

DATA @ scale



The Story of RocksDB

Embedded Key-Value Store for Flash and RAM

Dhruba Borthakur
Database Engineering




Dhruba Borthakur
Edit Profile

Update Status Add Photos/Video

What's on your mind?

Sort

Kai Liu
Ninja turtle completed! — with Wenjing Yu at Facebook HQ.



Like · Comment · Share · Yesterday at 3:46pm in Menlo Park · 14

Rongrong Zhang, Dheeraj Kumar Singh, Volodymyr Krestianynkov and 8 others like this.

Michel Tu It looks like you both are working hard 😊

Waiyan Wang's birthday is today

Sponsored

Sameet Agarwal and Manish Modi like Mercury Trip.

Mercury Trip
Like

Join Orbitz Rewards
earn immediately, redeem insta...

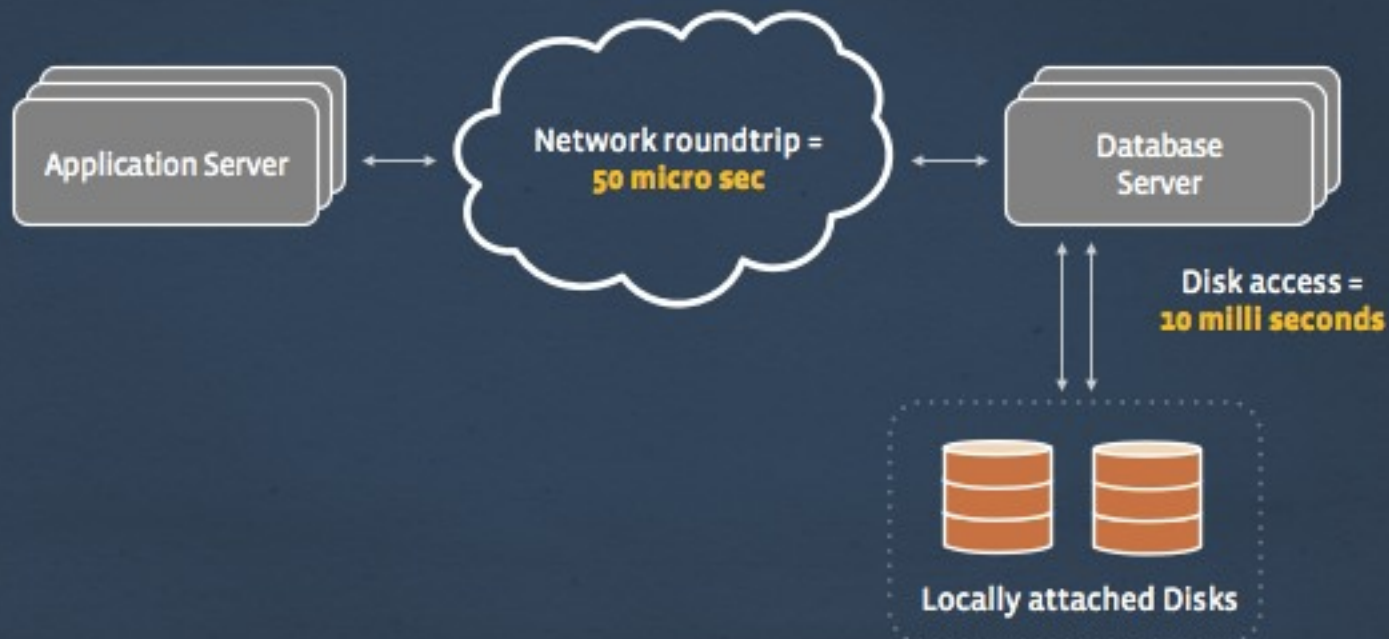
It's on us! Get a promo code for 15% off hotel when you join the N Orbitz Rewards.

Save at Lowe's with Amex
Home improvement on your to do list? Spend \$50 and get \$10 back 1x at Lowe's. See how.

Exclusive Customer Offer
att.com

AT&T wireless customers save more! Get U-verse TV Internet & Home Phone - \$74/mo for

A Client-Server Architecture with disks

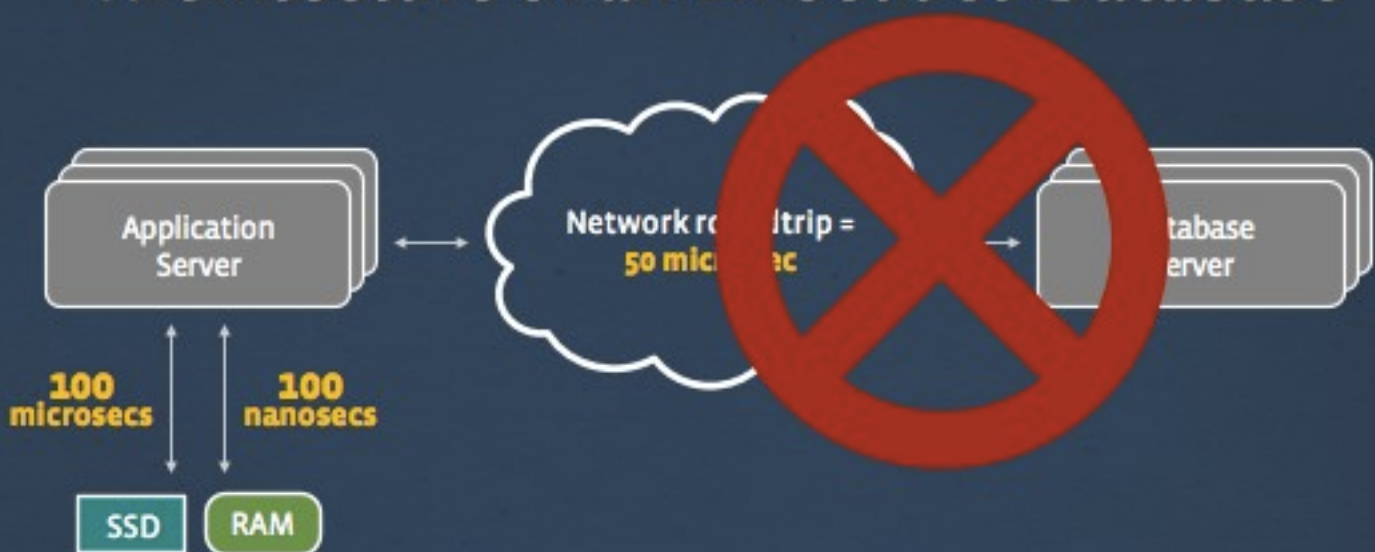


Client-Server Architecture with fast storage



Latency dominated by network

Architecture of an Embedded Database



Storage attached directly to application servers

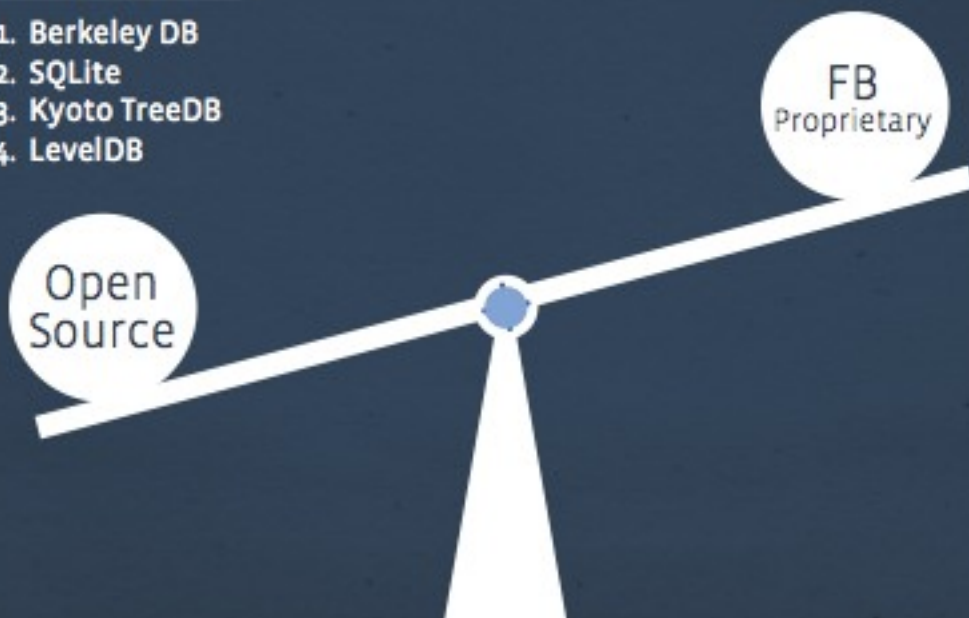
Any pre-existing embedded databases?



Any pre-existing embedded databases?

Key-value stores

1. Berkeley DB
2. SQLite
3. Kyoto TreeDB
4. LevelDB

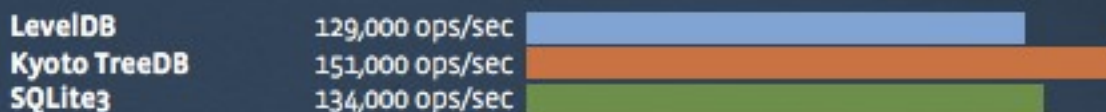


Any pre-existing embedded databases?

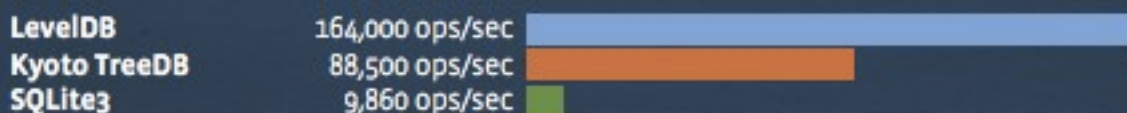


Comparison of open source databases

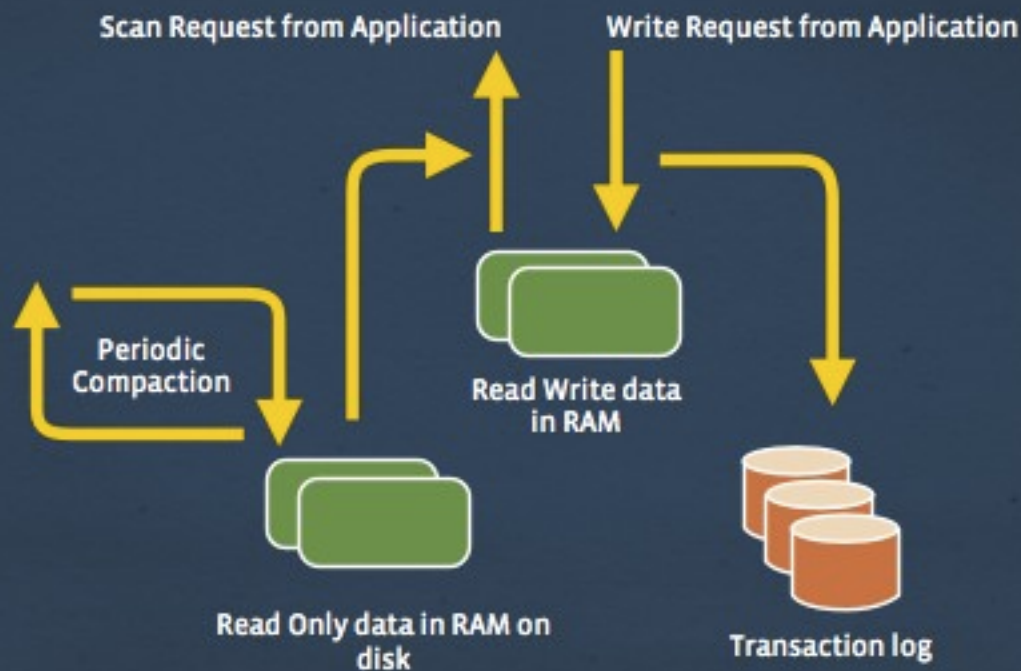
Random Reads



Random Writes



Log Structured Merge Architecture



Leveldb has low write rates

Facebook Application 1:

- Write rate 2 MB/sec only per machine
- Only one cpu was used

We developed multithreaded compaction



Leveldb has stalls

Facebook Application 2:

- P99 latencies were tens of seconds
- Single-threaded flush, conflict with compaction



We implemented thread aware compaction

Dedicated thread(s)
to flush memtable

Pipelined memtables

P99 reduced to less
than a second

Leveldb has high write amplification

• Facebook Application 2:

- Level Style Compaction
- Write amplification of 70 very high



Two compactions by LevelDB Style Compaction

Our solution: lower write amplification

- Facebook Application 2:

- We implemented Universal Style Compaction
- Start from newest file, include next file in candidate set if
 - Candidate set size \geq size of next file



Write amplification reduced to **<10**

Leveldb has high read amplification

- Secondary Index Service:
- Leveldb does not use blooms for scans
- We implemented prefix scans
 - Range scans within same key prefix
 - Blooms created for prefix
 - Reduces read amplification



Leveldb: read modify write = 2X IOs

- Counter increments
 - Get value, value++, Put value
 - Leveldb uses 2X IOPS
- We implemented MergeRecord
 - Put “++” operation in MergeRecord
 - Background compaction merges all MergeRecords
 - Uses only 1X IOPS

Leveldb has a Rigid Design

- LevelDB Design
 - Cannot tune system, fixed file sizes
- We wanted a pluggable architecture
 - Pluggable compaction filter, e.g. TimeToLive
 - Pluggable memtable/sstable for RAM/Flash
 - Pluggable Compaction Algorithm

The Changes we did to LevelDB

1

Inherited from LevelDB

- Log Structured Merge DB
- Gets/Puts/Scans of keys
- Forward and Reverse Iteration

2

RocksDB

- 10X higher write rate
- Fewer stalls
- 7x lower write amplification
- Blooms for range scans
- Ability to avoid read-modify-write
- Optimizations for flash or RAM
- And many more...

RocksDB is born!

- Key-Value persistent store
- Embedded
- Optimized for fast storage
- Server workloads



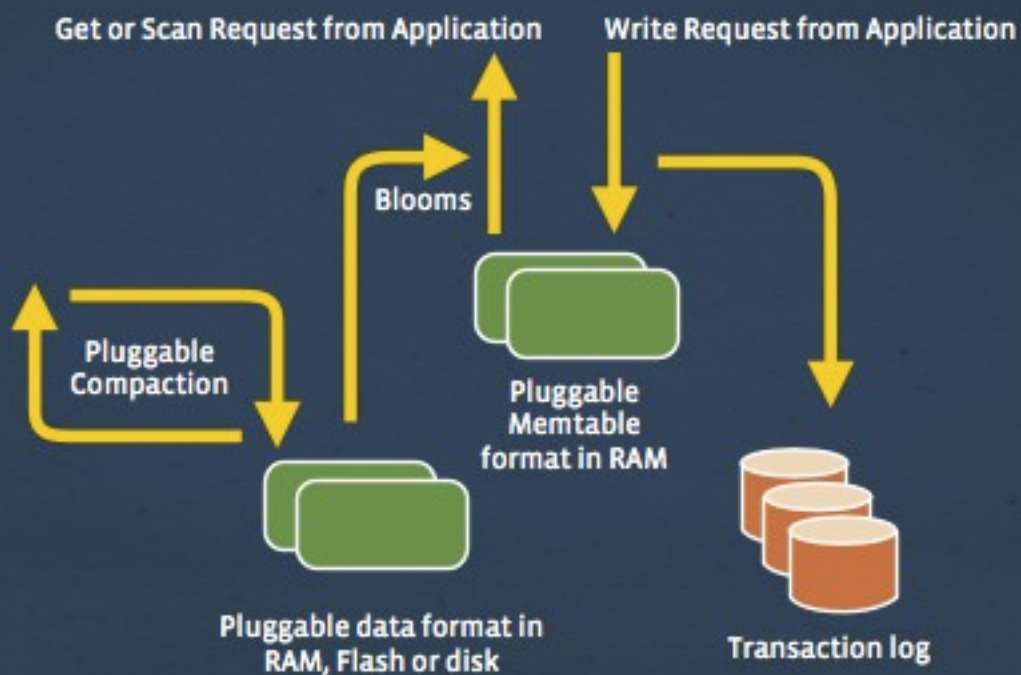
RocksDB

What is it not?

- Not distributed
- No failover
- Not highly-available, if machine dies you lose your data



RocksDB Architecture



Futures

- Scale linearly with number of cpus
 - 32, 64 or higher core machines
 - ARM processors
- Scale linearly with storage iops
 - Striped flash cards
 - RAM & NVRAM storage

Come Hack with us

- RocksDB is Open Sourced
 - <http://rocksdb.org>



- Help us HACK RocksDB



