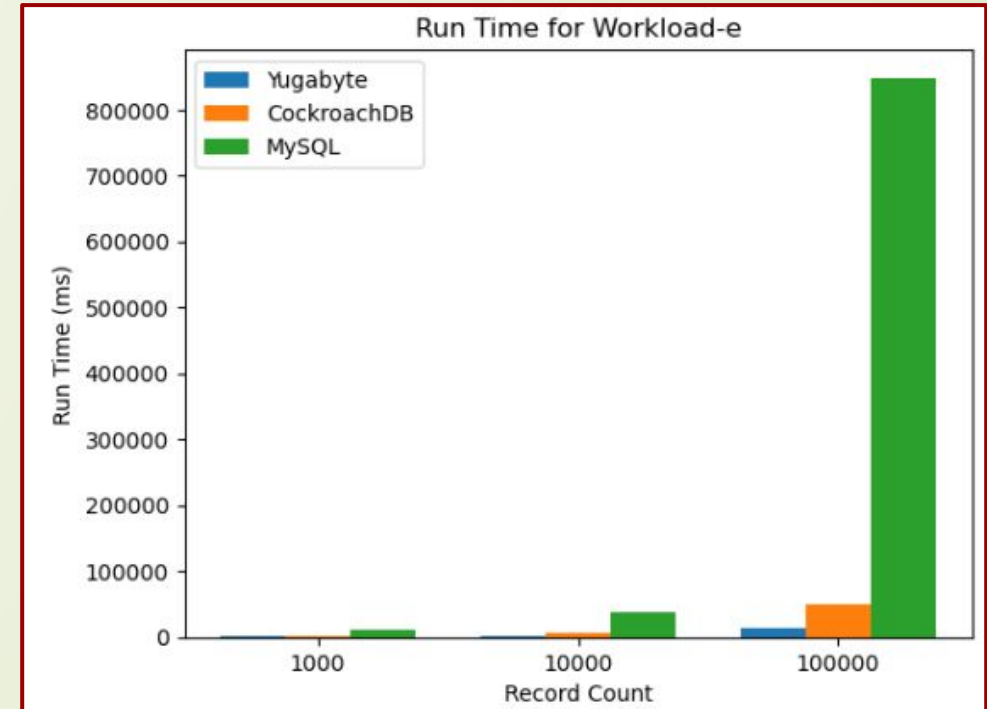
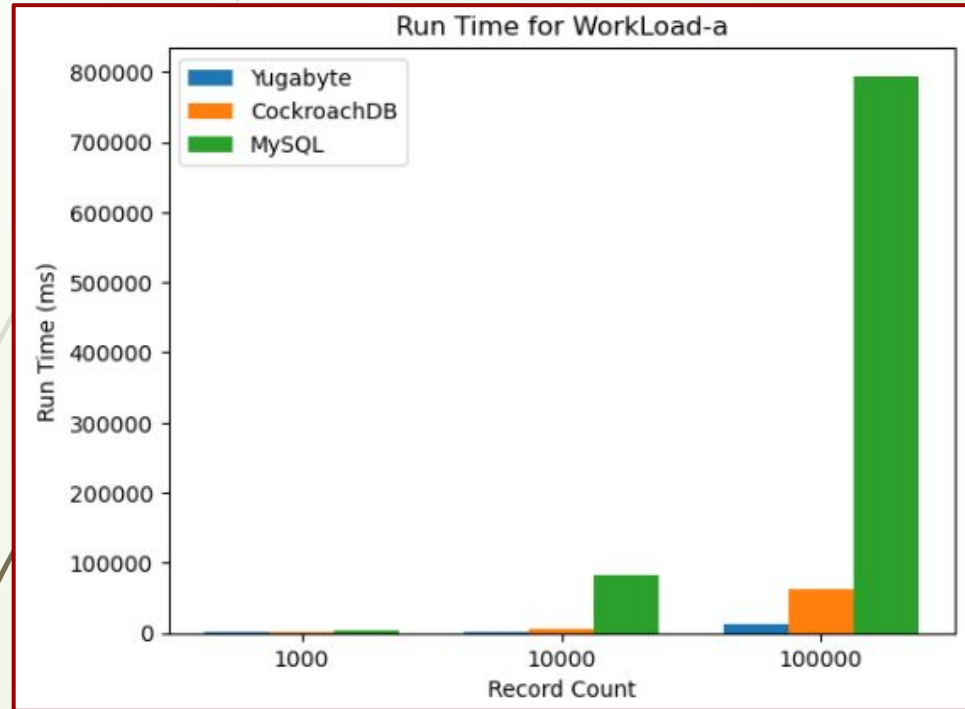
                elif db_name=='C':
                    command = """./bin/ycsb.sh {load_or_run} jdbc
                        -s -P ./cockroach-binding/cockroach.properties
                        -P workloads/workload{workload_name} -p recordcount={recordcount_name}
                        -p operationcount={operationcount_name}
                    """.format(
                        workload_name=wl,
                        recordcount_name=rec,
                        operationcount_name=op,
                        load_or_run=lr
                    )

                else:
                    print('Invalid DB parameter!!')
                    exit(-1)

                output_file = "./log/lr_{load_or_run}-wl_{workload_name}-rc_{recordcount_name}-oc_{operationcount_name}.txt".format( workload_name=wl,
                    recordcount_name=rec,
                    operationcount_name=op,
                    load_or_run=lr)
                run_bash_command(command,directory,output_file)

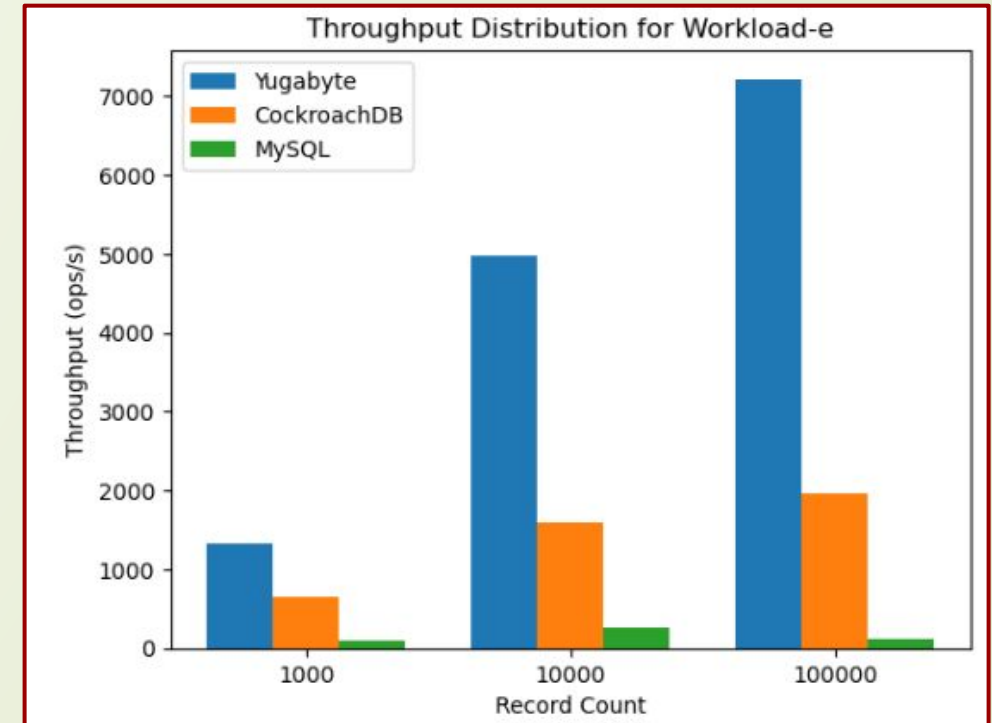
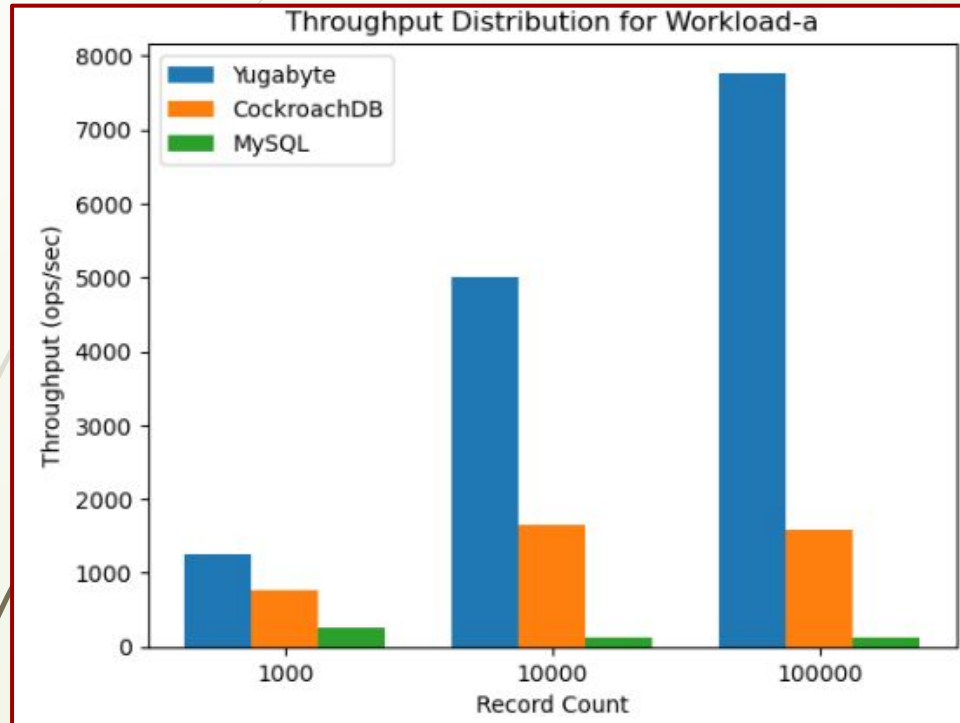
```

Run Time for Workload-a and Workload-e



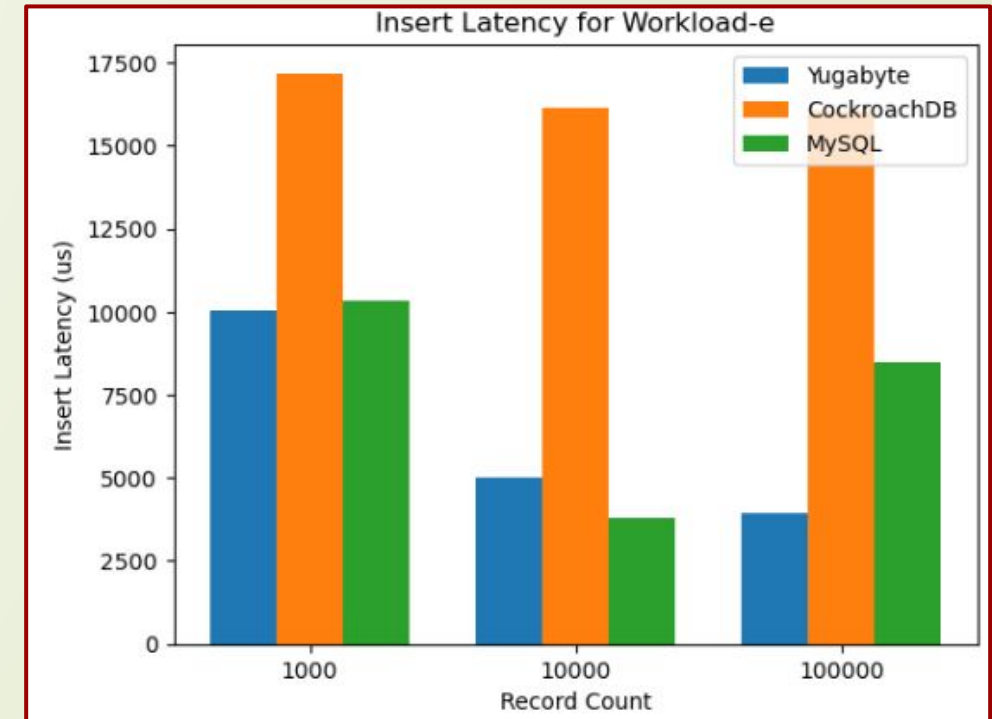
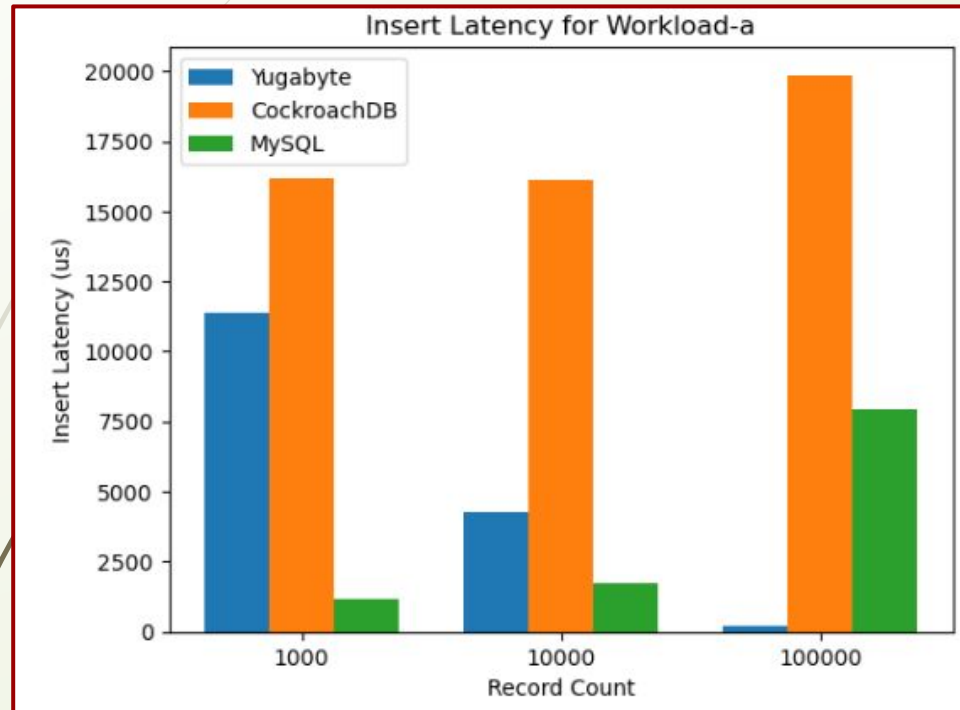
- ❑ Run Time (during load phase) measures the amount of time needed to load data into database.
- ❑ Best Performer - Yugabyte
- ❑ Worst Performer - MySQL

Throughput Distribution for Workload-a and Workload-e



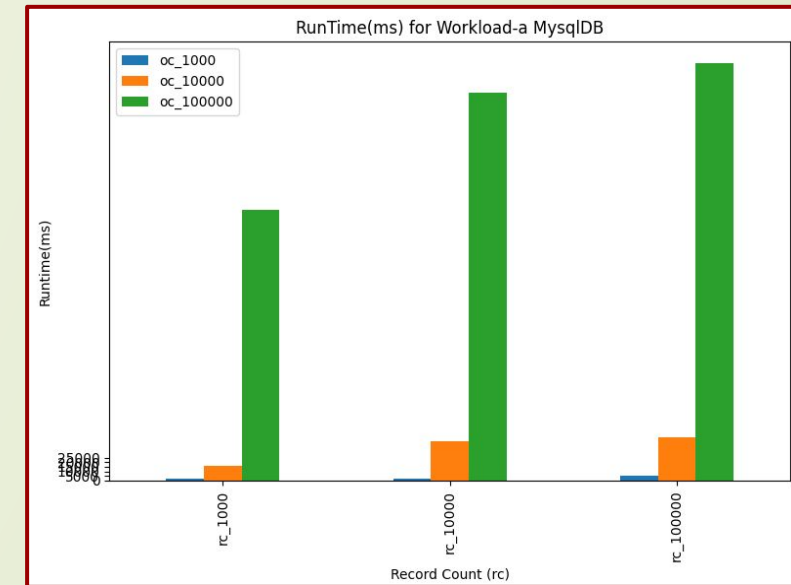
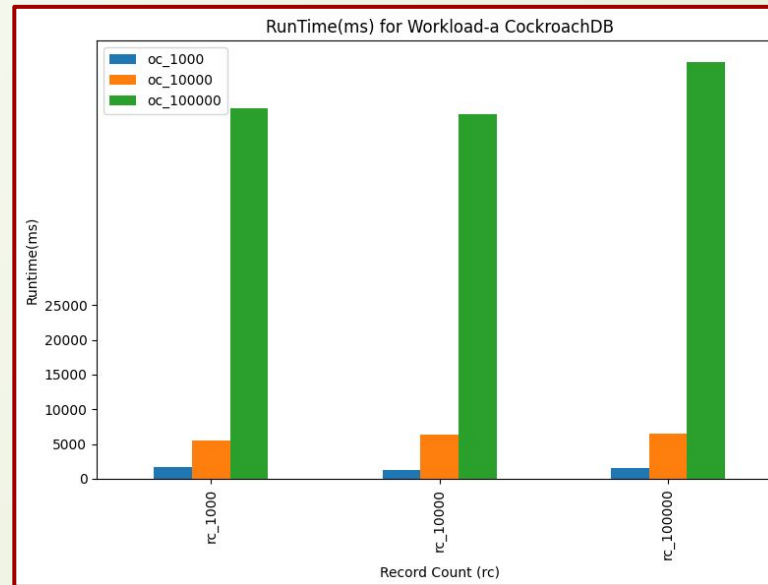
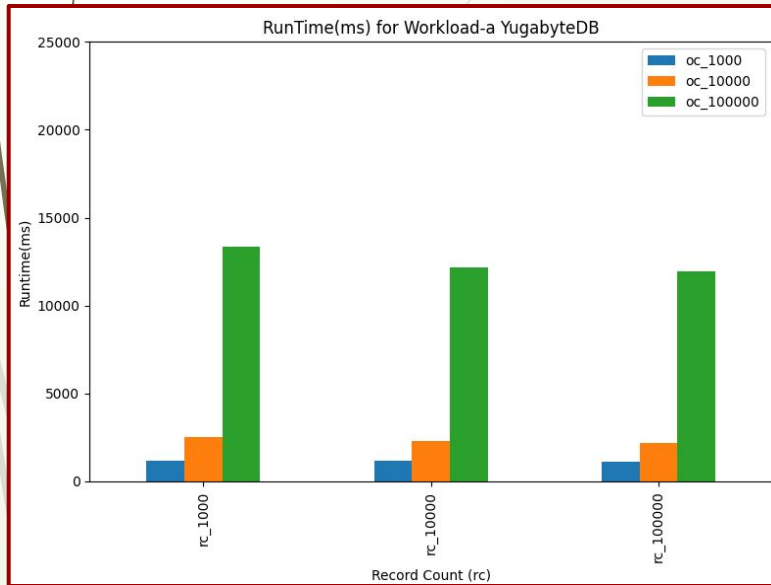
- ❑ Throughput Distribution measures the number of operations performed per second.
- ❑ Best Performer - Yugabyte
- ❑ Worst Performer - MySQL

Insert Latency for Workload-a and Workload-e



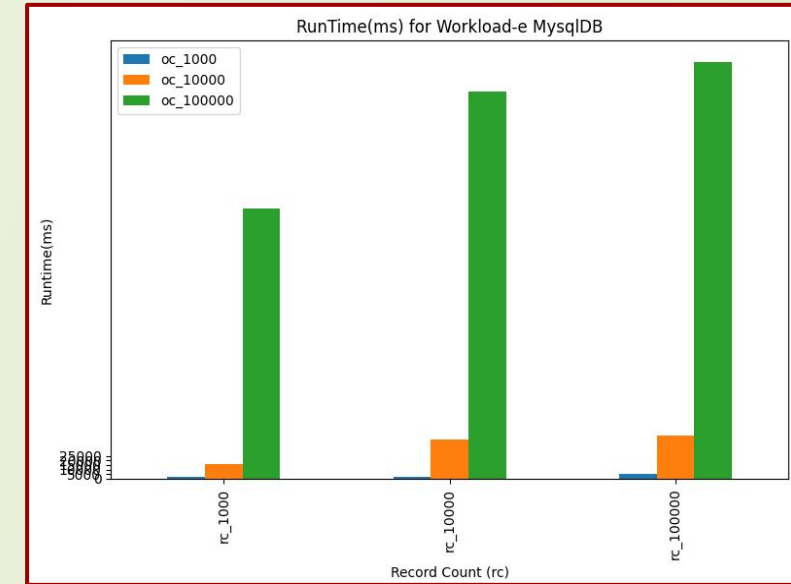
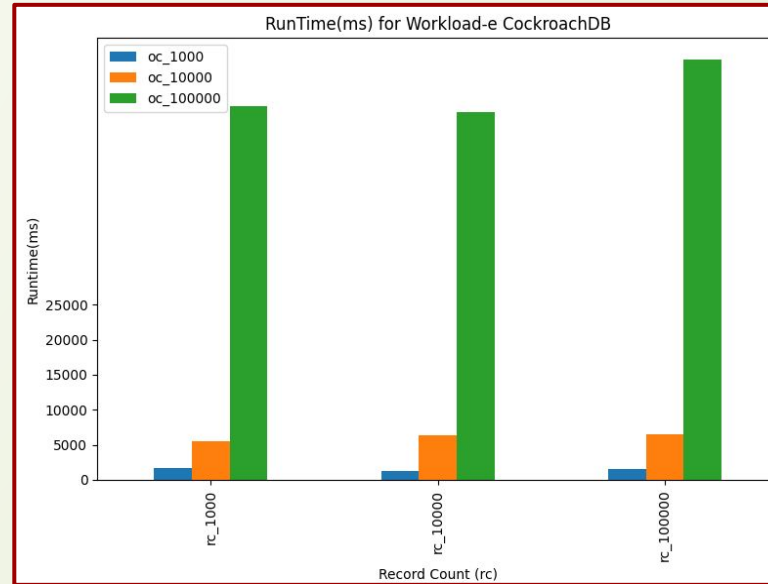
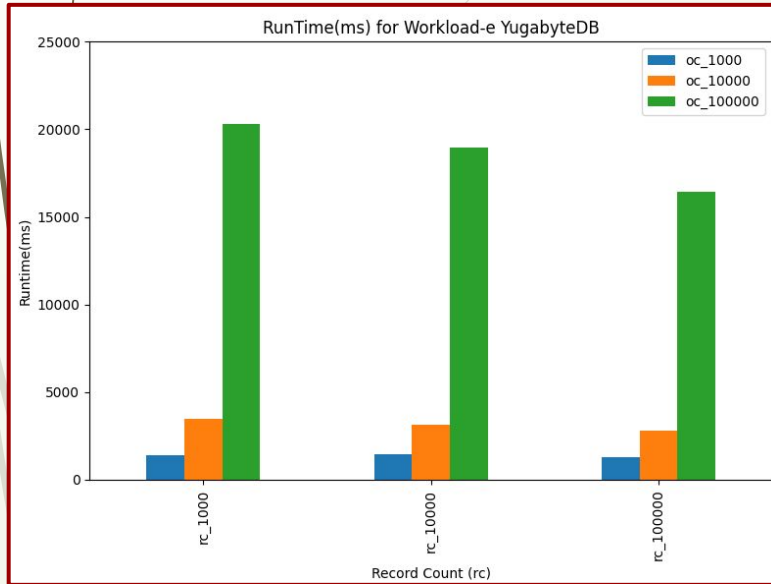
- ❑ Insert Latency measures the time delay between the initiation of an insert operation and its completion.
- ❑ Best Performer – Yugabyte (for high record count) /MySQL (for low record count)
- ❑ Worst Performer - CockroachDB

Run Time for Workload-a



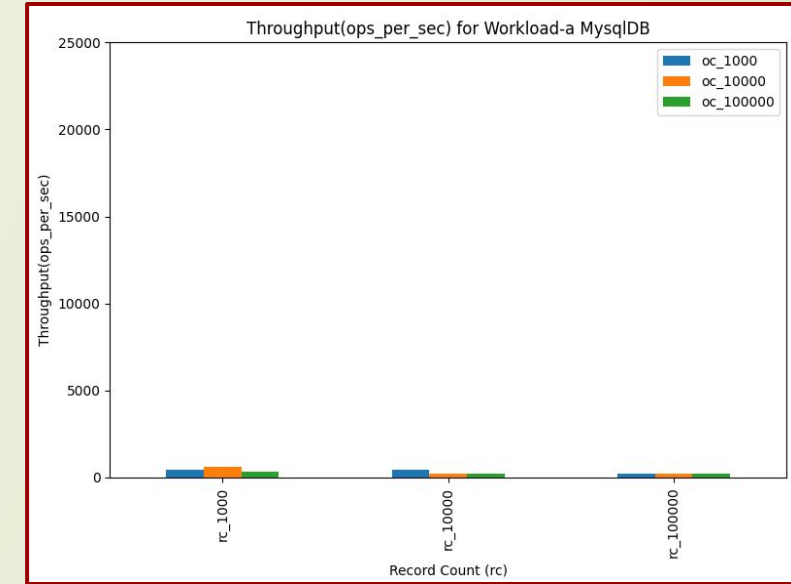
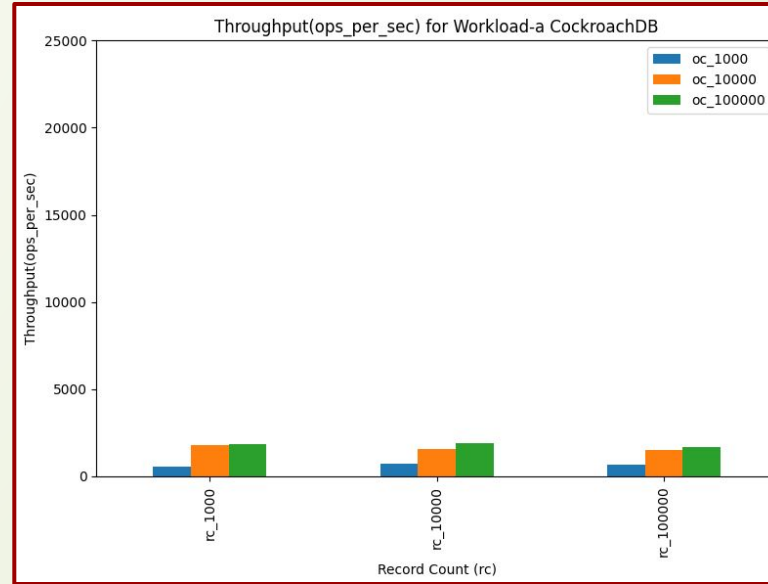
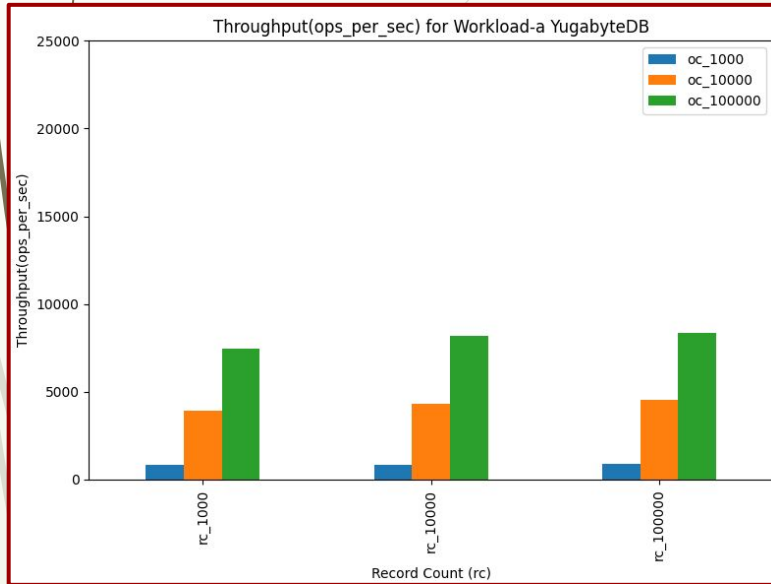
- ❑ Run Time measures the amount of time needed to execute different operations in database
 - ❑ Best Performer - Yugabyte
 - ❑ Worst Performer - MySQL

Run Time for Workload-e



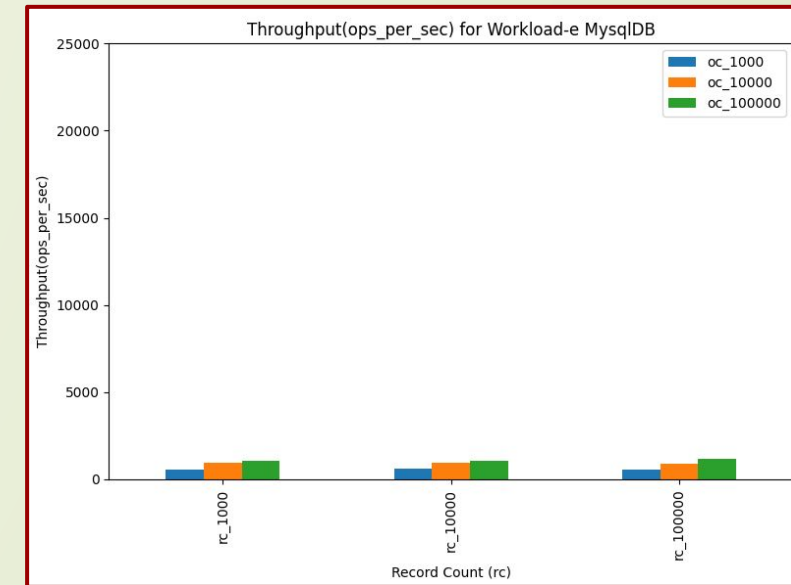
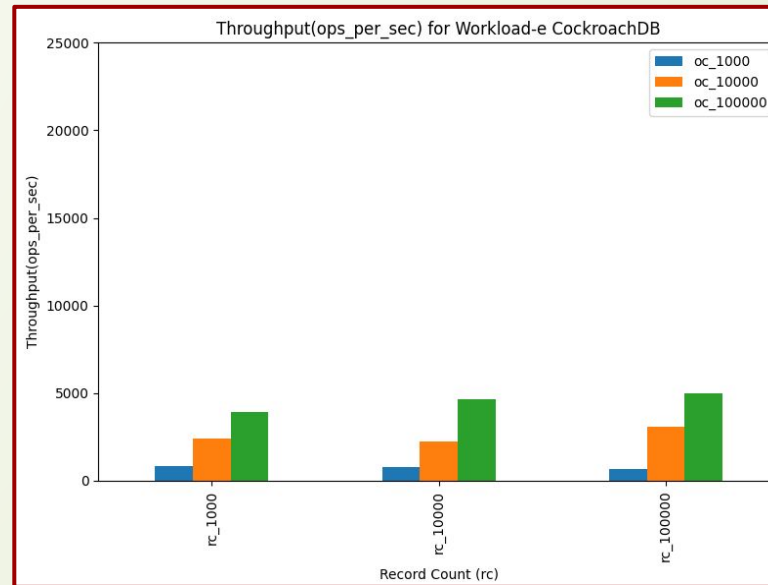
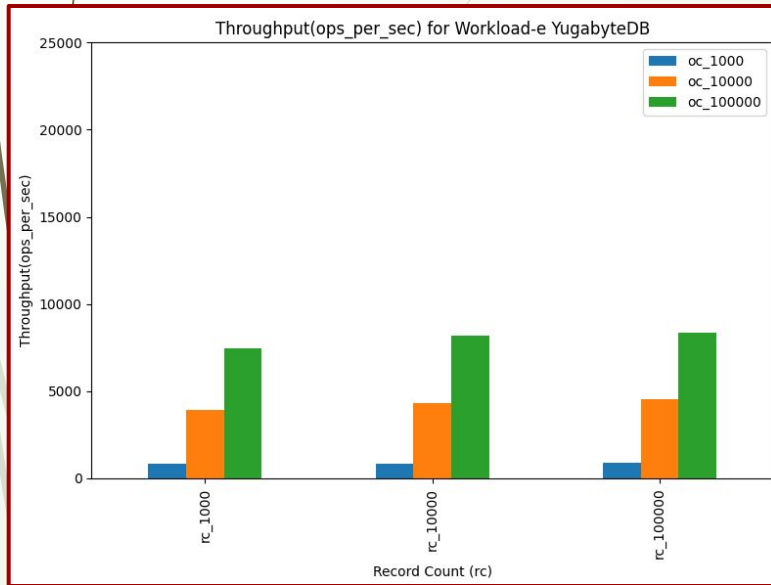
- ❑ Run Time measures the amount of time needed to execute different operations in database
 - ❑ Best Performer - Yugabyte
 - ❑ Worst Performer - MySQL

Throughput Distribution for Workload-a



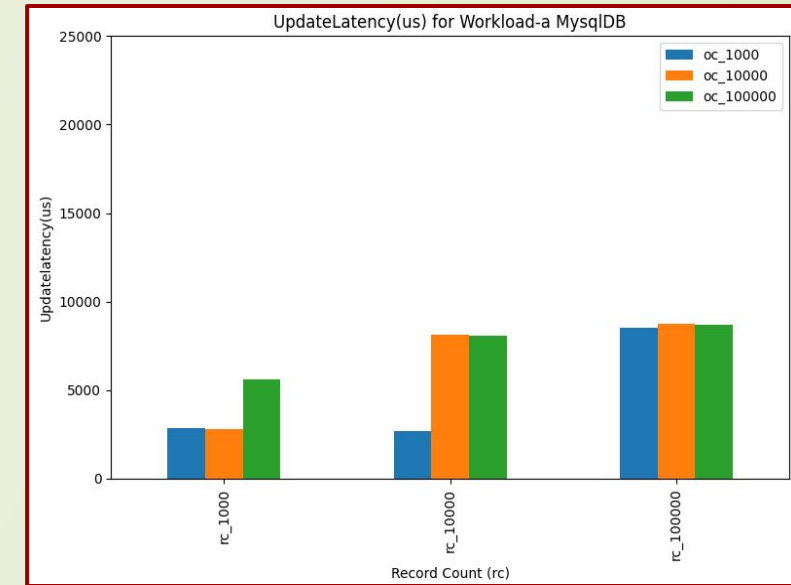
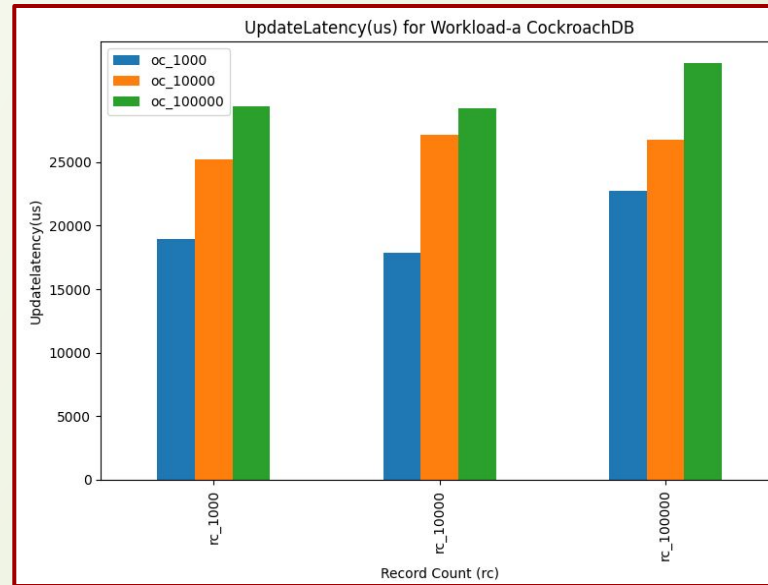
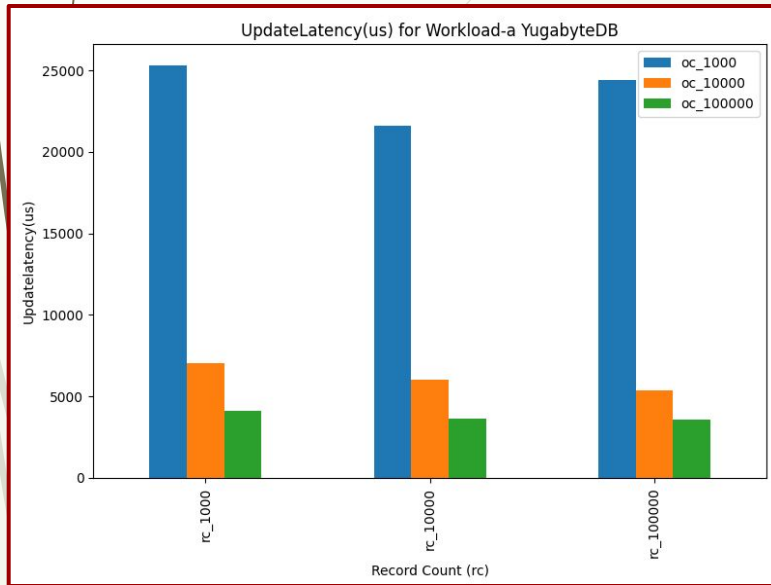
- ❑ Throughput Distribution measures the number of operations performed per second.
- ❑ Best Performer - Yugabyte
- ❑ Worst Performer - MySQL

Throughput Distribution for Workload-e



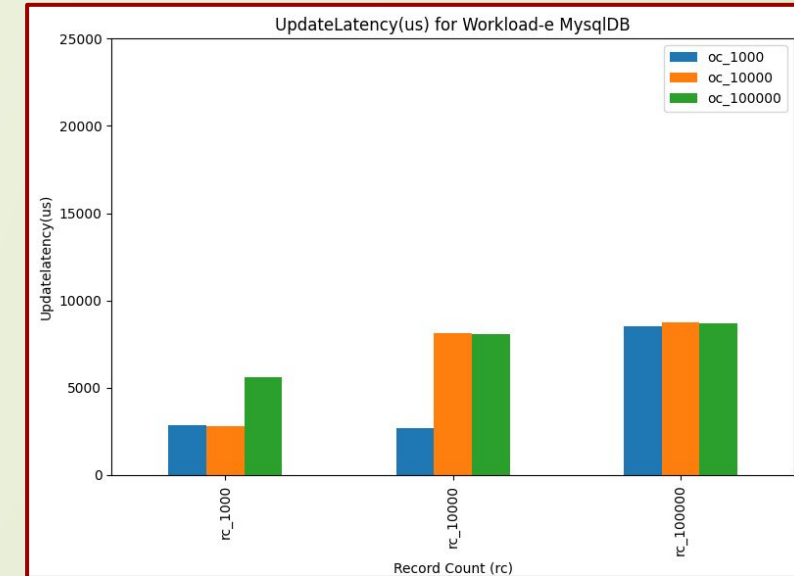
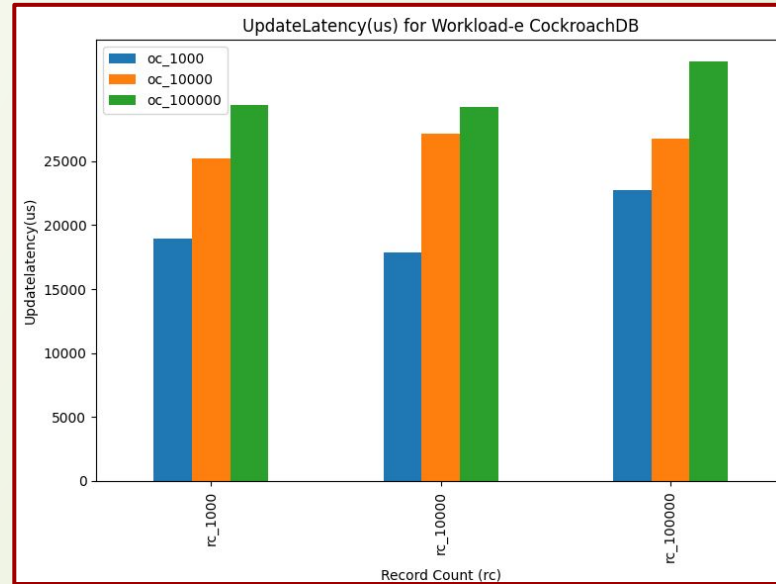
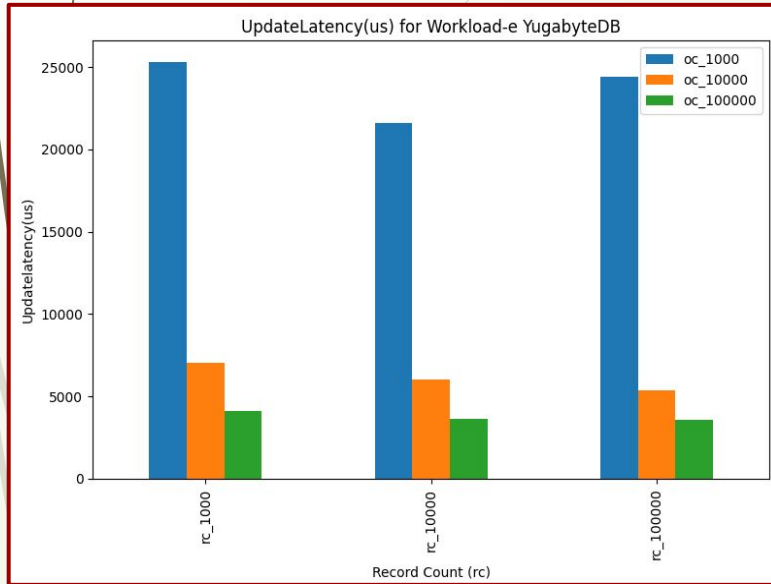
- ❑ Throughput Distribution measures the number of operations performed per second.
- ❑ Best Performer - Yugabyte
- ❑ Worst Performer - MySQL

Update Latency for Workload-a



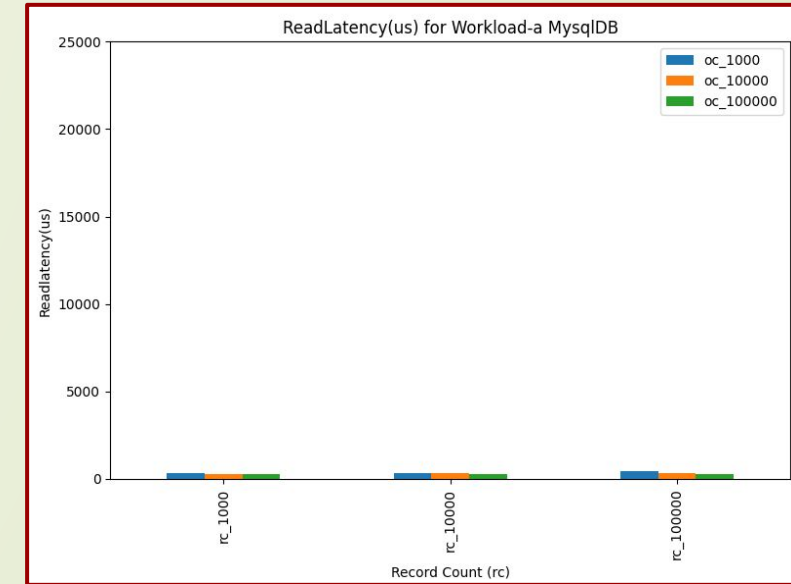
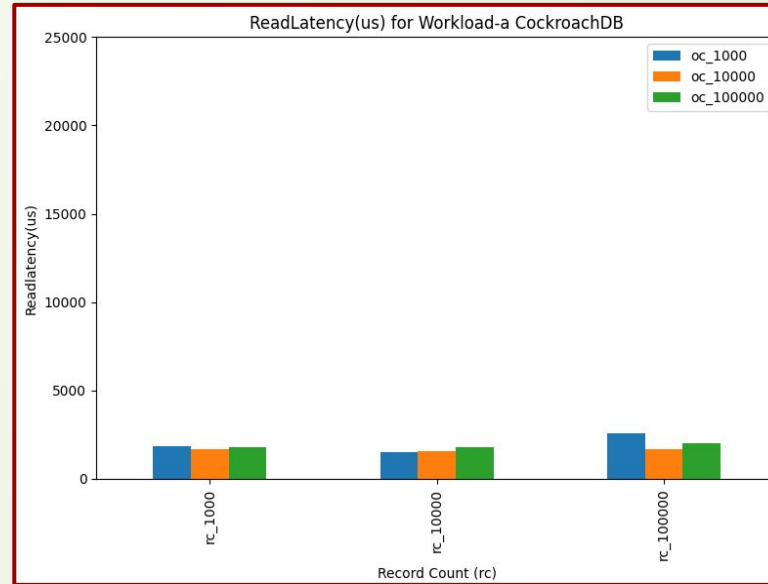
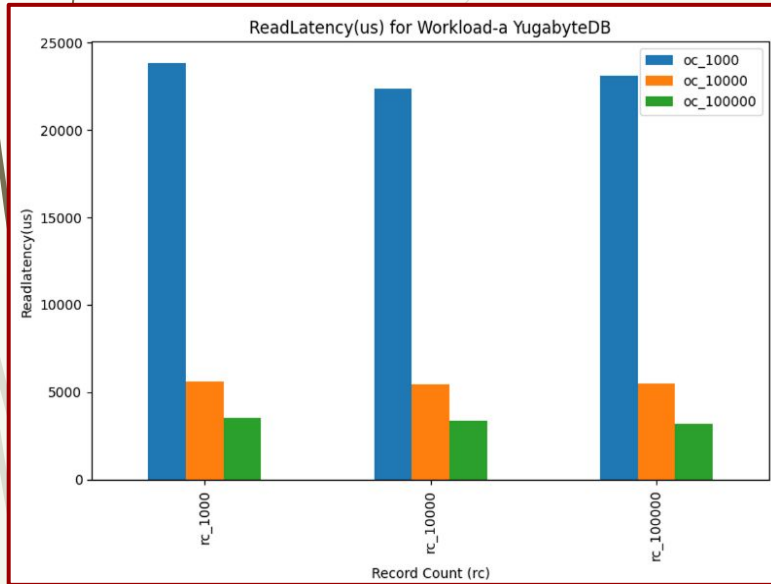
- ❑ Update Latency measures the time delay between the initiation of an update operation and its completion.
- ❑ Best Performer: YugabyteDB (for higher operation count), MySQL (consistent performance)
- ❑ Worst Performer
 - ❑ CockroachDB (for high operation count)
 - ❑ YugabyteDB (for low operation count)

Update Latency for Workload-e



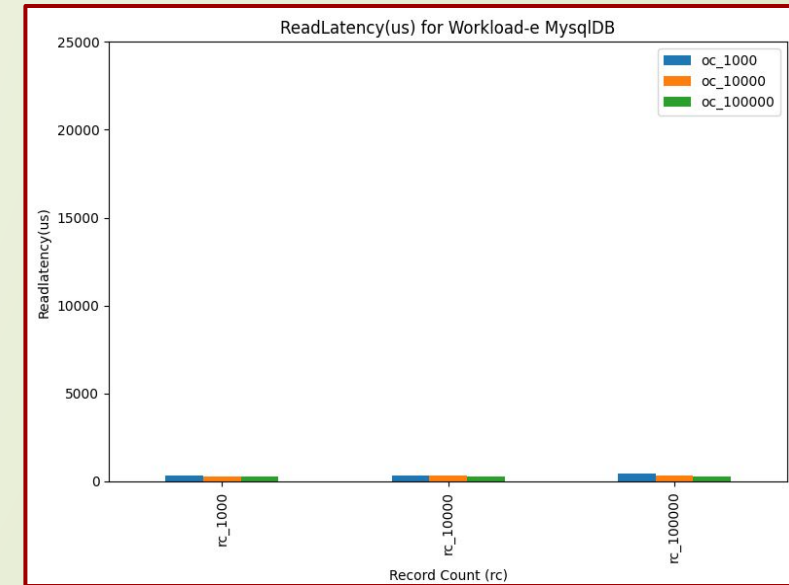
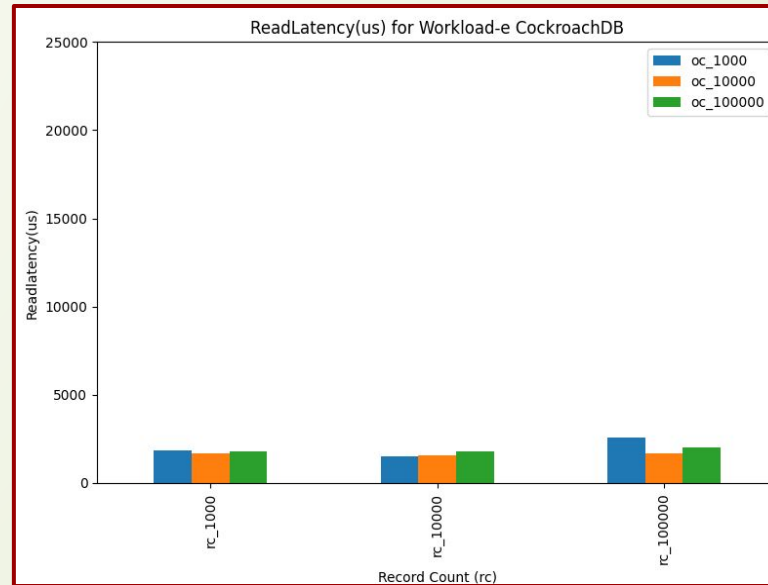
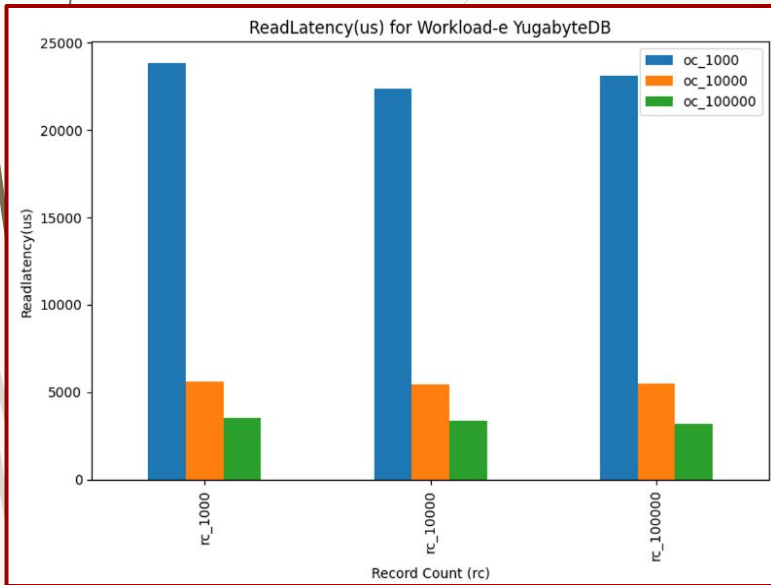
- ❑ Update Latency measures the time delay between the initiation of an update operation and its completion.
- ❑ Best Performer: YugabyteDB (for higher operation count), MySQL (consistent performance)
- ❑ Worst Performer
 - ❑ CockroachDB (for high operation count)
 - ❑ YugabyteDB (for low operation count)

Read Latency for Workload-a



- ❑ Read Latency measures the time delay between the initiation of a read operation and its completion.
- ❑ Best Performer - MySQL
- ❑ Worst Performer – Yugabyte
- ❑ Consistent Performer - CockroachDB

Read Latency for Workload-e



- ❑ Read Latency measures the time delay between the initiation of a read operation and its completion.
- ❑ Best Performer - MySQL
- ❑ Worst Performer – Yugabyte
- ❑ Consistent Performer - CockroachDB

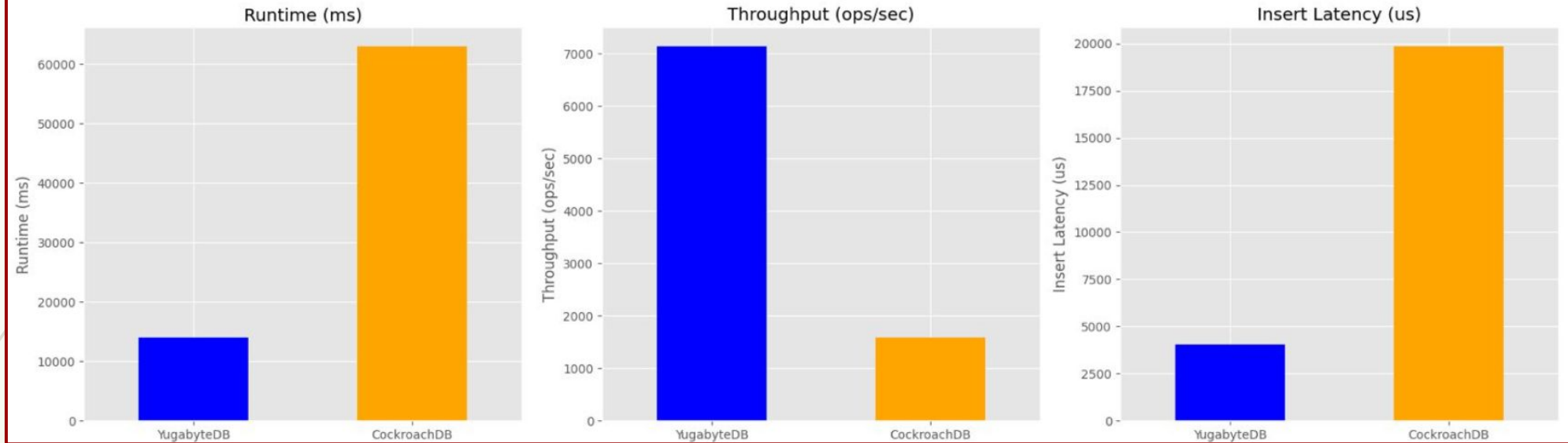
Use Case: E-commerce Platform

- ❑ Read Operation: 50%
 - ❑ Read Operation is the most frequently performed operation in e-commerce platforms
 - ❑ Occurs every time a user browses products, views details, or checks inventory
- ❑ Scan Operation: 25%
 - ❑ Scan operations involves searching and filtering products
 - ❑ Users often filter products based on categories, prices, or other attributes
- ❑ Insert Operation: 10%
 - ❑ Insert operations occur when new products are added to the catalog
 - ❑ This happens less frequently than reads or scans
- ❑ Update Operation: 10%
 - ❑ Update operations are necessary for changing product details or inventory levels
 - ❑ They are as frequent as inserts but are crucial for maintaining accurate and current data
- ❑ Delete Operation: 5%
 - ❑ Delete operations are the least frequently performed operation
 - ❑ Products are removed from listings less often than they are added, viewed, or updated

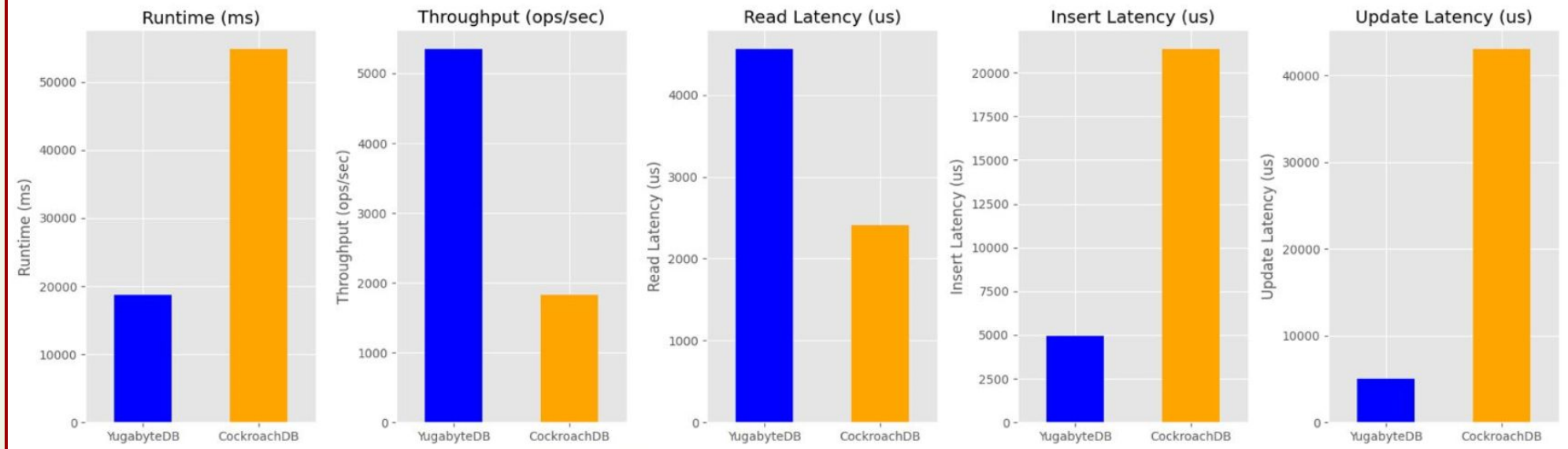
Use Case: Workload Configuration

```
recordcount=100000  
operationcount=100000  
workload=site.ycsb.workloads.CoreWorkload  
  
readallfields=true  
  
readproportion=0.5  
updateproportion=0.25  
scanproportion=0.1  
insertproportion=0.1  
deleteproportion=0.05
```


Comparison of Performance Metrics during Load Phase



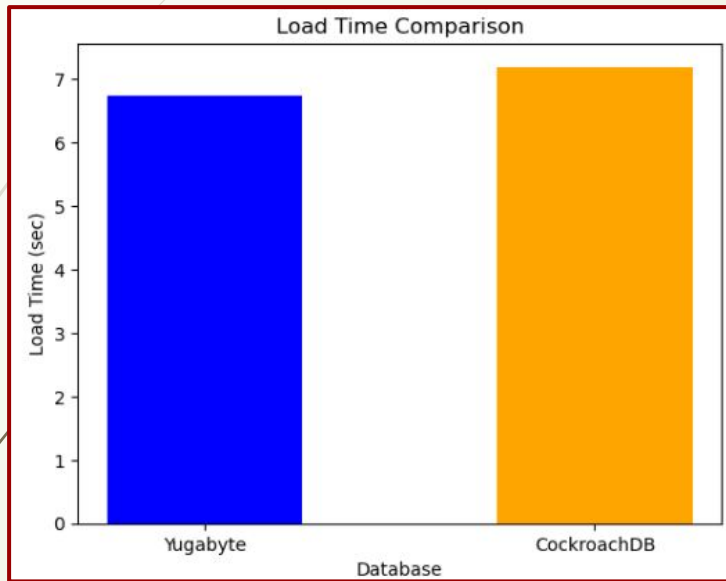
Comparison of Performance Metrics during Run Phase



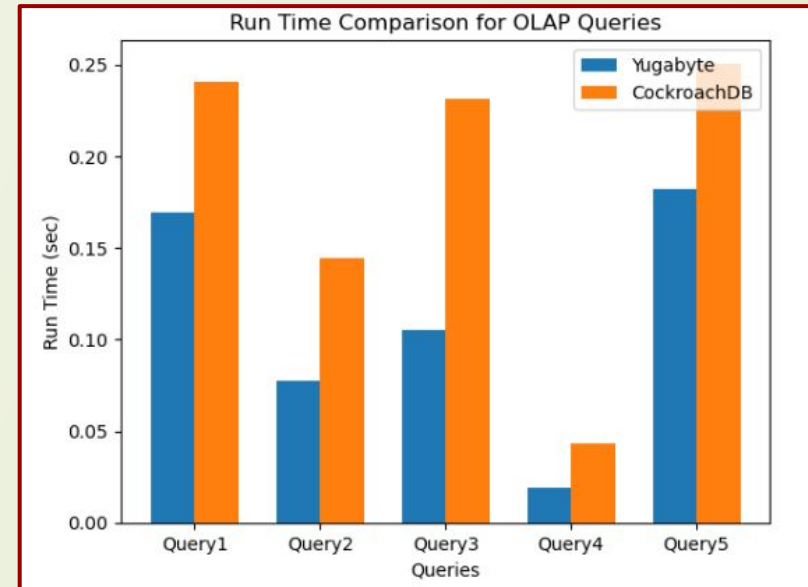
YugabyteDB: High Read Latency

- ❑ YugabyteDB
 - ❑ Provides linearizable consistency
 - ❑ A read operation waits for any ongoing writes to be fully replicated
 - ❑ A record with N columns is splitted into N key/value pairs during data storage. So, fetching of a record requires fetching all these key/value pair
 - ❑ LSM trees are great for write-intensive applications, the compaction process can sometimes affect read performance and lead to temporary spikes in latency
- ❑ CockroachDB
 - ❑ Provides serializable isolation (highest level of isolation)
 - ❑ Transactions appear to execute one after the other without any interleaving, even though they may be executed concurrently.
 - ❑ A record with any number of columns is stored as a single key/value pair. So, fetching of a record requires fetching corresponding single key/value pair

Performance Test in OLAP Queries



- ❑ Tables Loaded
 - ❑ Products (10,000 records)
 - ❑ Customers (30,000 records)
 - ❑ Sales (100,000 records)



- ❑ OLAP Queries executed
 - ❑ Total Sales by Category for a Given Year
 - ❑ Monthly Sales Trend for a Specific Product
 - ❑ Top 10 Customers by Sales Volume
 - ❑ Stock Analysis
 - ❑ Sales Performance by Region

Conclusion

- ❑ In terms of performance metrics like load time, throughput, update latency and insert latency, YugabyteDB seems to be performing better.
- ❑ However, read latency for YugabyteDB is high, making CockroachDB better in this aspect.



THANK YOU