

# CockroachDB: The Road from 1 Node to 100

CodeMesh 2016

*presented by* Spencer Kimball / Co-Founder & CEO

# CockroachDB

## **Do we need another database?**

Existing database solutions place an undue burden on application developers



# CockroachDB

- Scalable
- Survivable
- Strongly Consistent
- SQL
- Open Source



## Some Background...

1.5 years into development

Team of 20 developers

5 months into beta

But...can't stand up a 10-node cluster for two weeks



# Agenda

What went wrong?

Remedial actions

Communication strategies

Technical fixes

Conclusions



# What Went Wrong?



# What Went Wrong?

Focus?

- Rush to v1.0 features

- Correctness

- Performance

Stability is not an emergent property

Accepting reality: Orwell had it right



## Past Experience

Centuries of engineering experience

Didn't save us from bad assumptions

Provided belief in a solution

Hypothesis:

**A small team could stabilize in isolation**

Don't underestimate belief!

A "Stability Code Yellow"



# Remedial Actions



# The Root Cause Wasn't Technical

Yes, technical solutions fixed stability

However, mgmt and process was the key

Thought experiment identified approaches

- Rapid churn overwhelming stability efforts

- Instability localized to decentralized & leaderless core components

- Insufficient scrutiny on critical changes



# One Step Forward, One Step Back?

Churn may have been overwhelming stability fixes, but  
no way to tell

Isolation has psychological advantages

**Solution:** split master branch

*Master*: frozen, but for stability fixes

*Develop*: all other changes

Freeze dependencies

Expensive solution, not well-loved



# Experimental Cooking

We needed a head chef

...and a team of experienced sous chefs

Time consuming: opportunity costs



# Testing the Small Team Hypothesis

Bigger probably isn't better

Focus, focus, focus

Increased scrutiny, gatekeeping

Fewer brains for holistic efforts

The question of physical proximity

Clearly defined goals, standups, war rooms

Volume per engineer?



# Visualizing Pull Requests



# Communication Strategies



## A Precipitous Decision

5 months in, what's another day?

Definitely reached a tipping point

Communication was next step

Internal

External?



# Corporate Values 101: Transparency

Two quarters of OKRs didn't pull punches

We were transparent, but honest?

Gulf between reality and expectation

A devastating critique

Initial communication via team email

Succeeded: explain bg, process, exit criteria

Failed: team deliberation



# Are We an OSS Project or Not?

No immediate decision, but easy consensus  
OSS

Forum vs. email, Gitter vs. Slack

While some risks, more of an opportunity

Clarify, set expectations

Build trust

Still no easy task to write blog post

Mix of  and support



# Technical Fixes



# The Many Faces of Instability

Precipitous slowdown

Deadlock

Out of Memory

Corruption

Raft impedance mismatch



# Rebalance and Recovery via Snapshots

Snapshots used for rebalance & recovery

Expense: typically ~32M

Disk I/O

Network I/O

Memory consumption

We thought we'd delay getting fancy

Streaming RPCs

Avoid locks during generation



# The Real Problem With Snapshots

The fog of war (stability edition)

Made all of the difficult fixes first!

Sometimes to see what's in front of one's nose...

The rebalancing algorithm

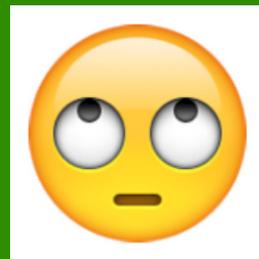
- Advertise capacity stats

- Compute cluster mean

- Xfer from overfull → underfull



# Rebalancing to Exact Means



# Diagnosing and Fixing Lock Contention

Excessively slow or deadlocked clusters

No graceful degradation

Tracing tools to the rescue

[Net/tracer](#)

[Lightstep](#)

Major refactorings

Raft processing

Replica GC



## Tracing Tool Trouble

What the Lord giveth, the Lord taketh away

Showing commands in traces is great

...until the command is a snapshot

The silver lining



# Why We Lose Sleep

Sources of system corruption

- Replica divergence

- Broken invariants

A meta2 record goes missing; hell breaks loose

Elementary, my dear Watson

**In a sufficiently large distributed system, if something can happen, it will happen**



# The Epic Struggle to Tame Raft

What is Raft?

Is it the right choice for CockroachDB?

Busy protocol

CockroachDB uses a Raft / range

Many ranges

Again thought we could ignore fancy stuff



## Raft Fixes

**Goal:** traffic proportional to client activity, not total storage

Lazy initialization

A further insight: quiescing



# Naive vs. Quiescing Raft



# Conclusions



# Code Yellow Status

80% of Q3 goals met

Still winding down

10 nodes stable

Successfully ran 100 nodes



# Winners and Losers

Split branches?

- Significant merge pain

- At best psychological benefit

Focused stability team w/ leader

- Proximity, standups, war room

- Opportunity costs

Smaller team, more scrutiny

- More or less permanent – prevention



# Could Instability Have Been Avoided?

According to Hacker News “experts”, yes

Argument from experience (a fallacy): no

Mostly impractical ideas

- Proving correctness

- Literate programming

Practical ideas

- Smaller MVP, earlier, faster instability

- Plan for focused stability effort



# Questions

[CockroachLabs.com](https://CockroachLabs.com)

[@cockroachdb](https://@cockroachdb)

