http://martinsumner.github.io/presentations/leveled_euc#/

MY JOURNEY TOWARDS ERLANG - 2004

I'm the network guy on a huge health database project

Every problem looks like a network problem ...

Started fixing things in the application ...

... the business decided to fix things through process/management

MY JOURNEY TOWARDS ERLANG - 2011

Spine now had:

- more than 3000 servers
- more than 18 thousand people years behind it
- more than £30m in change costs ... per change
- total bill has passed £1bn

Is this the genuine cost of availability?

Lets replace it with a fundamentally different approach

MY JOURNEY TOWARDS ERLANG - 2014

Spine II Core goes live!

Better than five nines availability since go-live

Less than hundred people years to go-live

Base of open-source Erlang products - Riak, RabbitMQ

Architecture based on message passing between processes

Architecture based on normalising failure

Change is normal, weekly and automated


Comfort
Zone
Challenge

WHY AN ERLANG KEY-VALUE STORE?

Riak has been a rock - durability and availability

- I know it, and know of problems with it, and have a path to production

Pluggable backends, but no fully-featured Erlang backend

- Except for HanoiDB, so someone else thought this was worth doing

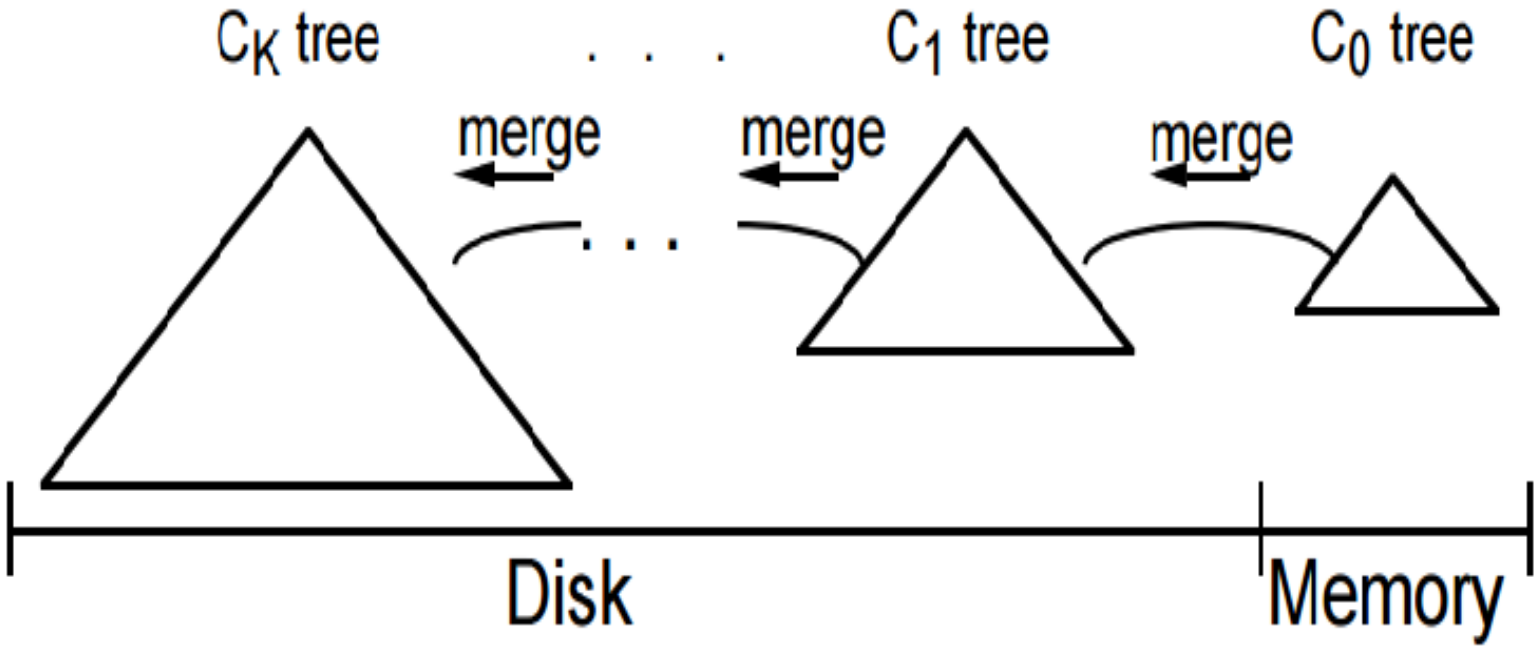


Figure 3.1. An LSM-tree of K+1 components

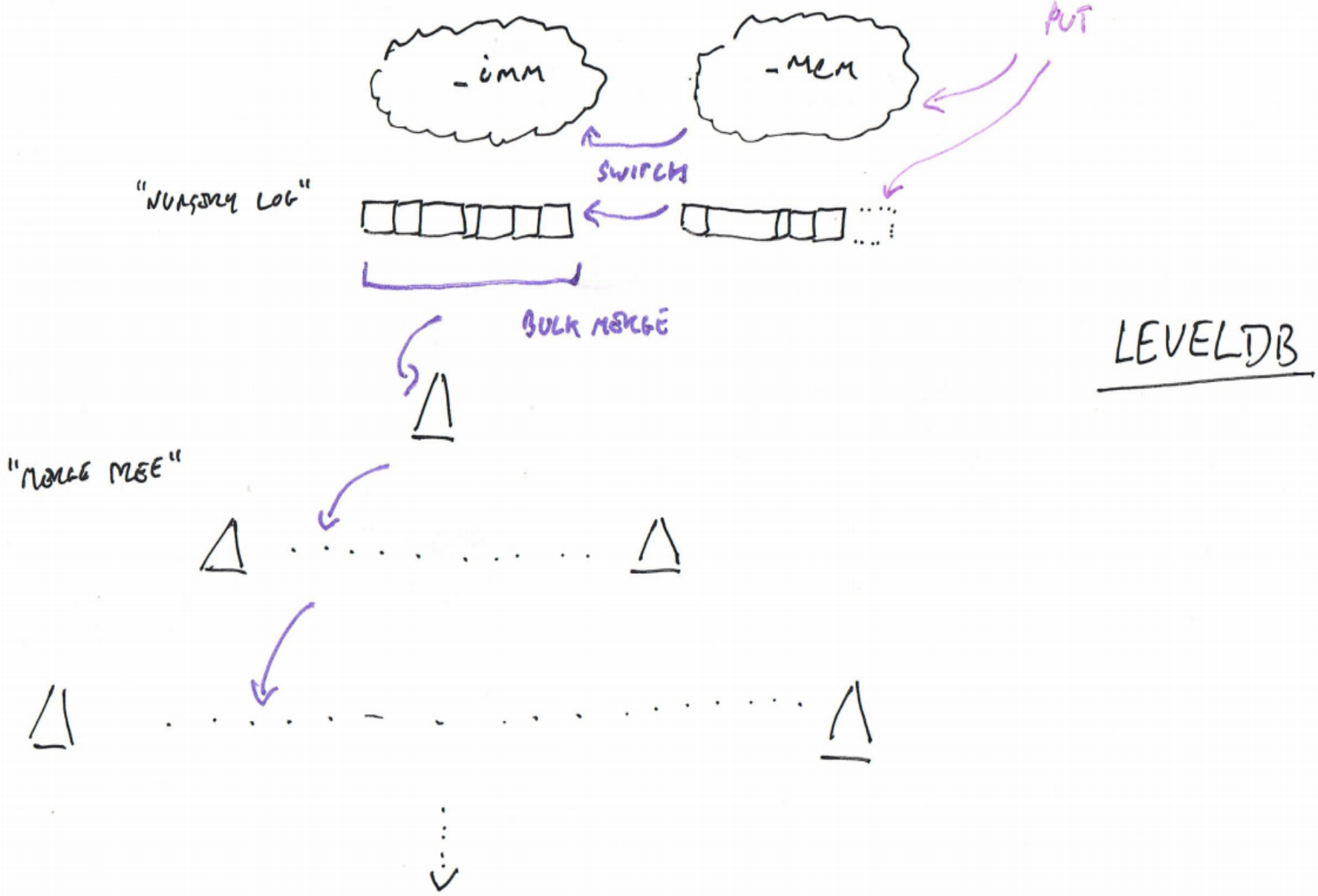
LEVELED - THE HYPOTHESIS

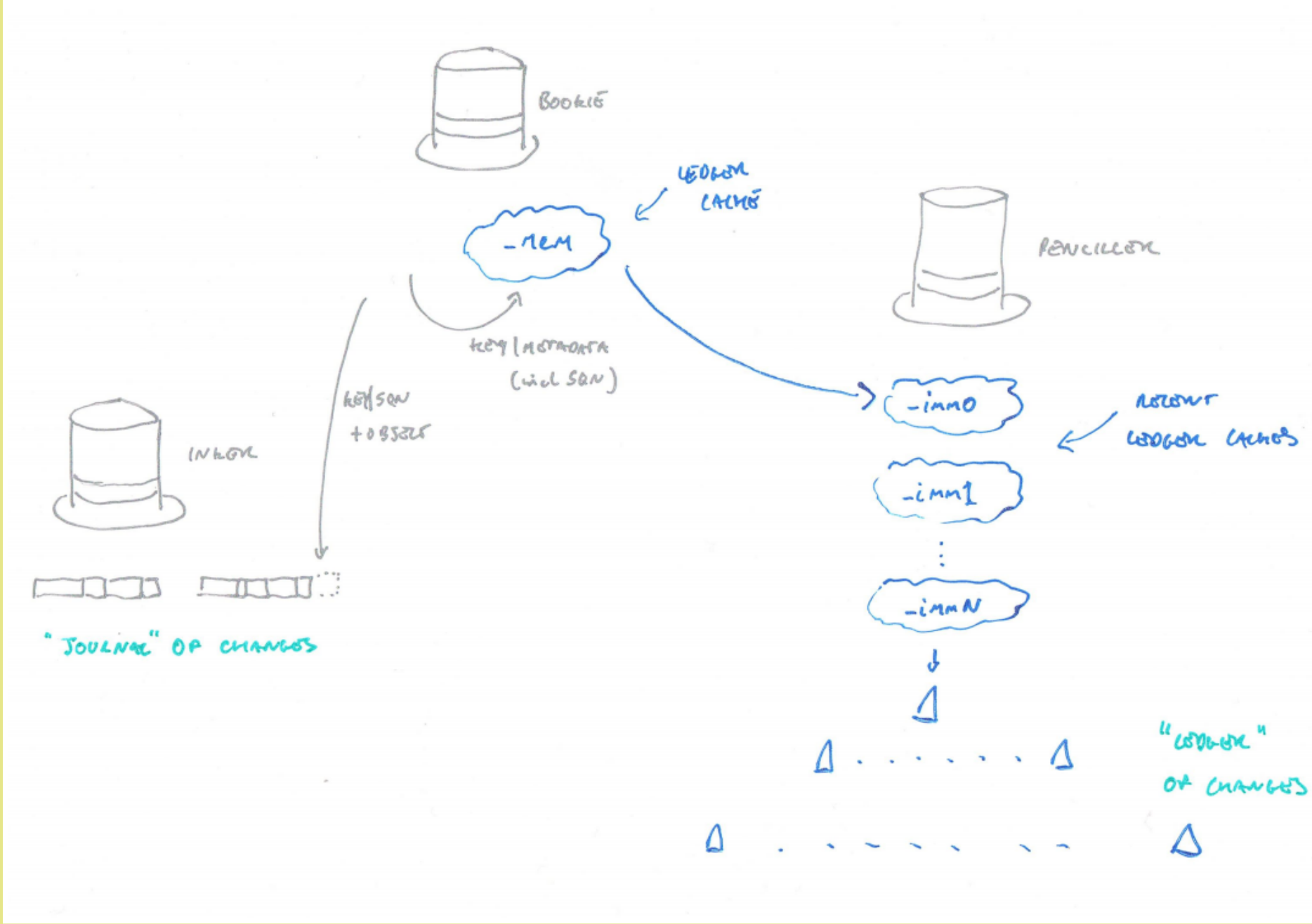
Disk I/O is an unpredictable bottleneck -> split VALUE

... See also WiscKey, Badger

Riak doesn't always need to know the value -> HEAD

Store behaviour may differ by object -> TAG


PUT
-imm
-mem
SWITCH
"NURSERY LOG"
BULK MERGE
LEVELDB
"MERGE MEE"


BOOKIE
LEDGER CACHE
-MEM
key | metadata (incl SQN)
PENICILLIN
key | SQN + OBJECT
INFLUX
-imm0
RECENT LEDGER CACHES
-imm1
-immN
"JOURNAL" OF CHANGES
"LEDGER" OF CHANGES

LEVELED - OPERATIONS

PUT - Inker commits to Journal, Bookie caches change to Ledger

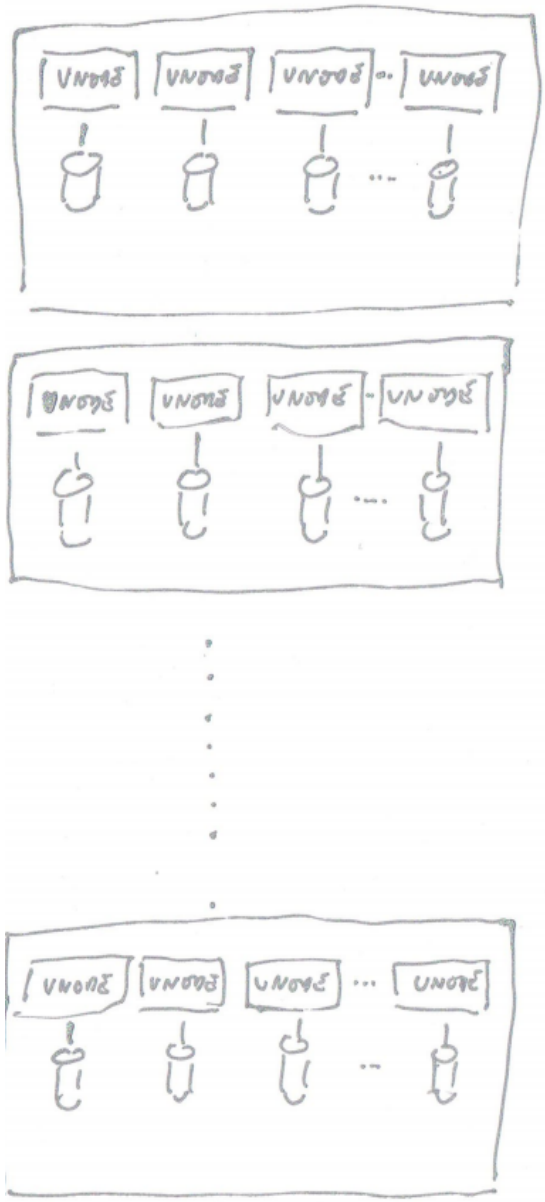
GET - Penciller fetches SQN, Inker fetches value

HEAD - Penciller fetches metadata from Ledger

INDEX - Additional key/metadata changes in Ledger

FOLD - Efficient in key-ordered ledger through clones of Penciller

CLONE - By manifest copy, with delete_pending file state, allowing reads in parallel



CURRENT "GET" FSM

BETTER FOR SMALL VALUES

1. REQUEST THREE (N) GETS
2. BUILD RESPONSE ON FIRST TWO (R) RESPONSES
3. SEND RESPONSE
4. WAIT FOR TRAIL RESPONSE (OPTIONAL)

ALWAYS LOWER MEDIAN LATENCY

ALTERNATE "GET" FSM

BETTER FOR LARGE VALUES

1. REQUEST THREE (N) HEADS
2. IF ON TWO (R) RESPONSES, ONE DOMINATES
   - L SEND SINGLE "GET" REQUEST

   OR ELSE
   - L, SEND MULTIPLE "GET" REQUESTS

MAYBE LESS I/O

3. BUILD RESPONSE FROM ALL RESPONSES


WORK
IN PROGRESS

LEVELED - STATUS

Functionally complete backend

Initial integration testing into Riak

Four months of cloud-based volume tests with improvements

Good ct/eunit coverage, plus initial propery-based testing

LEVELED - VOLUME TESTS

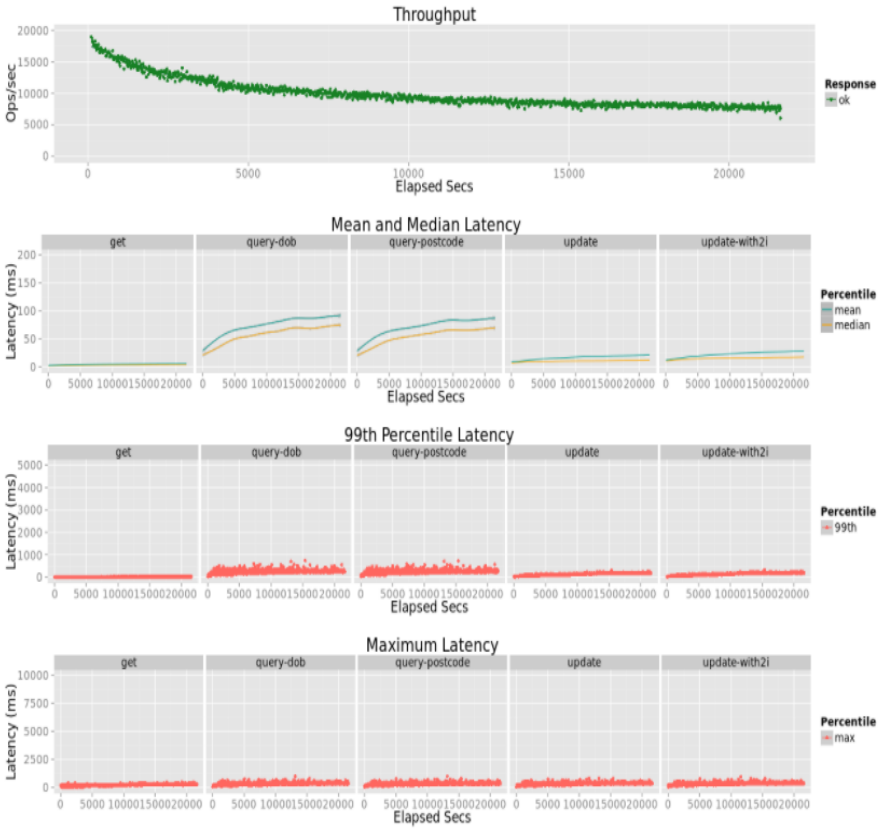
Significant throughput improvements where disk I/O is the dominant constraint

- With sync enabled (flushing each and every write)
- With spinning disk drives not solid-state drives

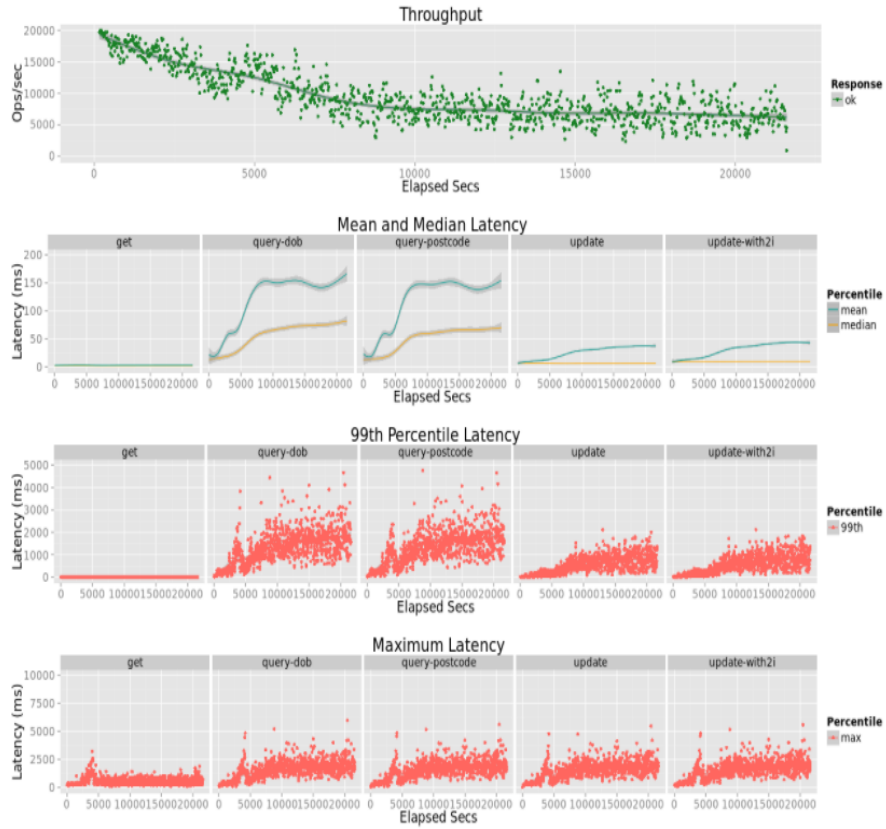
Focused on testing without sync on SSD since

- Throughput advantage at > 4KB values
- Advantage increases with value size
- Lower mean PUT times, higher median GET times
- Dramatic reduction in tail latency and volatility

Riak + leveled



Riak + leveldb



LEVELED - THE HARD BITS

Picking data structures in Erlang

Handling OTP16 compatability

Compacting the Journal (value store)

Vnode coordination issues

Avoiding long-tail blocking (e.g. the 40ms cast)

Naming things

LEVELED - WAS IT WORTH IT?

Learned loads about Erlang - will continue to use

Erlang/OTP coped well with my mistakes

Pleasantly surprised by the throughput comparison

Actors made more sense to me than objects

Relevance increased by support issues with Riak

Now progressing to pre-production testing on Spine

@masleeds

https://github.com/martinsumner/leveled