Scale it easy: YDB's high performance in a nutshell

Evgenii Ivanov

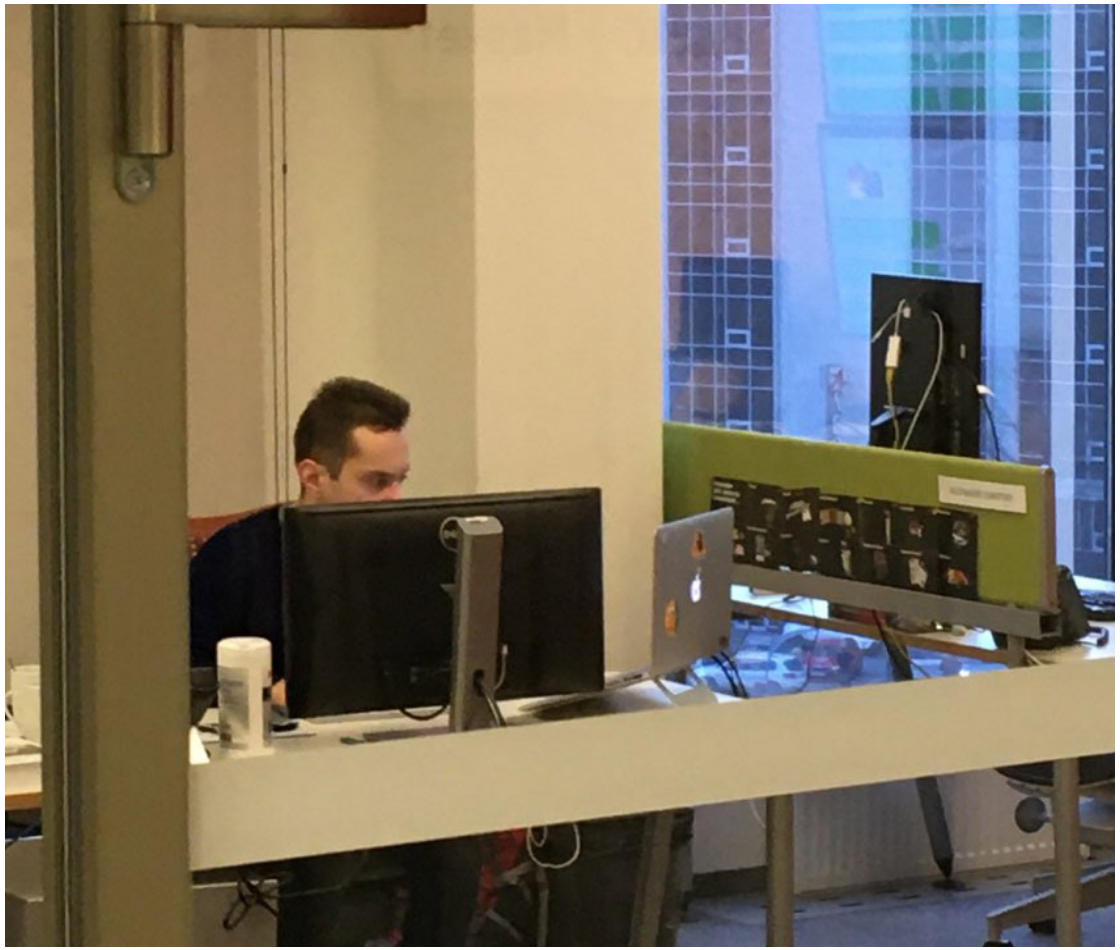
Senior software engineer, Yandex

Co-organizer

Yandex

About myself

- YDB developer
- Outside YDB I enjoy aerial photography, spending time with my family and reading



Costs of a wrong DB choice

- Customer's data loss
- Increased expenses
- Low performance
- Inability to scale

Goals of this talk

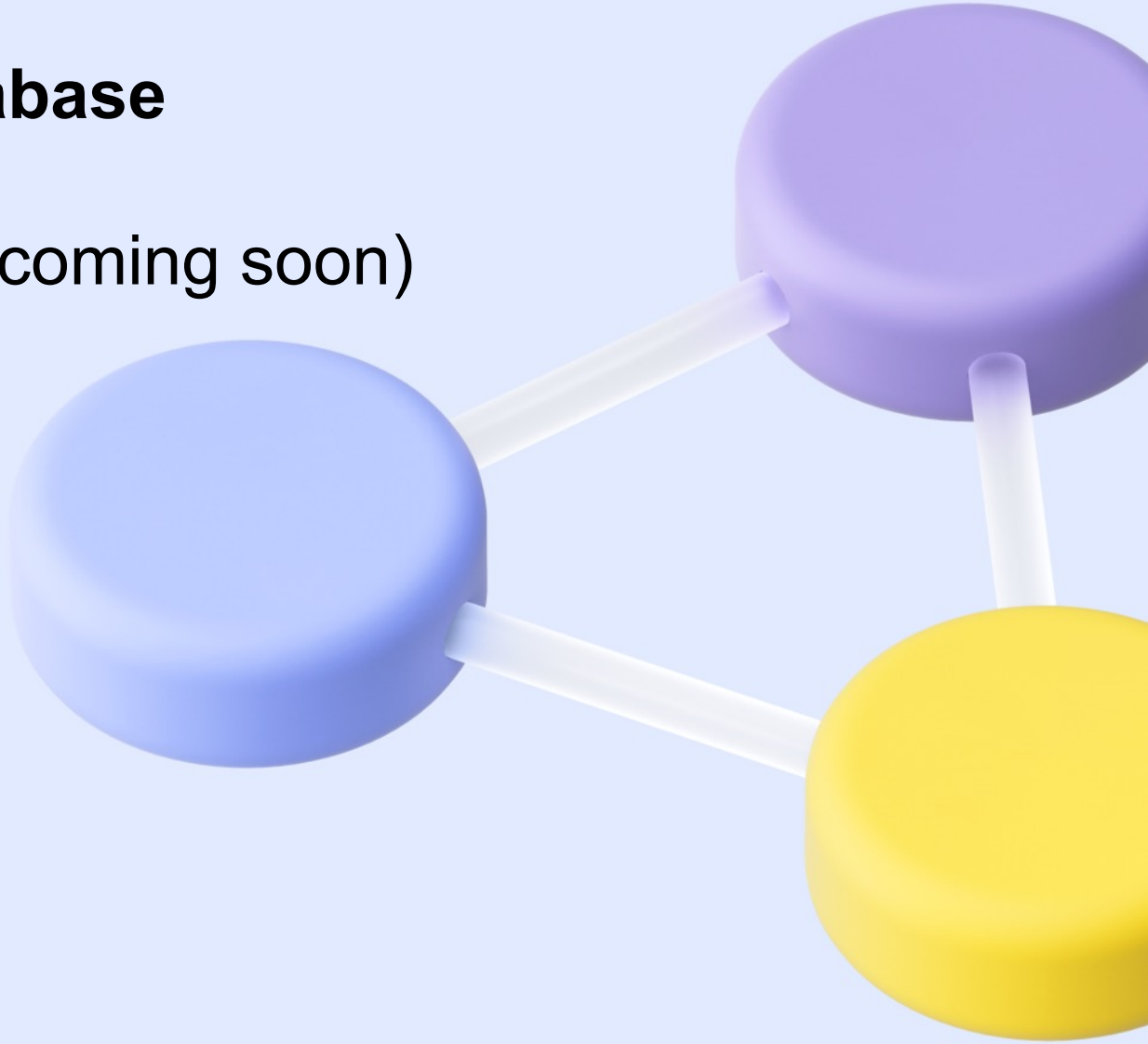
- Demonstrate the high availability and superior performance of distributed databases
- Illustrate scenarios where YDB is the optimal choice for your data management needs
- Delve into intriguing technical aspects underpinning YDB's high performance

YDB

Open-Source Distributed SQL Database

- Relational DB (mainly OLTP, OLAP coming soon)
- Clusters with thousands of servers

- Apache 2.0 license
- Star ydb-platform on GitHub



YDB is …

Strictly consistent

- CAP-theorem – we choose CP
- Serializable transaction execution

YDB is …

Highly available and fault tolerant

- Multiple availability zones (AZ): automatic recovery
- YDB is read-write available even after losing AZ and a rack simultaneously

YDB is …

A mission critical database

- 365x24x7 (366x24x7 when needed)
- Zero maintenance window

What is performance?

- Throughput
- Latency

Usually at YDB we are focusing on achieving max possible throughput with 50 ms latency limit on 99 percentile

Performance has no goal, only path

- New hardware
- New algorithms
- New challenges and applications
- Never ending improvements

Performance saves money

- The better DB performance is – the less hardware you need
- With the high availability and scalability you simply earn more

01

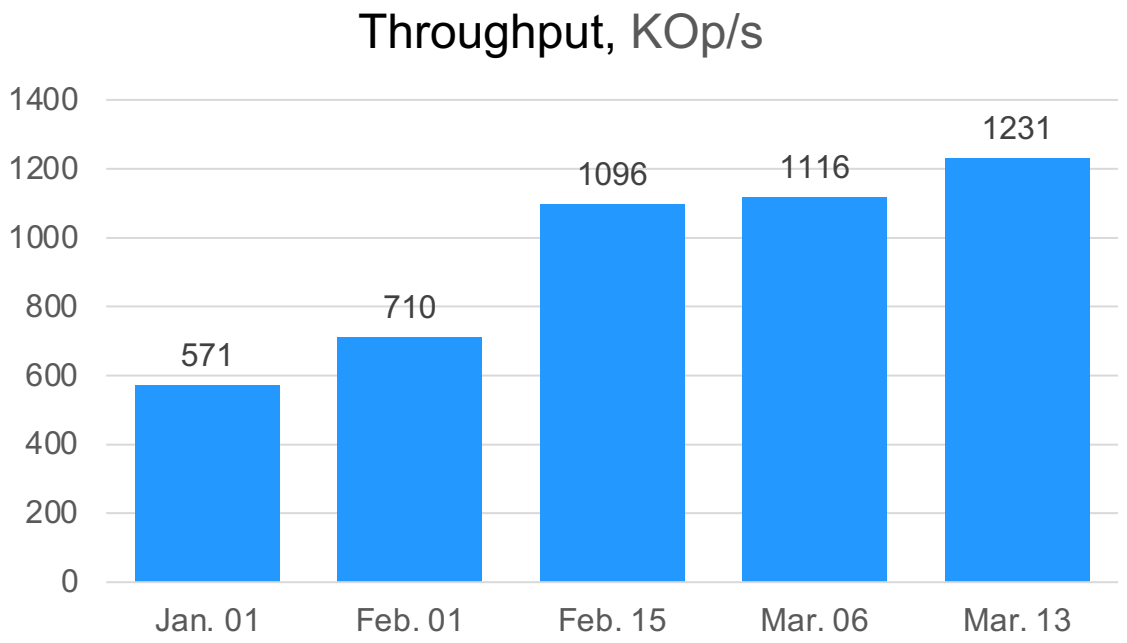
Performance tips and tricks for everybody

Co-organizer

Yandex

The source of tips is our path

- There are a lot of architectural decisions
- Some system configuration tweaks
- Anti bottlenecks campaign
- Lots of flamegraph sessions and experiments



Whoa, can we still process more?

Yes! CPU affinity

- No code changes – same binary
- Applicable to any multithreaded software and any DB

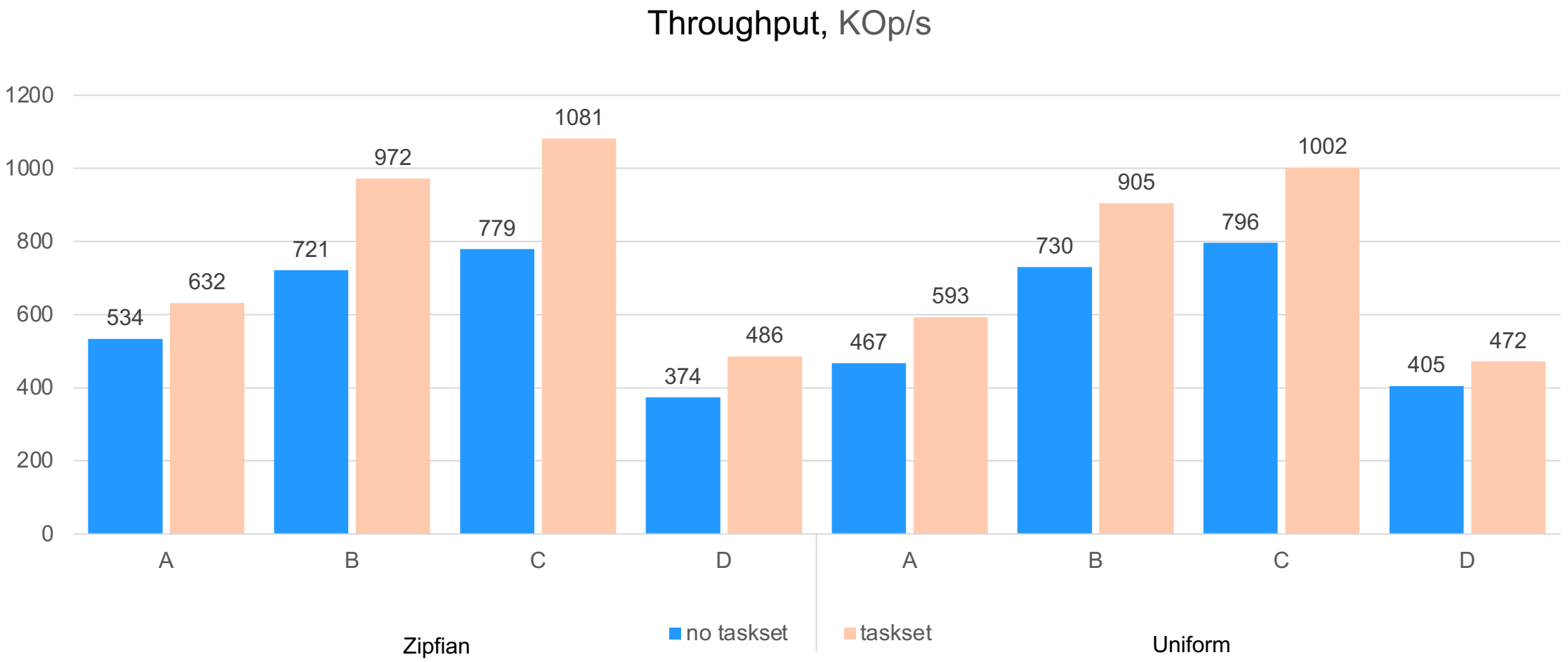
Yes! CPU affinity

- No code changes – same binary
- Applicable to any multithreaded software and any DB
- CPU caches are crucial for performance
- Important metric: instructions per cycle (IPC)

Yes! CPU affinity

- No code changes – same binary
- Applicable to any multithreaded software and any DB
- CPU caches are crucial for performance
- Important metric: instructions per cycle (IPC)
- We want to pin threads to a subset of cores
- Solution: CPU affinity (cpusets/tasksets)
- Throughput increases by 20-40% with same CPU consumption (proportionally to IPC)

Yes! CPU affinity



Hugepages

- Cache misses are expensive
- TLB misses are even more expensive

Hugepages

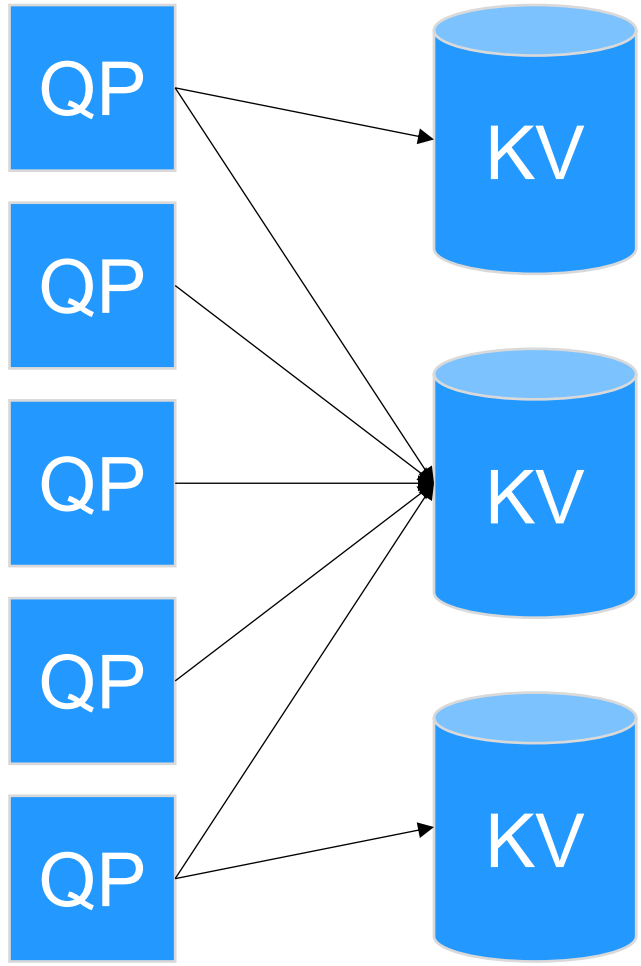
- Cache misses are expensive
- TLB misses are even more expensive
- Solution: transparent hugepages
- +3% of free throughput

Actor System

- Is a concurrency model
- Actor – the single threaded entity with own state
- Actors receive messages, send messages and create other actors

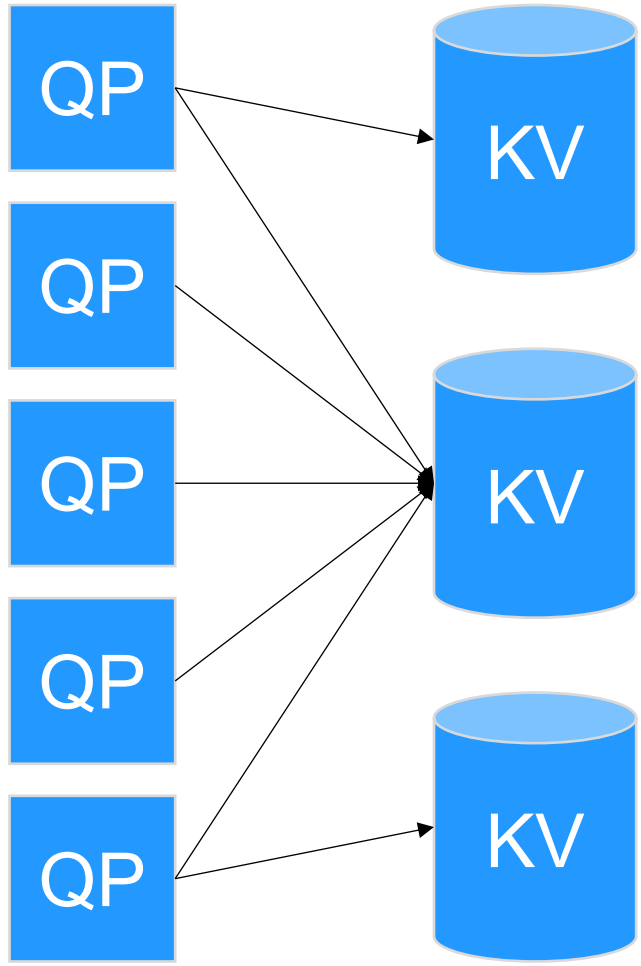
Latency bound antipattern

- QP – query processor
- KV – key-value storage
- QP and KV are single threaded
- Queries might be complex
- Push-down is a traditional approach



Latency bound antipattern

- Load distribution is uneven
- Pareto principle (the Zipfian distribution)
- Requests to KV are latency bound



Some compiler magic

- Profile-guided optimization (PGO): +20-30% of throughput
- BOLT: post-link binary optimizer: +3-5% of throughput
- Link time optimization (LTO): +1-3% of throughput

Architecture, platform and features

- Separate compute and storage layers
- Own storage layer directly on disks (talk)
- Own advanced actor system (talk)
- C++
- MVCC (talk) and other features
- Beyond Calvin

02

YCSB: key-value performance

Co-organizer

Yandex

Yahoo! Cloud Serving Benchmark (YCSB)

- A popular key-value benchmark
- Created for NoSQL key-value DBs, but still loved by everybody
- Supports almost all modern databases
- It’s hard to do distributed transactions well if you can’t do key-value workloads well

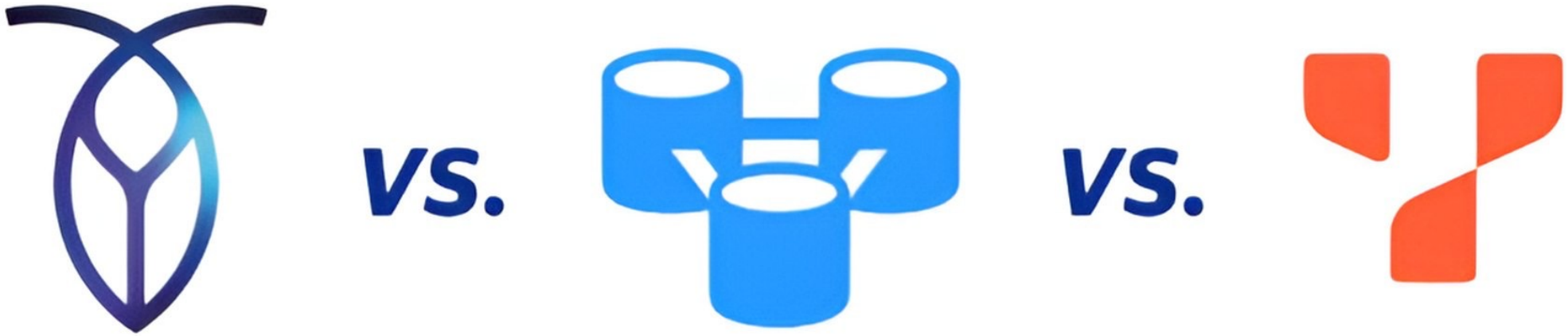
YCSB workloads

- A (update heavy workload): 50% reads and 50% updates
- B (read mostly workload): 95% reads and 5% updates
- C (read only)
- D (read latest workload): 95% reads, 5% inserts
- F (read-modify-write): 50% reads and 50% read-update operations
- E (short ranges): 95% scans and 5% inserts.

Test setup

- 128 cores: 2x32-cores Intel Xeon Gold 6338 CPU @ 2.00GHz with hyper-threading turned on
- 4xNVMe
- 512 GB RAM
- 50 Gb network
- Transparent hugepages turned on
- Ubuntu 20.04.3 LTS

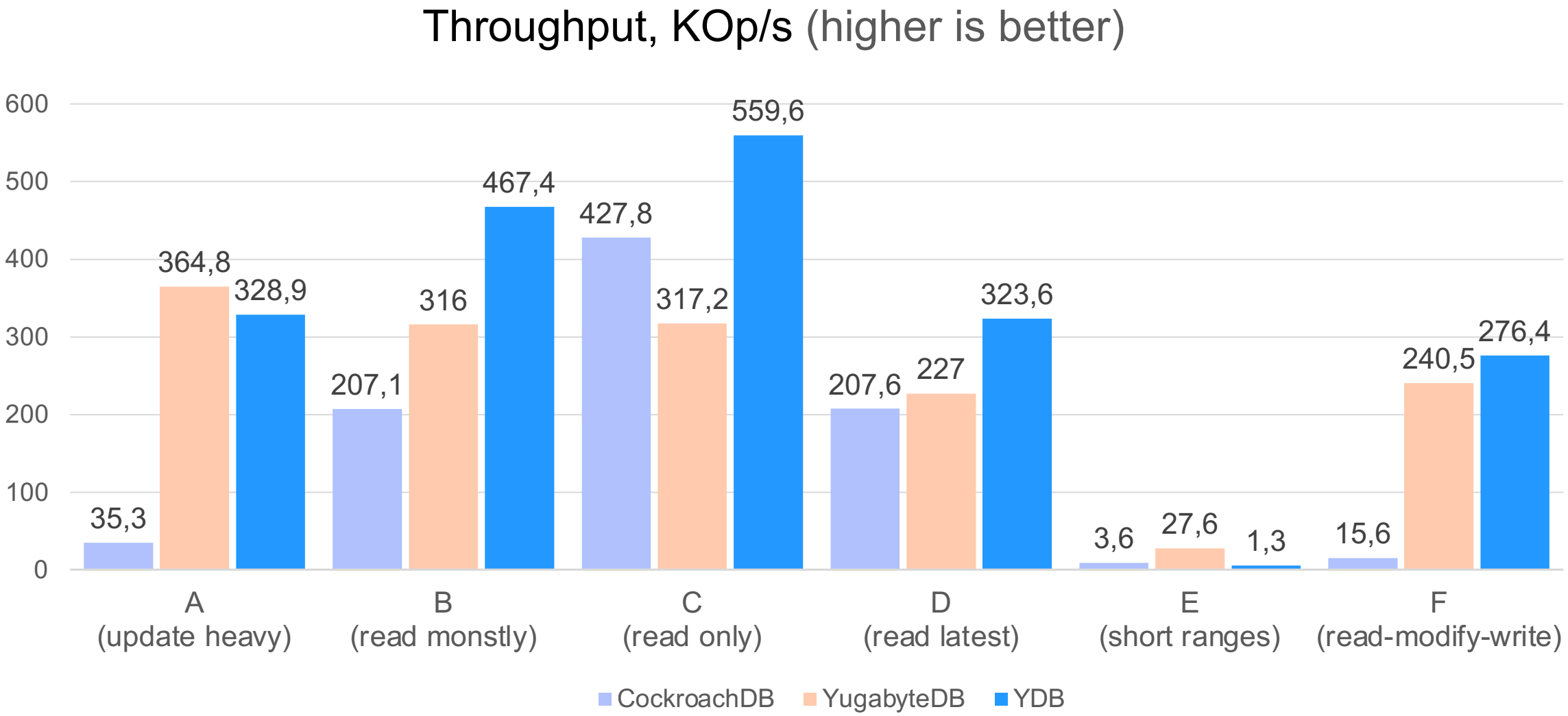
Distributed SQL Databases



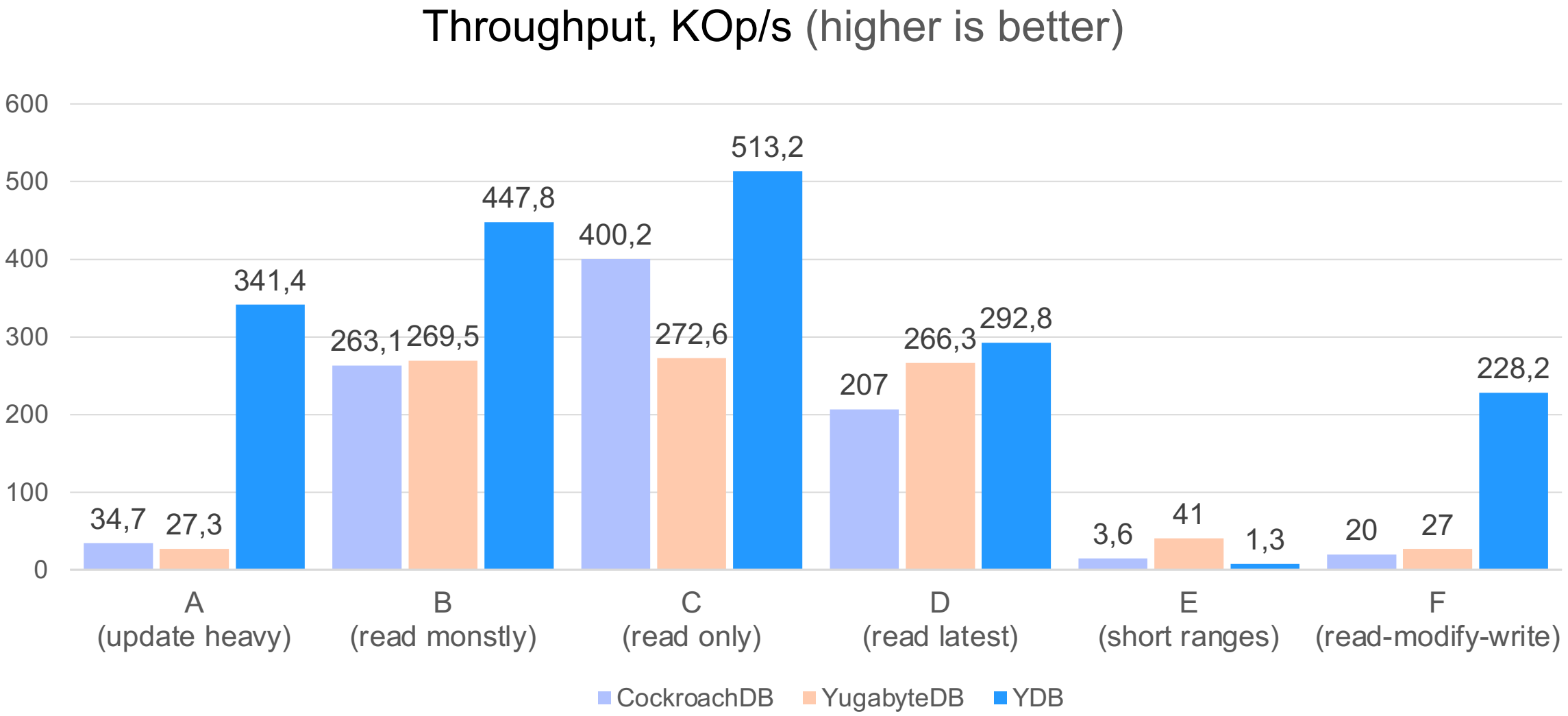
A bar fight?

Like this?

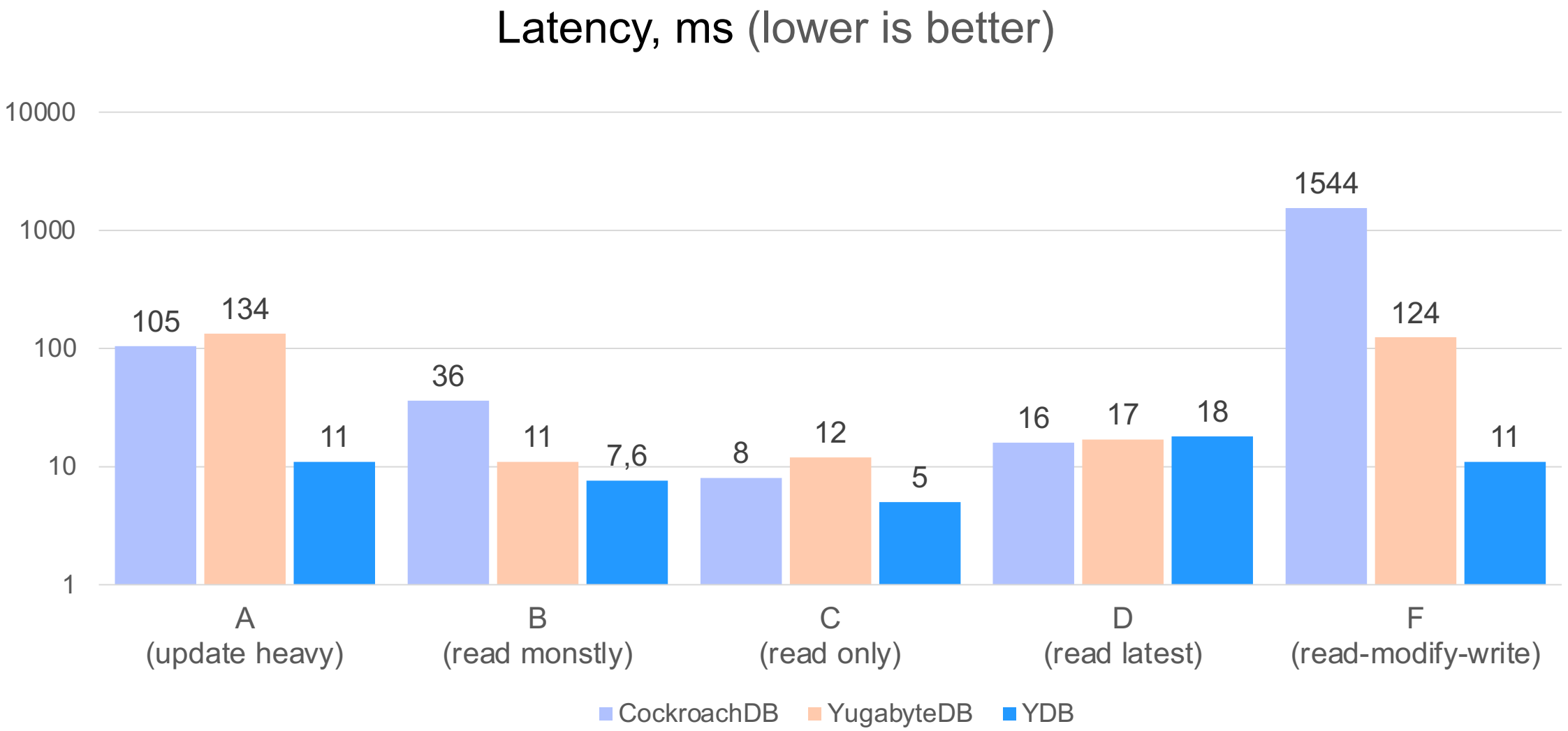
YCSB: 300 GB data



YCSB: 2 TB data



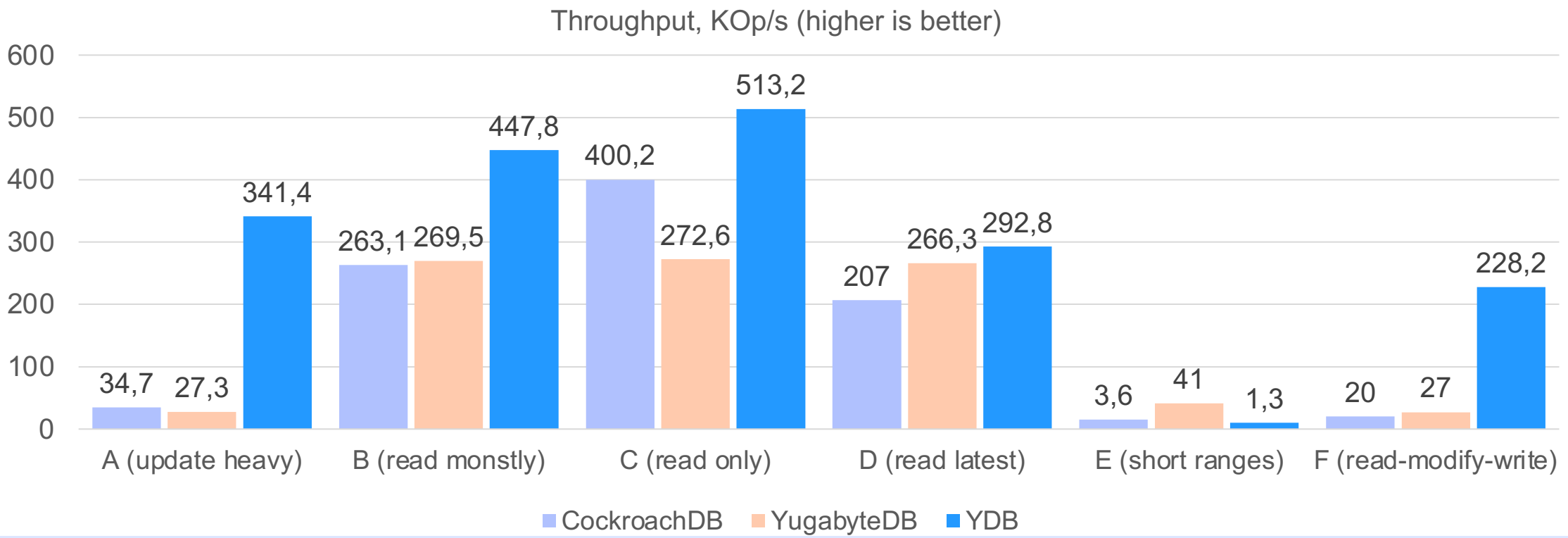
YCSB: 2 TB data



A silver bullet?

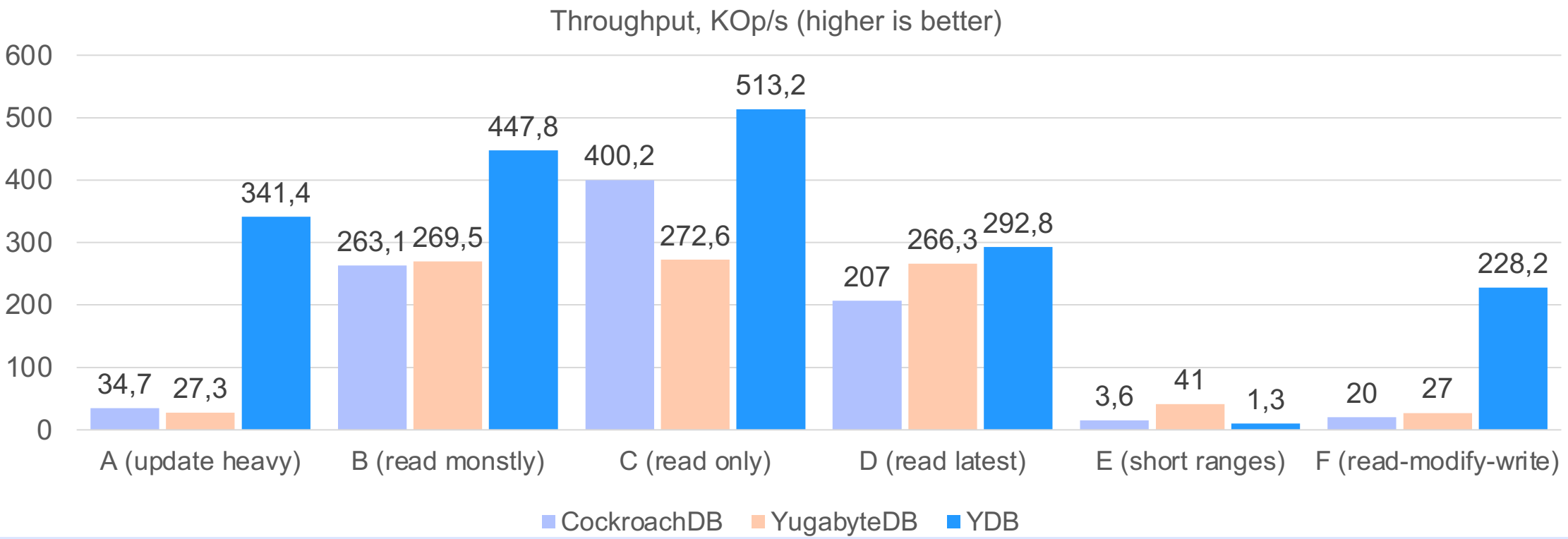
Analyses

- UPSERT vs. UPDATE: blind write VS. read-modify transaction
- CockroachDB uses UPDATE in workloads A, B and F



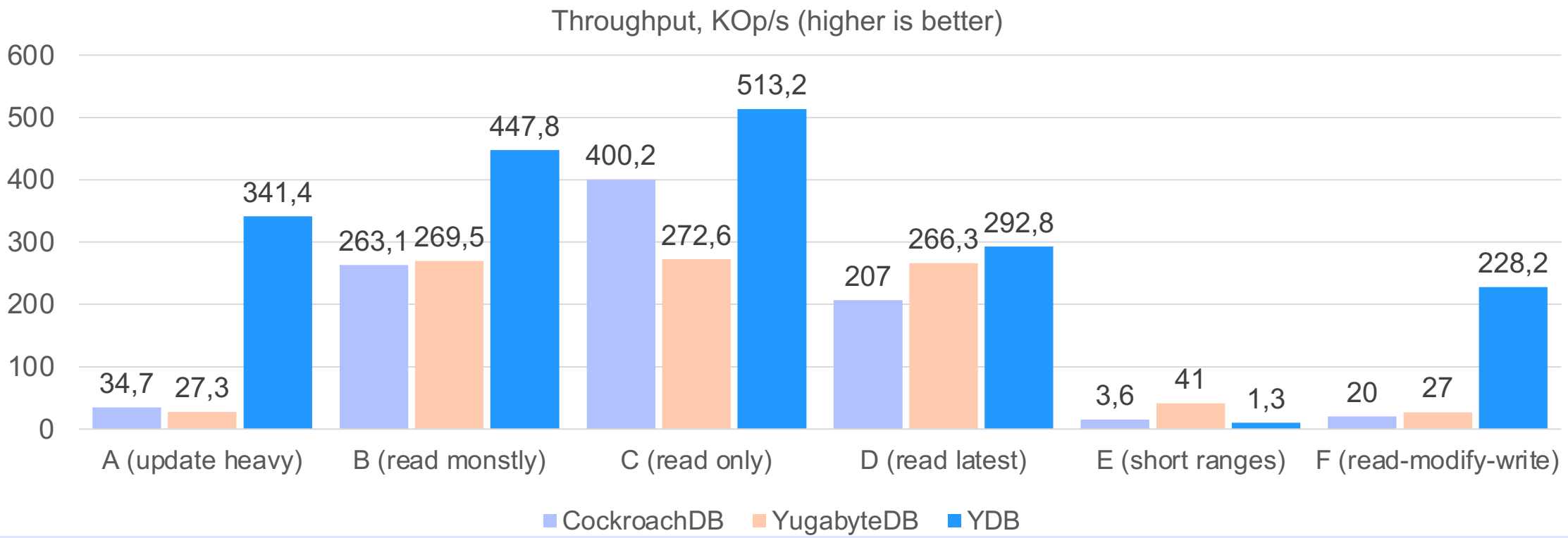
Analyses

- YugabyteDB has a bare metal horizontal scaling issue
- YugabyteDB has a vertical scaling issue



Analyses

- Workload C is the same among all DBs
- YDB has an issue with short ranges (workload E)



YCSB conclusion

- Be careful when you implement your application
- YDB outperforms others in many YCSB workloads
- In write intensive workloads all 3 DBs exhibited decent performance
- We strongly recommend YDB for read-mostly and read-only workloads

Full YCSB results: https://bit.ly/3WfMZEq

03

Distributed SQL vs “Centralized” SQL

Co-organizer

Yandex

Other DBs to compare with?

Ask your DB a question

- What will happen if a disk/server/datacenter fails?
- How can I reduce the latency if users are far from their DC?
- What latency will I have when working set exceeds RAM (typically 512 GiB, rarely above 1024 GiB)?

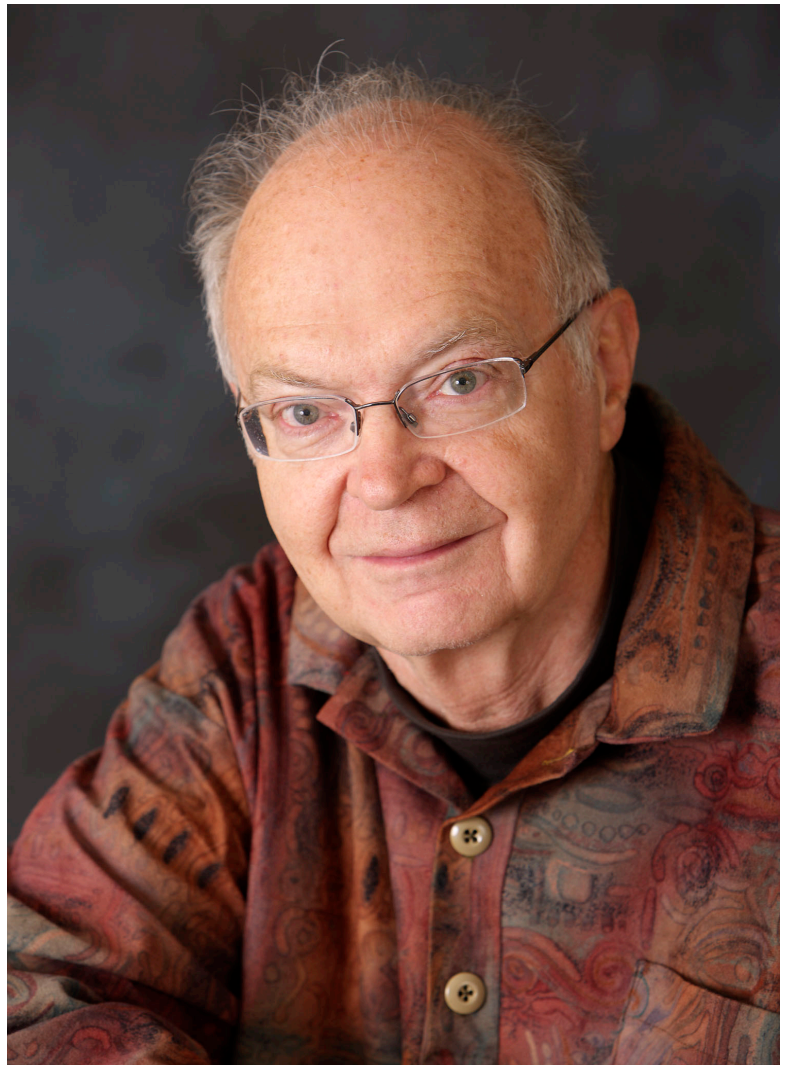
Big data questions to DB

- Can you store a petabyte of data?
- Can you handle more than 100 GB/s of data updates?

“The best theory is inspired by practice. The best practice is inspired by theory.”

Donald Knuth

- There are theoretical reasons behind the advantages of a distributed DB
- Distributed DB theory is based on the best practice



What do we expect from DBs?

- ACID – of course!
- Scalability
- Maintainability
- High performance
- Security
- Reliability / fault tolerance / availability

Nothing is reliable

- Software bugs
- Hardware bugs
- Hardware faults
- Unreliable networks

Can a single server DB be reliable and available? No.

Nothing is reliable

Is replication a solution?

Replication pitfalls

Async replication:

- risk of data loss (not durable)
- stale reads and anomalies

Avoid async replication until you know what you’re doing!

Sync replication pitfalls

Sync replication is better, but still there is one single leader:

- only reads can be split among replicas
- load balancer (like HAProxy)
- failover and failback (like Patroni)
- failover requires consensus: etcd, ZooKeeper, etc
- when failed node is back, recovery might take a long time

Vertical and horizontal scaling

IBM z16 mainframe

A small part of typical YDB cluster

Sharding

- Horizontal scaling
- Geographical scaling
- Many leader shards (partitions/tablets)

Sharding pitfalls

- Consistency – distributed snapshots – complicated!
- You need a coordinator like Citus as well as standby and failover procedure for the coordinator

The choice

Do you really want to maintain a DB with a separate

- load balancer
- consensus cluster
- failover and failback
- coordinator

Conclusion

Co-organizer

Yandex

Conclusions

- There are ways to ruin performance on the application level

Conclusions

- There are ways to ruin performance on the application level
- ”Centralized” SQL doesn’t match mission critical data
- Distributed DB is the proper solution for availability and scaling challenges

Conclusions

- There are ways to ruin performance on the application level
- ”Centralized” SQL doesn’t match mission critical data
- Distributed DB is the proper solution for availability and scaling challenges
- There are scenarios, where YDB outperforms other distributed DBs
- You might want to consider YDB when choosing distributed DB for your data

Please leave your feedback

Yandex on the conference

Evgenii Ivanov (tg, twitter, linkedin: @eivanov89)

Senior software developer, Yandex

Slides: bit.ly/3XufHIr

YDB

High Load ++ Serbia

Co-organizer

Yandex