Benchmarking Cloud Native Database

Iqbal Farabi and Tara Baskara
KubeCon + CloudNativeCon Europe 2019

KubeCon CloudNativeCon Europe 2019

Hola!

Iqbal Farabi
System Engineer
Go-Jek Indonesia
@iqbal_farabi

Tara Baskara
System Engineer
Go-Jek Indonesia
@iqbal_farabi

We’re from Jakarta, Indonesia


GO JEK


GOJEK
GOJEK
GOJEK
GOJEK
GOJEK

Brief History

01 Go-Jek 1.0

Call-center
for ojek service

02 Mobile App

First mobile app was
launched

03 More Apps!

Incl. growth of Go-Pay
into P2P, withdrawal, etc

04 Fintech

Acquired three fintech
companies in Indonesia

midtrans

kartuku
Secure Payments. Simplified.

mapan

2010 2015 2017 2018

JAKARTA

BANGALORE

SINGAPORE

THAILAND

VIETNAM

The Journey Continues…

GO JEK

Outline

Today we will discuss about…

Cloud Native Database

- What is cloud native database? What databases do we pick to run the experiments on? What are their characteristics?

YCSB

- What is YCSB? What does it measure? What are the different kind of workloads defined?

Experiments and Results

- Setup, experiments, and brief explanation of the results.

GO JEK

Cloud Native Database

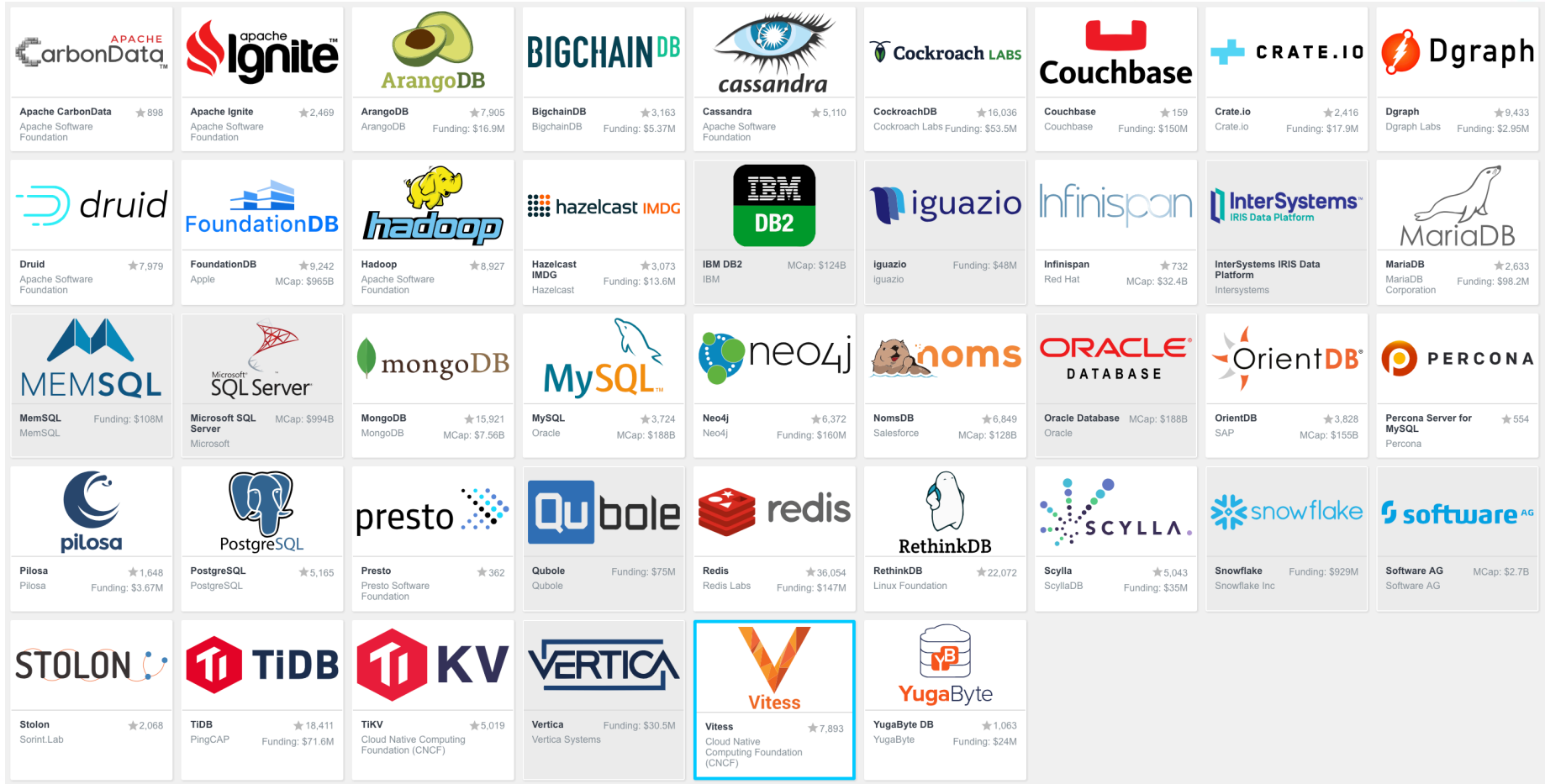
KubeCon | CloudNativeCon

Europe 2019

Definition

CNCF defines cloud native technologies as, "technologies that empower organizations to build and run scalable applications in modern and dynamic environments. Cloud native technologies enable loosely coupled systems that are resilient, manageable, and observable”.

Landscape


Apache CarbonData ★898 Apache Software Foundation
Apache Ignite ★2,469 Apache Software Foundation
ArangoDB ★7,905 ArangoDB Funding: $16.9M
BigchainDB ★3,163 BigchainDB Funding: $5.37M
Cassandra ★5,110 Apache Software Foundation
CockroachDB ★16,036 Cockroach Labs Funding: $53.5M
Couchbase ★159 Couchbase Funding: $150M
Crate.io ★2,416 Crate.io Funding: $17.9M
Dgraph ★9,433 Dgraph Labs Funding: $2.95M
Druid ★7,979 Apache Software Foundation
FoundationDB ★9,242 Apple MCap: $965B
Hadoop ★8,927 Apache Software Foundation
Hazelcast IMDG ★3,073 Hazelcast Funding: $13.6M
IBM DB2 MCap: $124B IBM
iguazio Funding: $48M iguazio
Infinispan ★732 Red Hat MCap: $32.4B
InterSystems IRIS Data Platform Intersystems
MariaDB ★2,633 MariaDB Corporation Funding: $98.2M
MemSQL Funding: $108M MemSQL
Microsoft SQL Server MCap: $994B Microsoft
MongoDB ★15,921 MongoDB MCap: $7.56B
MySQL ★3,724 Oracle MCap: $188B
Neo4j ★6,372 Neo4j Funding: $160M
NomsDB ★6,849 Salesforce MCap: $128B
Oracle Database MCap: $188B Oracle
OrientDB ★3,828 SAP MCap: $155B
Percona Server for MySQL ★554 Percona
Pilosa ★1,648 Pilosa Funding: $3.67M
PostgreSQL ★5,165 PostgreSQL
Presto ★362 Presto Software Foundation
Qubole Funding: $75M Qubole
Redis ★36,054 Redis Labs Funding: $147M
RethinkDB ★22,072 Linux Foundation
Scylla ★5,043 ScyllaDB Funding: $35M
Snowflake Funding: $929M Snowflake Inc
Software AG MCap: $2.7B Software AG
Stolon ★2,068 Sorint.Lab
TiDB ★18,411 PingCAP Funding: $71.6M
TiKV ★5,019 Cloud Native Computing Foundation (CNCF)
Vertica Funding: $30.5M Vertica Systems
Vitess ★7,893 Cloud Native Computing Foundation (CNCF)
YugaByte DB ★1,063 YugaByte Funding: $24M

Filtering

- Open source
- Operational database
- ACID compliance
- Provides SQL-like API

The List

CockroachDB

- https://www.cockroachlabs.com/

TiDB

- https://pingcap.com/en/

YugaByte DB

- https://www.yugabyte.com/

CockroachDB

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store.

It aims to:

- Scales horizontally
- Provides fault resiliency
- Supports strongly-consistent ACID transactions
- Provides a familiar SQL API with PostgreSQL-like sytax

CockroachDB Architecture

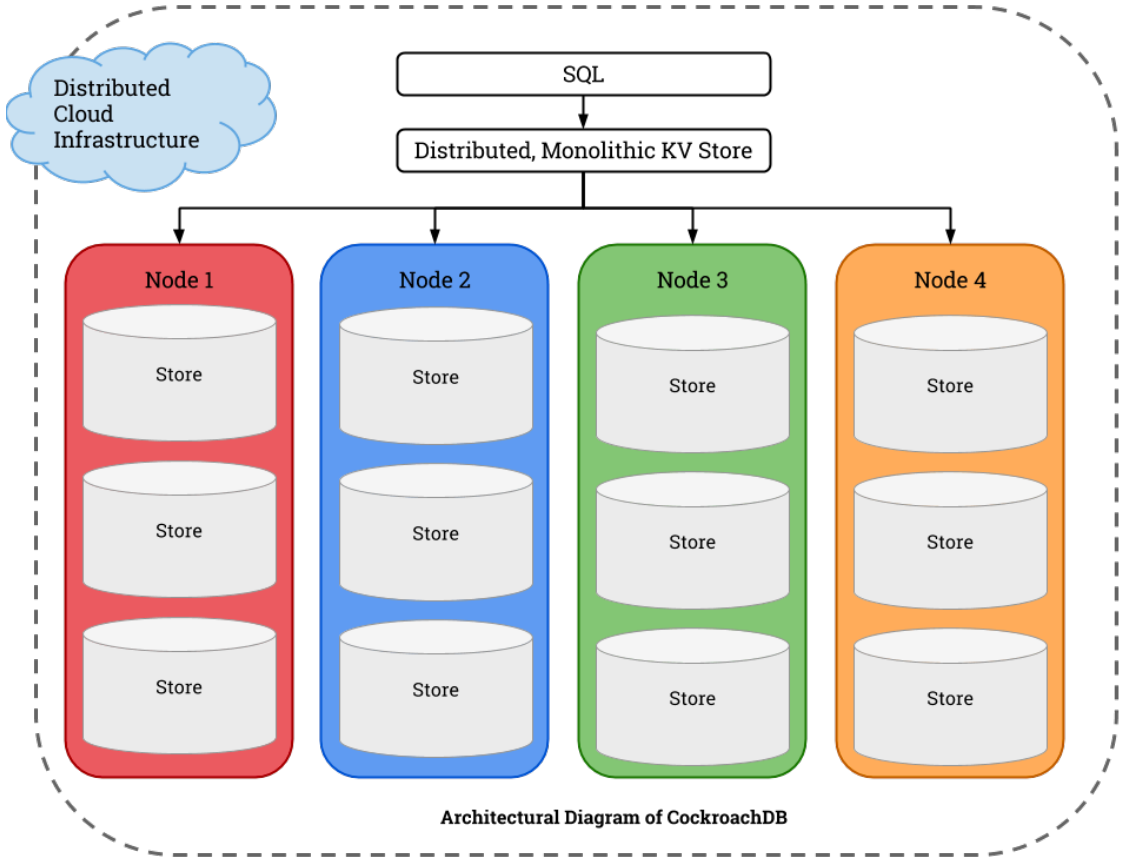


image source: https://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

TiDB

TiDB is an open-source NewSQL database that supports Hybrid Transactional and Analytical Processing (HTAP) workloads.

It aims to:

- Scales horizontally
- Provides fault resiliency
- Supports distributed transactions with strong consistency
- Provides a familiar SQL API with MySQL-like syntax

TiDB Architecture

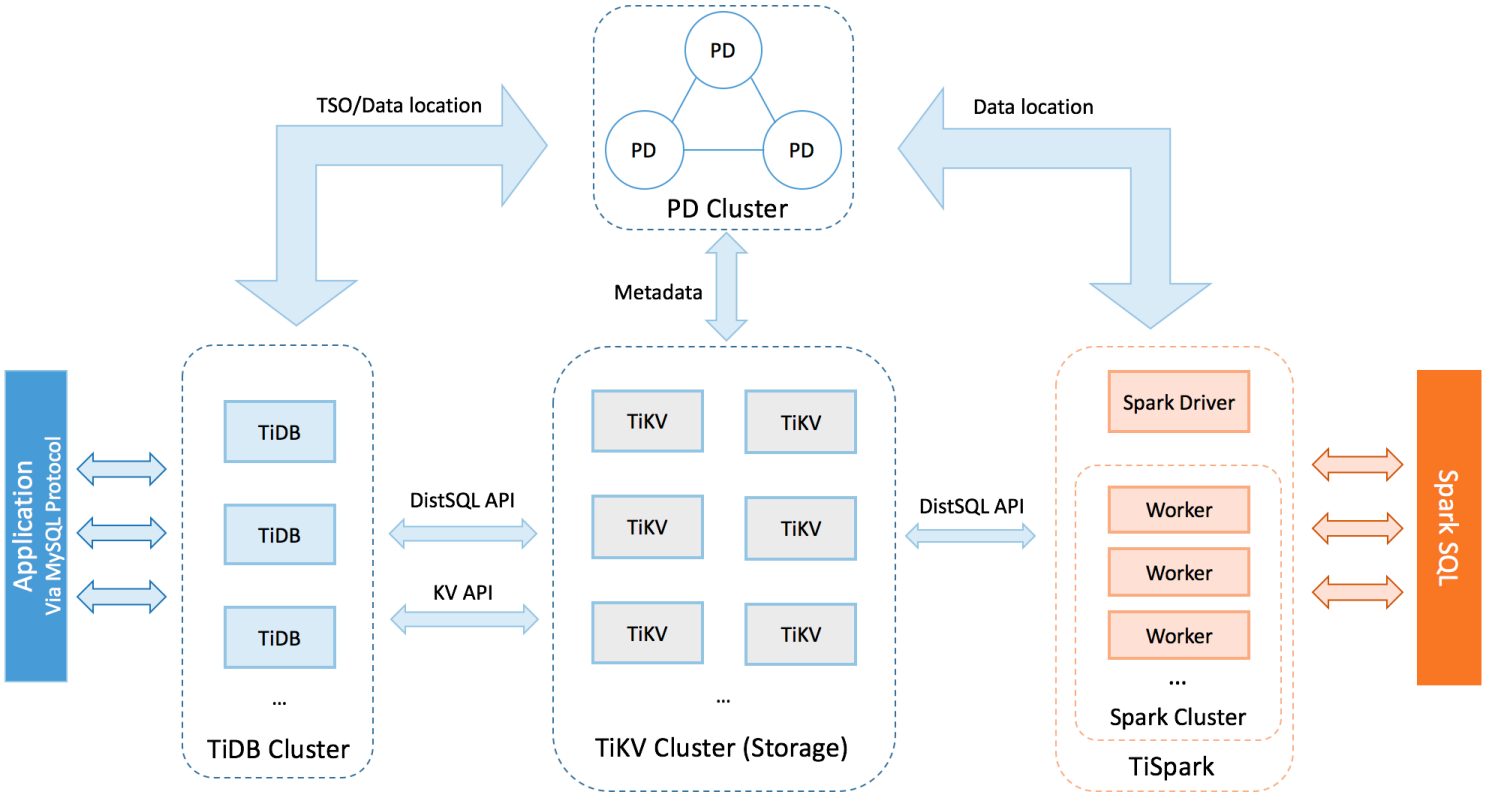


image source: https://pingcap.com/docs/architecture/

YugaByte DB

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store.

It aims to:

- Scales with autosharding
- Provides fault resiliency
- Supports multi-shard ACID transactions
- Provides YugaByte Structured Query Language (YSQL) and YugaByte Cloud Query Language (YCQL) APIs.

YugaByte DB Architecture

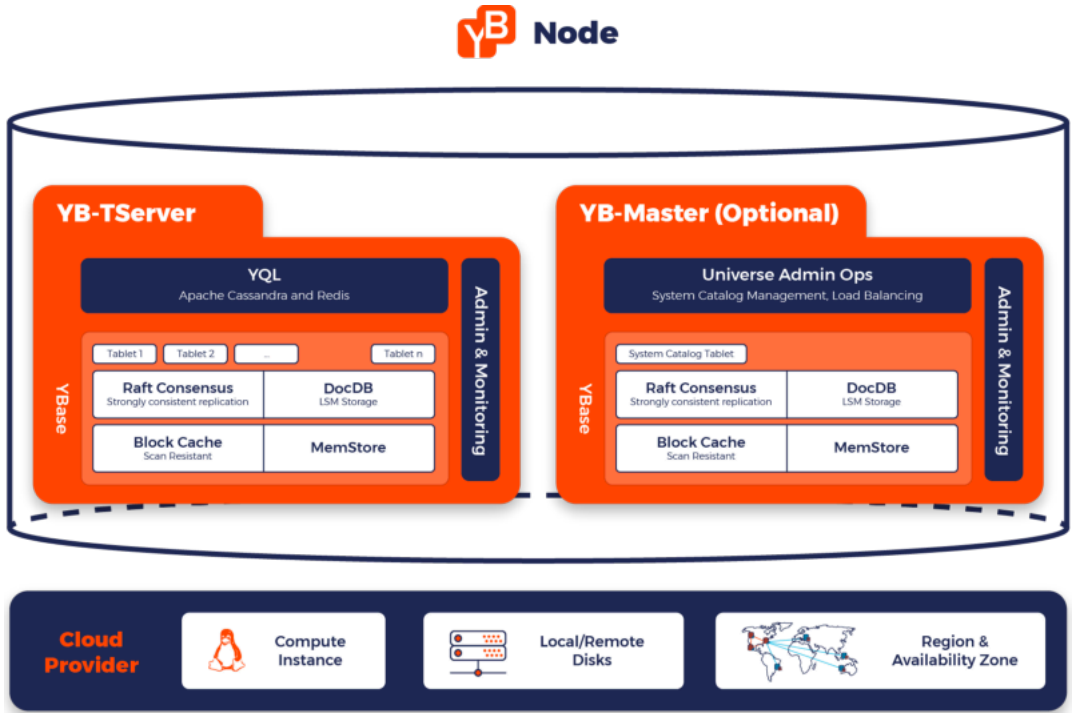


image source: https://blog.yugabyte.com/yugabyte-db-architecture-diverse-workloads-with-operational-simplicity/

Classifications (1)

According to Brian F. Cooper et al., in their YCSB paper, there are four main tradeoffs faced by cloud serving systems:

- read vs write performance
- latency vs durability
- synchronous vs asynchronous replication
- data partitioning

Classifications (2)

Latency vs Durability

Write operations synced to disk before returning success will increase durability in the case of system failures. However it might lower throughput and increase latency.

Write operations stored in memory and synced later will increase throughput and decrease latency, but might increase the risk of data loss in case of system failures.

Classifications (3)

Synchronous vs Asynchronous Replication

Synchronous replication ensures consistency among all nodes but might increase latency.

Asynchronous replication decreases latency but might cause data loss if failure happens to nodes with data that not replicated yet.

Classifications (4)

Data Partitioning

Strictly row-based partitioning allows efficient access to an entire record of data.

Column-based partitioning allows efficient access for a subset columns when retrieving multiple records.

Classifications (5)

Database	Latency/durability	Sync/async replication	Row/column partitioning
CockroachDB	Durability	Synchronous	Row
TiDB	Durability	Synchronous	Row
YugaByte DB	Latency	Asynchronous	Row

GO JEK

YCSB

Yahoo! Cloud Serving Benchmark

Created by Brian F. Cooper et al. to create a **standard benchmark** and benchmarking framework to assist in the evaluation of different **cloud systems**.

Focus on **serving systems,** which serve read and write workloads, over batch or analytical systems.

Workloads Data

YCSB workloads data look like the following:

- 1 table named “usertable”
- 10 string fields, 1 primary key with content like “user123456”, 9 fields with content a random string of ASCII characters with 100 bytes length
- 1,000,000 records
- 1,000,000 operations with 1,000 threads for Workload A, B, C, and D
- 1,000,000 operations with 100 threads for Workload E

Operations

Operations performed by YCSB are:

- Insert: insert new record.
- Update: update a record by replacing the value of one field.
- Read: read a record, either one randomly chosen field or all fields.
- Scan: scan records in order, starting at randomly chosen record key with randomly chosen number of records.

Distributions

To choose which operations (insert, update, read, or scan) to perform on which records and how many records, YCSB has several built-in distributions:

- Uniform: choose an item uniformly at random.
- Zipfian: some item will be extremely popular, most records will be unpopular.
- Latest: like Zipfian with preference of latest inserted records as popular distribution.
- Multinomial: probabilities of each item can be specified.

Workloads

Workload	Operations	Distribution	Application Example
A – Update Heavy	Read: 50% Update: 50%	Zipfian	User session: session store recording recent actions
B – Read Heavy	Read: 95% Update: 5%	Zipfian	Photo tag: add tag is update, but most ops are reading tags
C – Read Only	Read: 100%	Zipfian	User profile cache
D – Read Latest	Read: 95% Insert: 5%	Latest	User status updates; people want to read the latest
E – Short Ranges	Scan: 95% Insert: 5%	Zipfian/Uniform	Threaded conversations

GO JEK

Experiments and Results

Cluster Setup

3 nodes GKE cluster with following specifications:

- n1-standard-16 machine type
- 1000 GB Local SSD
- 60 GB RAM

Statefulset Setup

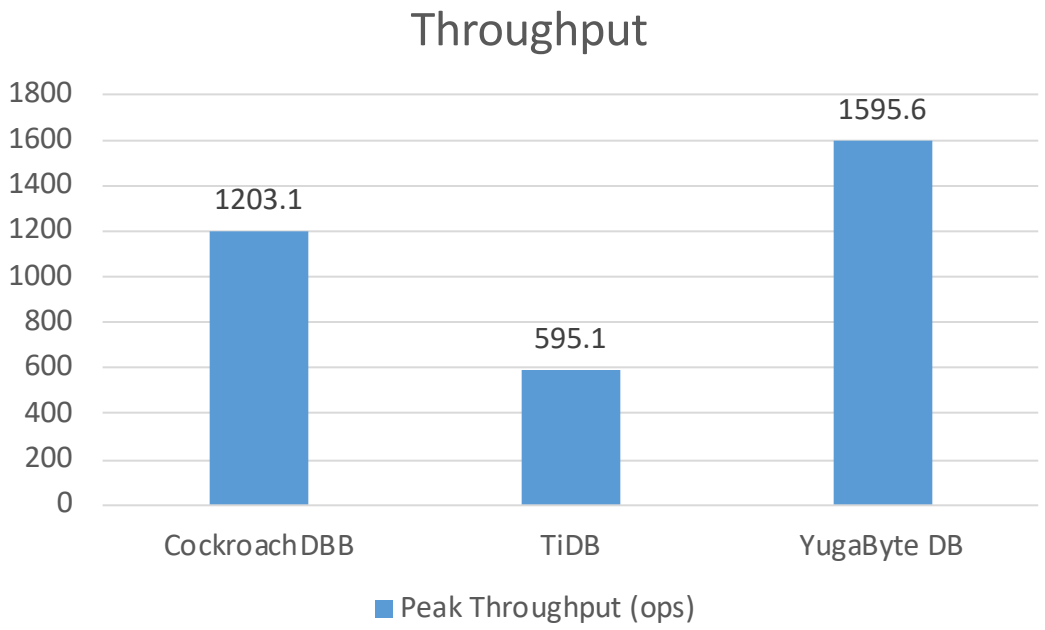
Resource:

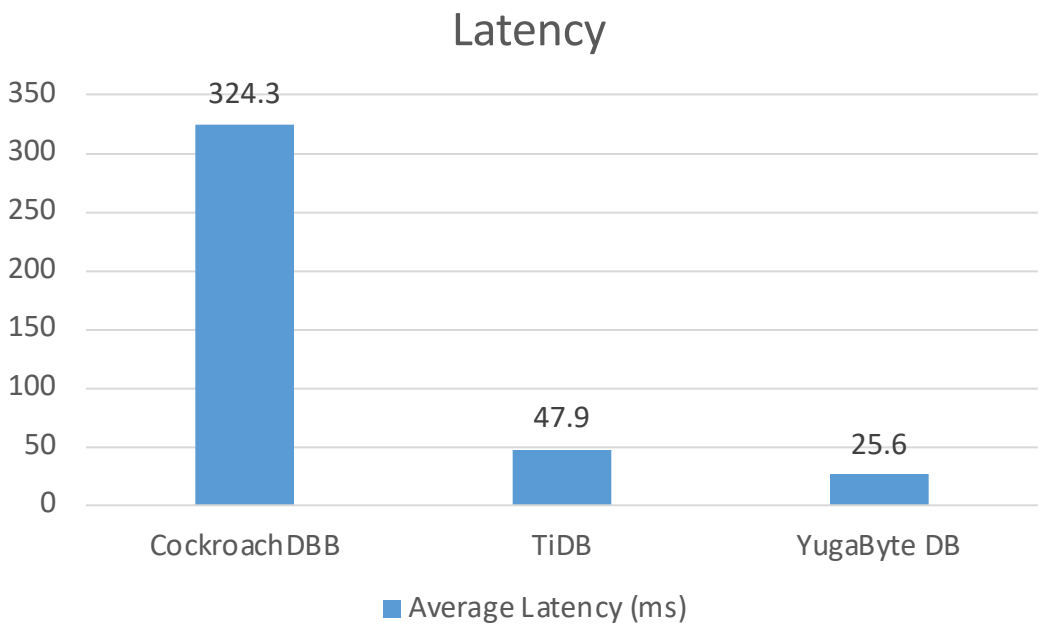
- 14 vCPU request, 16 vCPU limit
- 30 GB RAM request, 60 GB RAM limit
- 500 GB SSD local persistent volume

go-ycsb

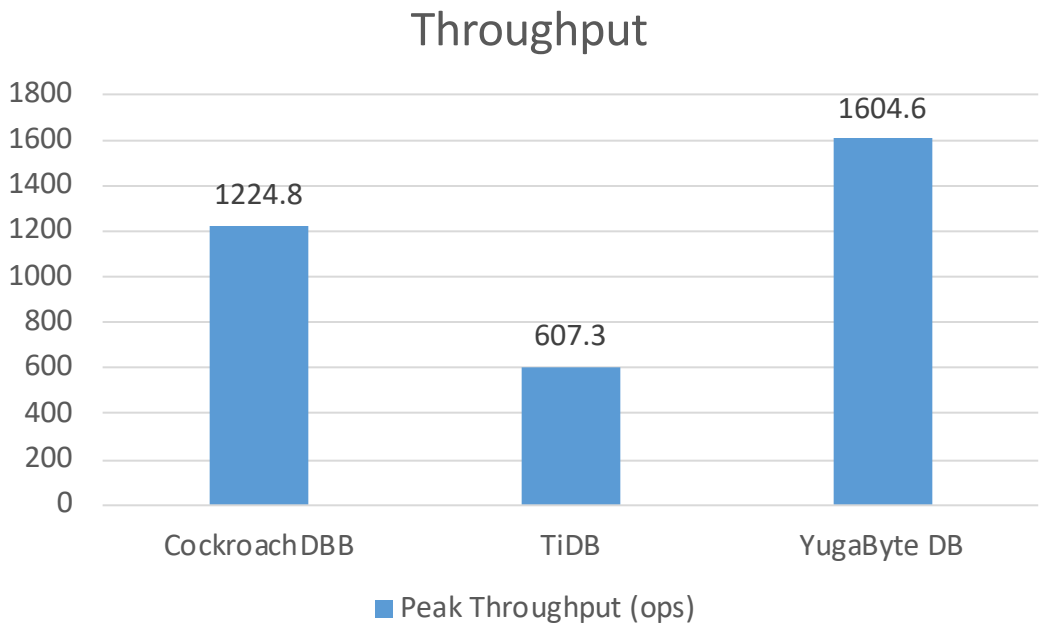
All experiments in this presentation all done using a Go port of YCSB called go-ycsb created by engineers at PingCap, the company that creates TiDB.

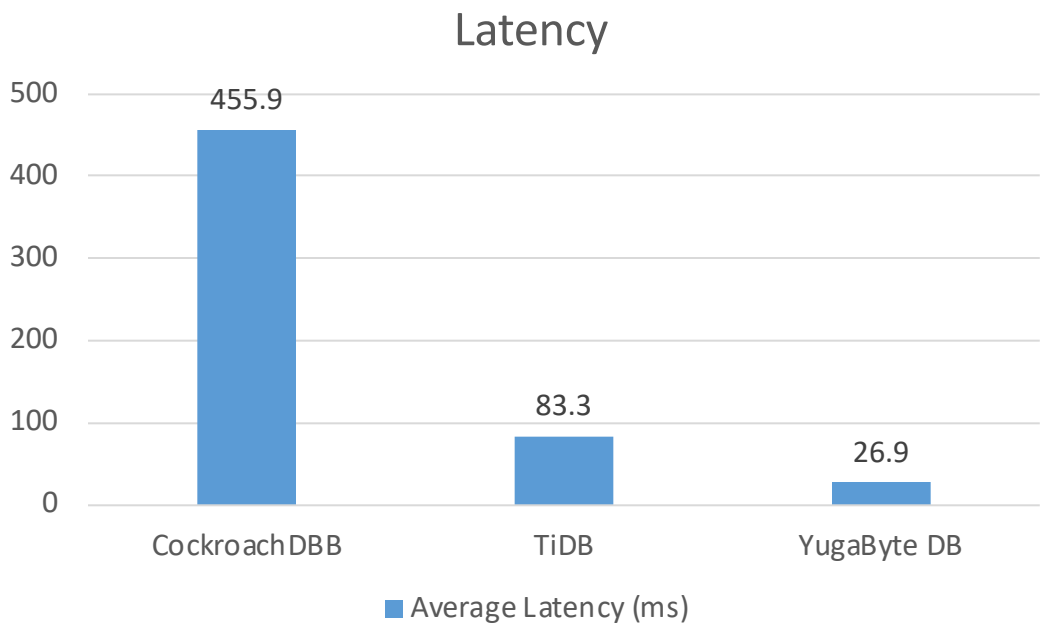
Workload A - Read



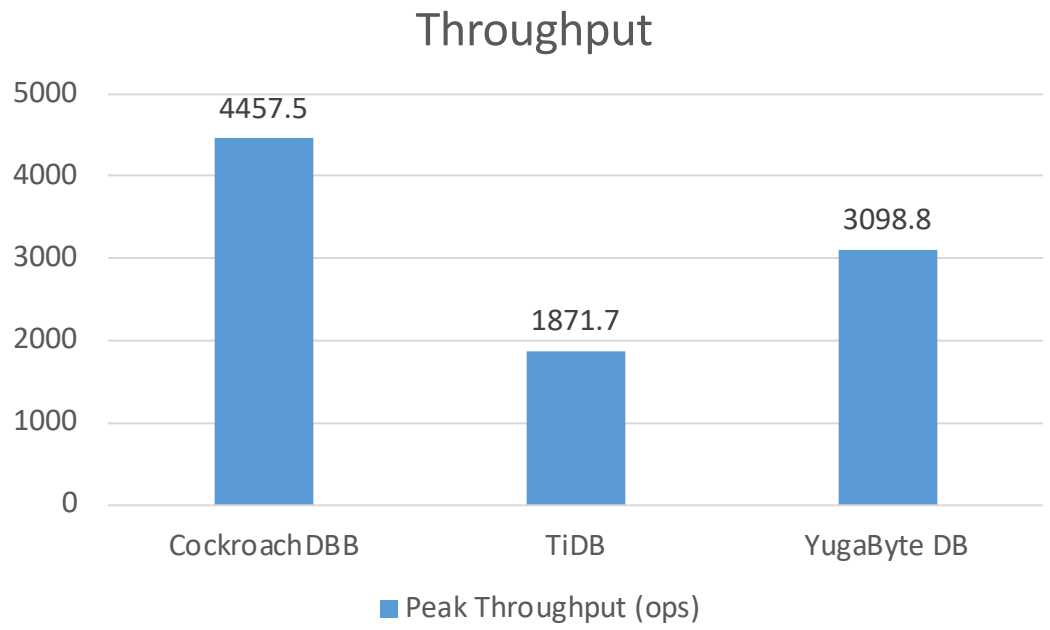


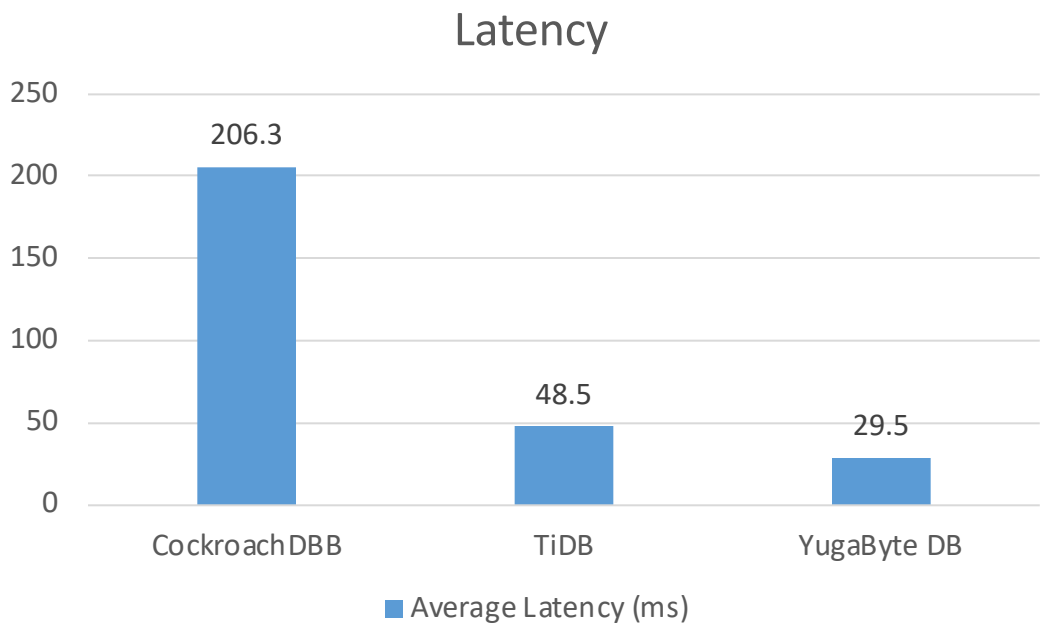
Workload A - Update



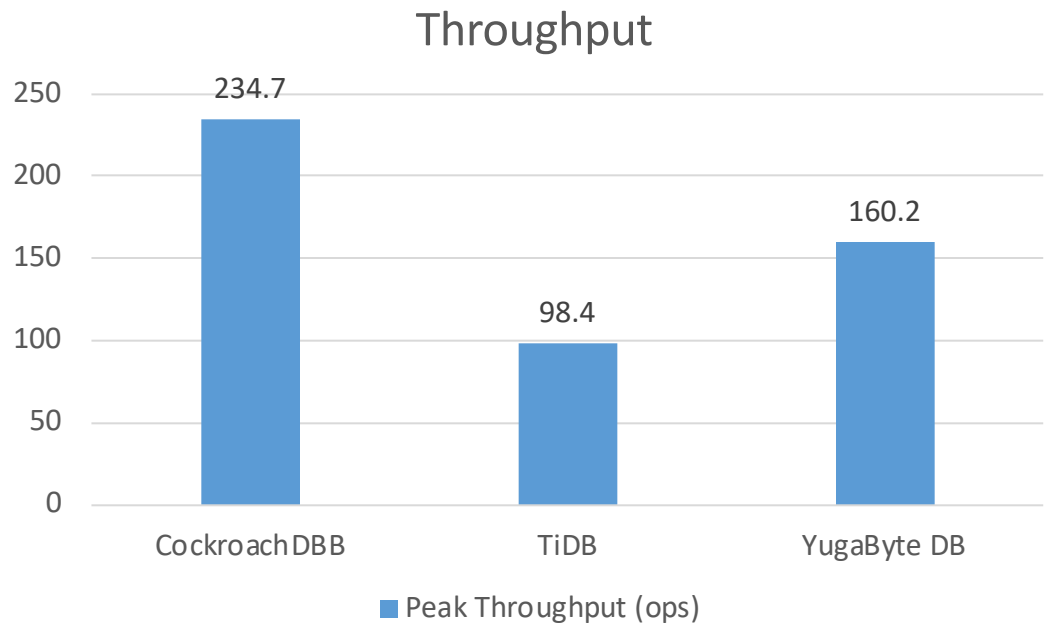


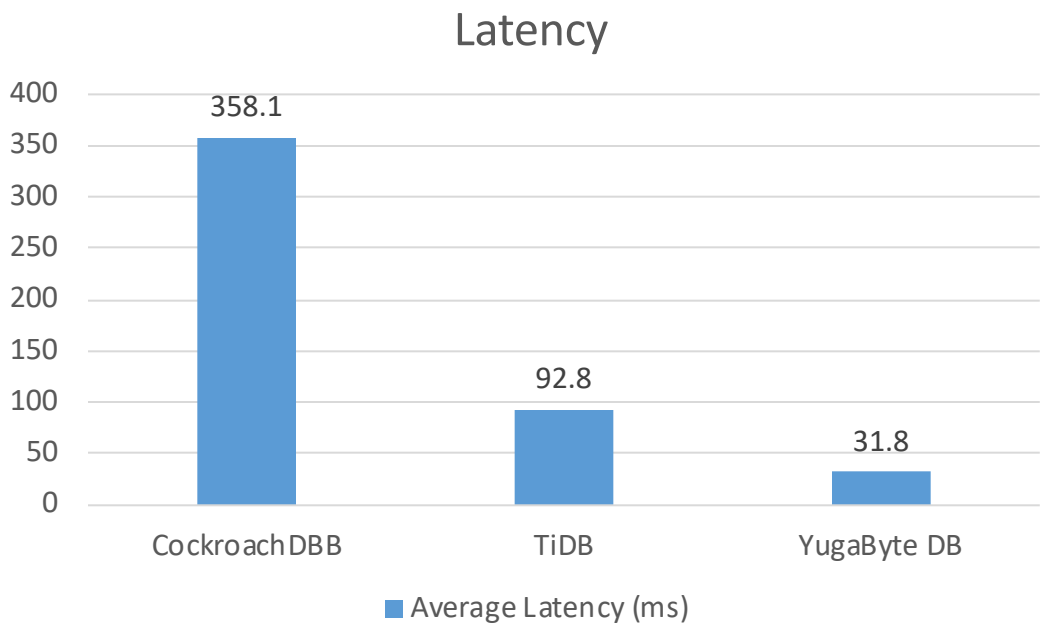
Workload B - Read



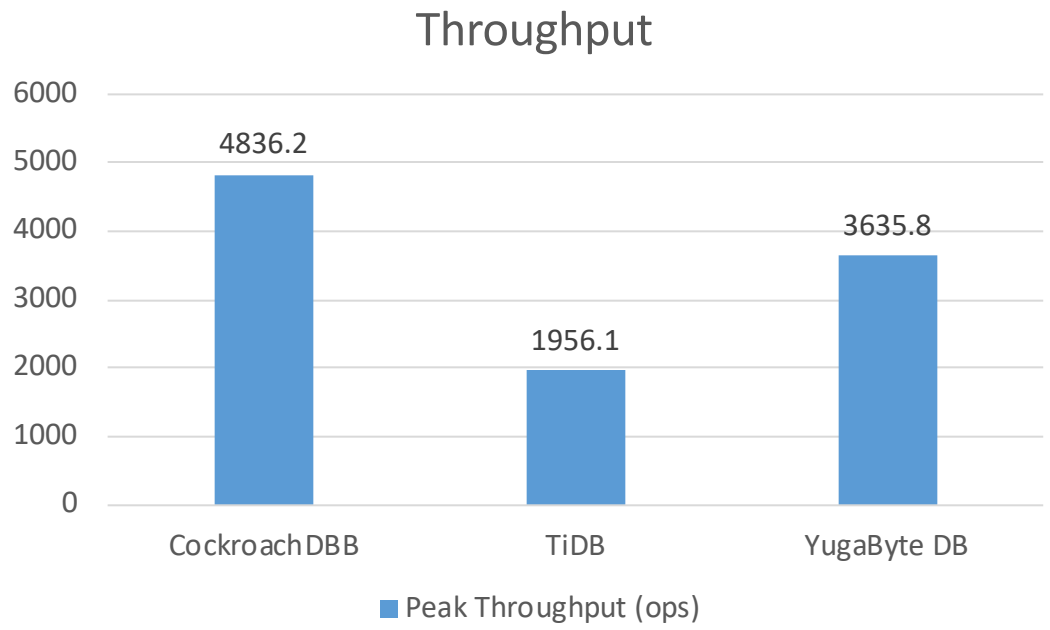


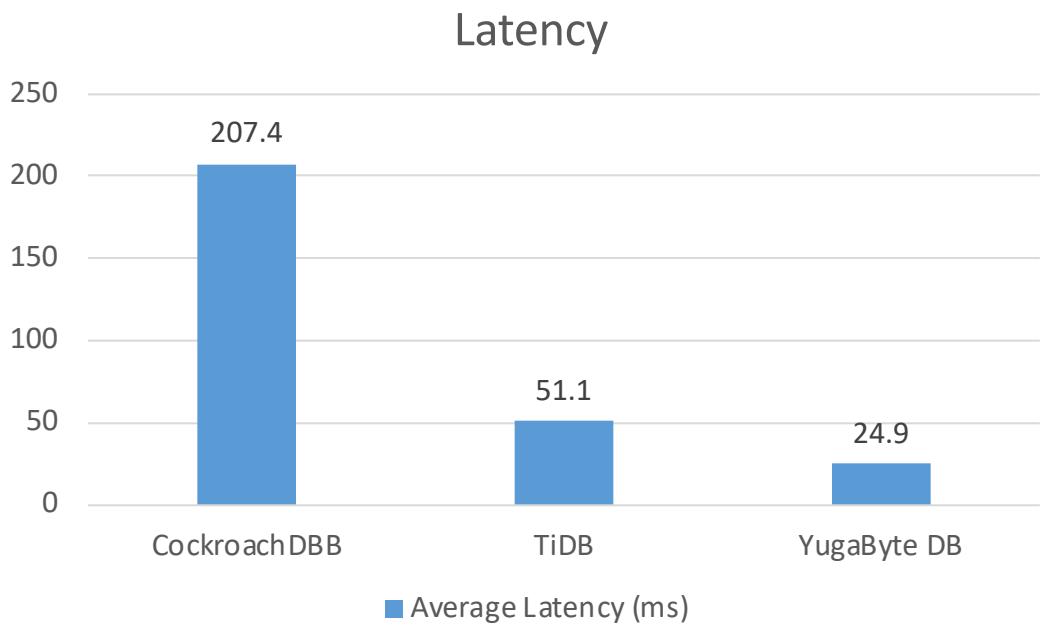
Workload B - Update



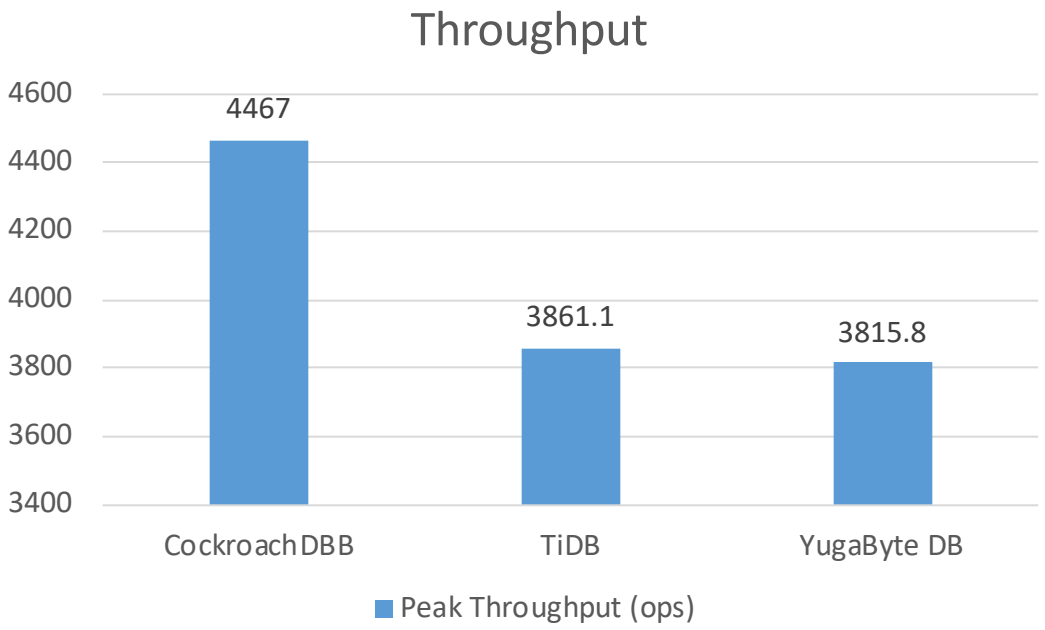


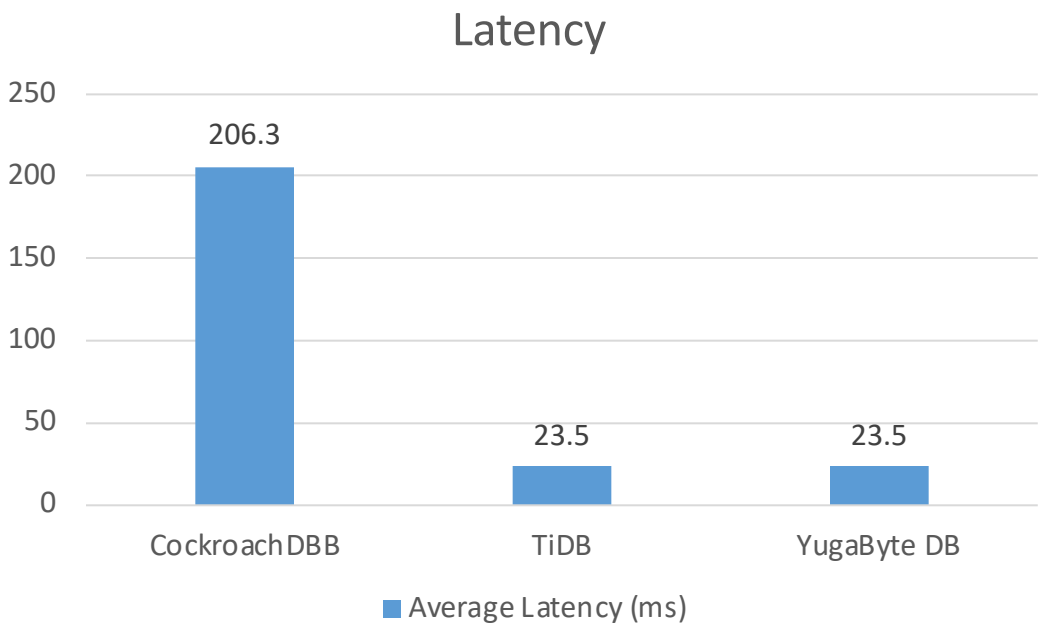
Workload C - Read



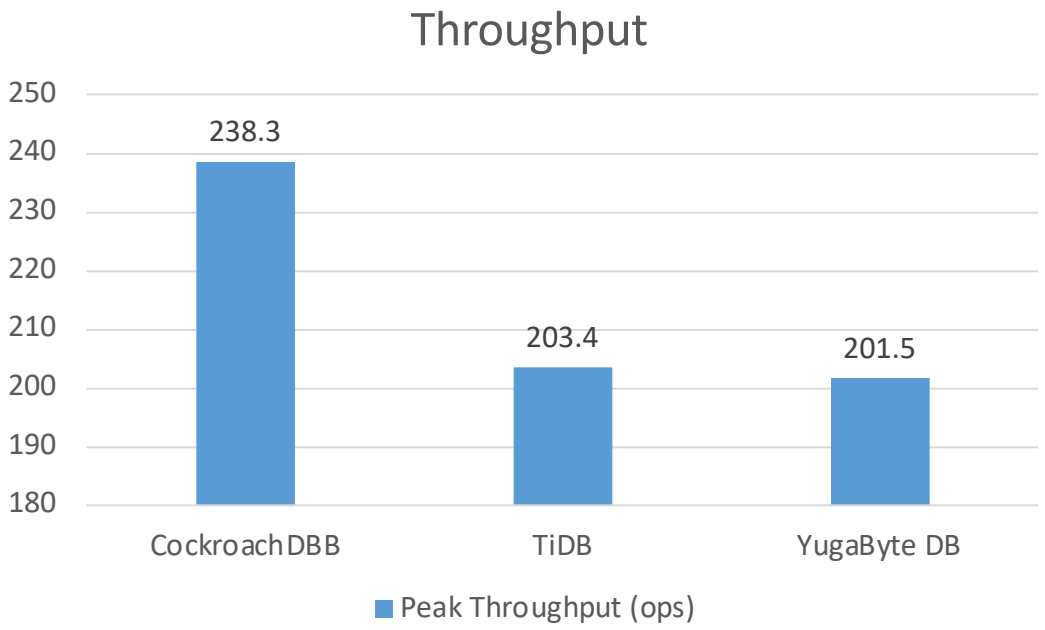


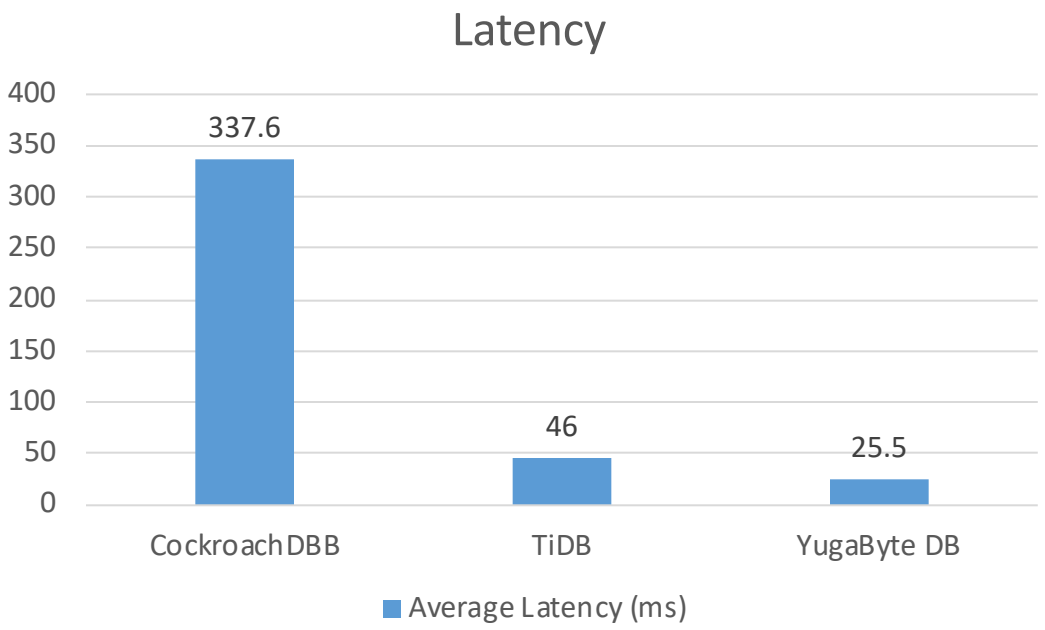
Workload D - Read



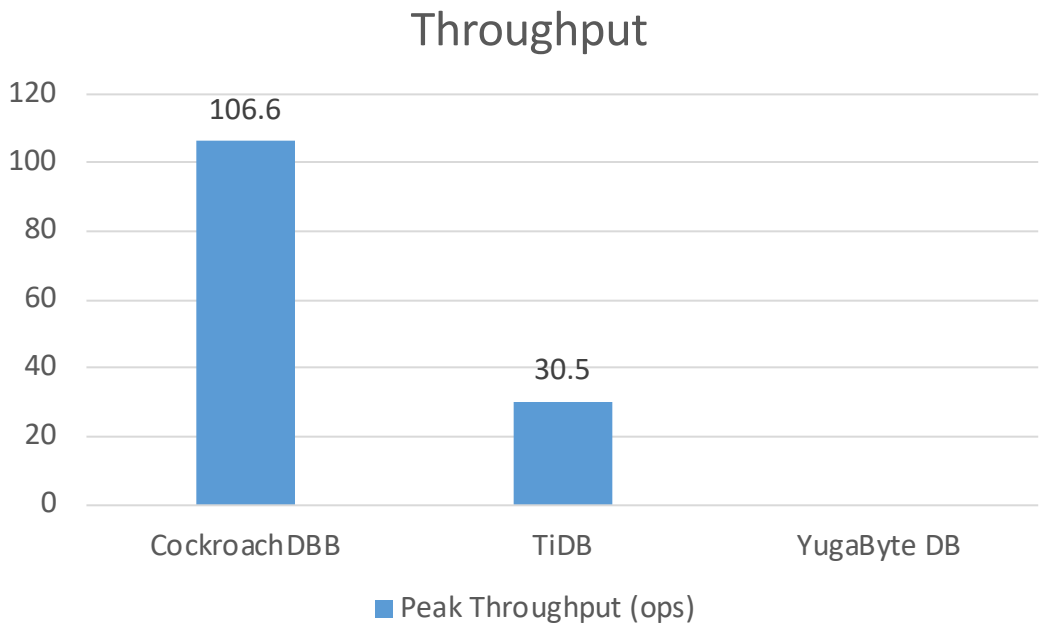


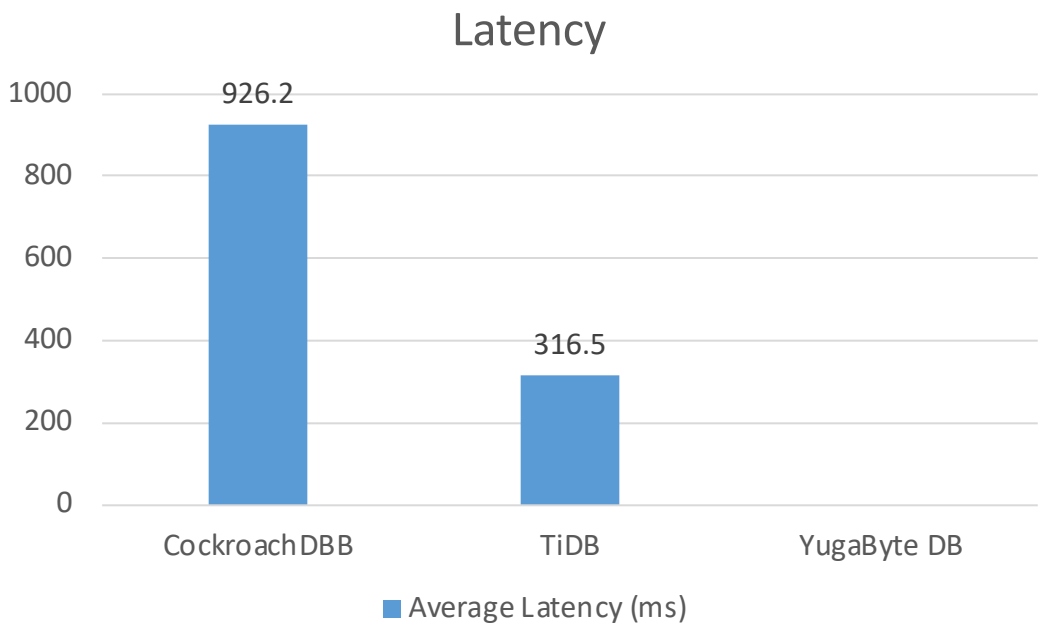
Workload D - Insert



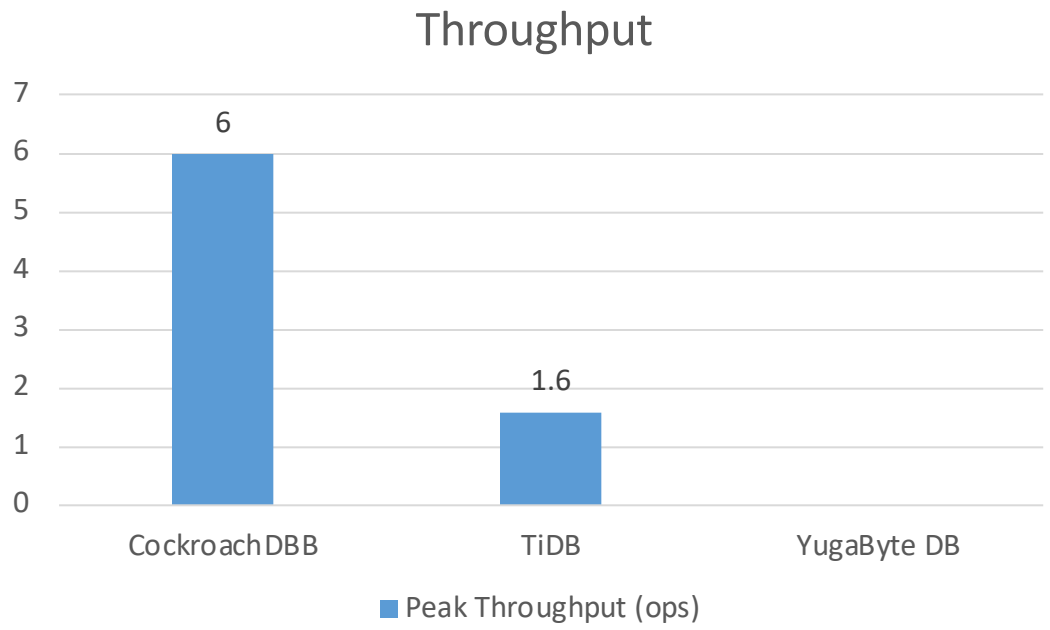


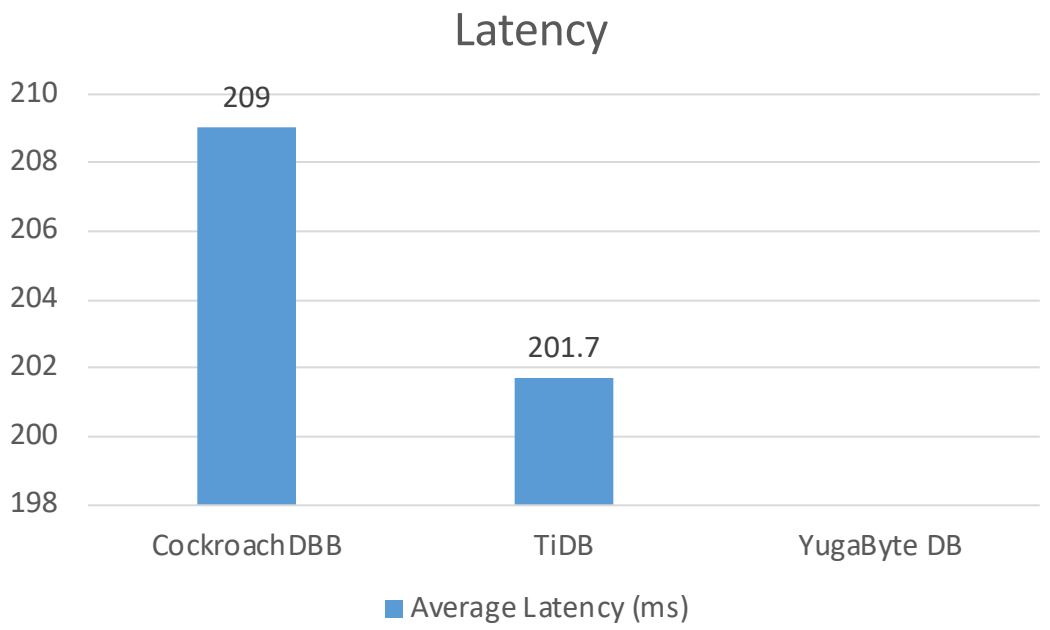
Workload E - Scan





Workload E - Insert





Discussions (1)

- All databases do not perform as well as they would if they run on a dedicated VM cluster with the same spec.
- All databases perform very well on read operations, especially in read-heavy workload (Workload B and Workload C).
- All databases perform fairly well on update operations in update-heavy workload (Workload A) but not so much in read-heavy workload (Workload B).

Discussions (2)

- All databases perform fairly well on insert operations in read-heavy workload (Workload D), but perform poorly in scan-heavy workload (Workload E).
- All databases does not perform well in scan operations in scan-heavy workload (Workload E).

Further Study

For further study, we recommend the following:

- Figure out the bottleneck that prevent databases to perform as well as they do in dedicated VM cluster instead of on top of Kubernetes cluster
- Communicate closely with engineers from respective database to gain more insights on how to fine tune each database

GO JEK

Q & A

KubeCon | CloudNativeCon

Europe 2019

Gracias!

KubeCon CloudNativeCon Europe 2019