# Prince Building Tech Talk

# Three Years in the Trenches with MongoDB

{ by : "Ryan Nitz", date : "01-30-2013" }

# Overview

- The Epoch
- 10gen PaaS
- Love at First Sight
- Expansion
- Trouble in Paradise
- The "Pivot"
- This Works
- Early Days
- Vertical Ceiling
- Feature History
- Broken Fingers
- Dogfooding
- Lessons Learned

# The Epoch

- 10gen started as a open source Platform as a Service (Paas) in August of 2007
- First lines of code were committed on September 12, 2007
- Albert Wenger published his famous blog post, "I Want a New Platform" on September 19, 2007

# The 10gen PaaS

- Custom App server written in Java ("Babble")
- User App Dev in JavaScript
- App files backed by Git for versioning
- Storage was done in the new/custom database, Mongo

# Love at First Sight

- March 2008 - I started working as a consultant on a browser based IDE for the PaaS
- The love affair starts... truly rapid development is realized
- With a child on the way, I took a full time opportunity with another company (hindsight is always 20/20)

# Expansion

- 10gen raised $1.5mm from Union Square Ventures - Albert Wenger takes a seat on the 10gen board
- Datacenter is online and friends are using the production PaaS
- Drastic expansion, added dev support for Python, Ruby and PHP

# Trouble in Paradise

- Google App Engine (GAE) is released in April of 2008 - the product was originally not received well
- As a fledgling startup, 10gen had bitten off more than they could handle. Working on:
  - Custom database
  - Custom app server (supporting JavaScript, PHP, Python and Ruby)
  - Production hosting environment

# The "Pivot"

- January of 2009 - decided to stop development of the PaaS and focus on MongoDB
- Stopped development of Babble app server - Repositories:
  - MongoDB (AGPL)
  - Babble App Server (Apache 2.0) - 10gen sponsored development stopped

# This Works

- Feb 2009 - Started playing with MongoDB and submitting bugs/feature requests to 10gen
  - Ability to tune cursor size/number of documents returned by a query
  - Binary data type in the shell
  - MD5 data type
  - JMX in Java driver
- July 2009 - Buenos Aires for a year
  - Planned sabbatical... instead I ended up doing research for an ad technology startup in NYC

# The Early Days - 2009

- Global lock is a huge problem
- Sharding not available
- AWS does not have guaranteed IOPs
- No journaling
  - Similar to the MyISAM storage engine in this respect
  - Thus data corruption & fsck-style repairs necessary on a hard kill or crash.  In particular when not using replication.
- Did not know about locking database for backups

# The Vertical Ceiling

- System was scaled up to ~ 5,000 updates per second on a single host - more capacity needed soon
- All vertical scaling - reached max AWS instance type
- Panicking!!!
- Sharding is supposed to be released by the end of 2009

# Vertical Ceiling Fixes - App

- Tuned application
  - Some data offloaded to S3
  - Updates for a document batched in CoR and handled as one upsert
  - Additional functionality offloaded to Solr
  - Moved away from skip/limit (when possible)
  - Moved away from count to recurring batch jobs that slowly walked cursor (when possible)

# Vertical Ceiling Band Aids - DB

- Decreased fsync frequency
- Bound database to a specific CPU
- Frequent manual compaction (repair db at the time)

# Sharding Alpha

- Original alpha releases buggy
- Not evenly distributing data properly
- I used a bad shard key (all random)
- We had to wait for further stability
- Contract ended... someone else took over development of the system

# Feature History

- v1.4
  - geospatial
  - sharding alpha
- v1.6
  - sharding
  - replica sets
- v1.8
  - journaling in the storage engine (crash recovery redo log)
- v2.0
  - lots of small improvements in just about all components
- v2.2
  - no global lock (first iteration database level... collection and page level coming)
  - agg framework

# Broken Fingers

- Moved back to NYC (August 2010)
- Started consulting for a music subscription service startup
- They were tied to MySQL (JPA)
- Productivity was CRUSHED
- Ported dozens of stored procedures to logic in the app server #pain

# 10gen

- Started consulting for 10gen in December of 2010
- Designed and implemented the MongoDB Monitoring Service (MMS)
- Rapid development again!!!!
- Beta launched service in ~ four weeks (holidays ate some time)

# Dogfooding

- Original/current MMS agent in Python
  - Pymongo was a fairly immature at the time
  - Pymongo replica set bugs
  - Pymongo memory leaks
- Pymongo is now in an excellent state
  - Shout-out to Bernie and Jesse
- V2 of the agent is moving to Go

# Dogfooding

- MMS tests Java driver releases once they hit RC status
- MMS typically tests all mongod releases before they reach GA

# Lessons Learned

- Application instrumentation is essential
  - Code changes over time - can impact db
  - Load/usage changes - can impact db
  - Custom Munin Plugins, Graphite, etc.
- Infrastructure instrumentation is essential
  - Cloud providers have issues
  - External networks have issues
  - Hardware fails

# Lessons Learned

- Database instrumentation is essential
  - MongoDB Monitoring Service (MMS)
    - https://mms.10gen.com
  - Munin Plugins
    - http://bit.ly/fXXV9d

# Lessons Learned

- Tune your write concerns carefully based on operation
- Application/system functionality must be degradable to handle infrastructure issues
- Schema design is CRITICAL
  - Never stop looking for improvments
  - MMS db rrd schema has gone through many iterations
  - Refactor code when new MongoDB functionality is released

# Lessons Learned

- Schedule batch jobs wisely
  - Document for all to see
  - Tune cursor sizes on batch jobs to minimize impact on a system
- System alerts are essential
  - Proper signal to noise ratio
  - MMS metric based alerts
  - Munin alerts
  - Nagios alerts

# Lessons Learned

- Know your scaling points
  - Set max levels that correlate to performance degredation
- Calculate IOP requirements properly
  - Scaling applications is not magic
  - http://www.wmarow.com/strcalc/
- Backup your data properly
  - A secondary in a replica set is not a backup
  - Lock your secondary when backing up - app logic must be able to accommodate (if reading from secondaries)

# Lessons Learned

- Tune applications/systems as you scale
  - Adding app servers requires tuning connection/thread pools
- Compact DBs
  - Promote secondaries and resync
  - 10gen working on online compaction
- Run SSDs
  - The future is here... anything else is a waste of time
- explain() is your best friend

# Lessons Learned

- Ask when you do not know
  - MongoDB User Google Group - http://bit.ly/JJz1TV
  - Stack Overflow - http://stackoverflow.com/tags/mongodb
  - IRC - freenode.net/#mongodb
  - Commercial Support - http://www.10gen.com

# Questions?

Image by Horia Varlan - http://bit.ly/dsflj0

# Thanks!

- Me: https://github.com/rgnitz
- Slides: http://bit.ly/VXUcVc