

Blend in Chicago: MongoDB World 2017

Yubo Su

Blend

June 21, 2017

- Tom Schenk, Chief data officer, Chicago. *WindyGrid*.
 - Track colocated data, 911 calls to Tweets to weather.
 - Flexible schema: {what, when, where}
 - Predictive analytics (example, where to send food inspector) using visualization of multiple causal layers.
- Dev Ittycheria, CEO MongoDB
 - 2007 is watershed year, AWS, iPhone, Android, and many others.
 - Argue b/c storage costs dropped below a critical point.
 - MongoDB also in 2007: document model, distributed systems + aggregation.

- Eliot Horowitz, CTO, MongoDB
 - 3.6 ships November, already on Github.
 - MongoDB Charts (3.6)
 - Business Intelligence: BI Connector is SQL interface.
 - Coercing data to table is difficult: polymorphic schemas, arrays.
 - Solution: *MongoDB Charts!* Data visualization tool, handles above.
 - 3.6 document model features:
 - `$lookup` takes sub-pipelines!
 - `$update` can operate on arrays natively! Takes a filter over array entries, can iterate over nested.
 - JSON Schemas.
 - 3.6 distributed systems:
 - Native retryable writes
 - *Change Streams* can get a stream of changes to a db.

- Eliot Horowitz, CTO, MongoDB (continued)
 - Mongo Atlas
 - “Should be irresponsible to run MongoDB in cloud w/o Atlas”
 - Built in security, one-click spin up, built in scaling elasticity.
 - Data browser + performance viewer in UI (utilization stats, examine queries as stream, explore data),
 - Live migration service (not very live in demo, requires downtime for mirror to catch up and change source of truth).
 - Now with MS Azure + Google Cloud support too (+ AWS).
 - Performance Adviser.
 - CRUD support in data browser.
 - Charts!
 - LDAP Auth.
 - Cross-region, cross-cloud!
- MongoDB Stitch (Beta as of today in Atlas, 06/20/17)
 - “Backend as a service”
 - REST API for MongoDB
 - Configuration-based auth/security
 - Service composition to govern how services talk to each other.

Squeezing the Most out of Your Document Model

06/20/17 1050-1130: Norberto Leite, Lead Curriculum Engineer, MongoDB

- Nested schema, spectrum of highly normalized or denormed storage.
 - Normalized requires foreign keys, requires looking into many collections.
 - Denorm is simpler query, complex schema.
- Consider three possible behaviors:
 - Get player: Denorm outperforms.
 - Add new field to doc: either add new collection or modify every doc, the same.
 - Change existing field: If a highly shared field, normalized is very fast.
- Optimizing highly normalized:
 - Can optimize with aggregate, but more importantly `db.createView()`.
 - Views are basically stored aggregates.
 - Better `$project` support.
 - Also consider, if reading much more than writing, should store calculated fields!
- Optimizing denormed:
 - Should normalize fields that are infrequently updated.
- `tl;dr` normalized have fast write, slow reads. Should embed everything that is infrequently updated.

Advanced Schema Design Patterns

06/20/17 1140–1220: Daniel Coupal, Senior Curriculum Engineer, MongoDB

- Axiom: data models maximize performance + scalability despite latency, costs, hardware.
- Common issues #1, too many optional fields:
 - Use attribute array, `[{key: keyName, value}]`.
 - Accommodates optional fields.
- Common issues #2, working set does not fit in RAM.
 - Can subset, truncate data
 - Probably also useful for showing users too.
- Common issues #3, data consistency.
 - Accept instantaneous inconsistency, duplicate at regular intervals ☺.
- Common issues #4, repeated computations
 - Reads generally outnumber writes, apply computation on write.
- Common issues #5, expensive tracking
 - e.g. expensive to increment on every page view
 - Solution: random number in range $[1, N]$, increment by N .
- Common issues #6, large data easily overflow
 - Bucket, store buckets into a separate collection.

- Microservices vs. monolith, preferable b/c web scale, faster iteration, compartmentalized.
- One common rule of thumb is that one developer can own the whole thing, a couple hundred lines, but not everybody
- Hard metal vs. Docker (Kubernetes) vs. Atlas.
- Kafka can run general events while Mongo streams (the new feature) only handles database updates.

Index Usage for Nested Logical Queries

06/20/17 1440-1520: Tess Avitabile, Software engineer, MongoDB

- Query system overview:
 - Input: JSON
 - Parse into tree
 - Generate plan (which indices for which leaves of the tree)
 - Plan selection: try all of the plans for a trial period, see which one was fastest (Note: plan caching)
 - Execution & return
- ORs inside of ANDs is a pain for plan generation.
 - AND is considered indexed when one child has index.
 - OR is considered indexed when all children have indices.
 - ORs have to dedupe by hashing to merge the two results.
- Problem: no tight index bounds on these queries
 - *Tight* index bounds are when all documents in index bounds match the query.
 - (As opposed to when a parent node imposes a filter, `FETCH`)
- Bounds are not tight b/c two branches of children cannot talk to each other! e.g. the OR will not be tight since the AND above will have to further fetch against its other child.

Index Usage for Nested Logical Queries (cont)

06/20/17 1440-1520: Tess Avitabile, Software engineer, MongoDB

- Solution: Disjunctive Normal Form?
 - AND with OR child solved!
 - Exponentially many plans though, index choices at each child.
- Solution: OR-pushdowns! Predicates pulled up to the AND parent and pushed down into any OR children if they can tighten index bounds.
 - Note that this is not imposed as an extra AND condition, just metadata for the recursive query planner to plan against.
- Paper: Query Optimization by Predicate Move-Around

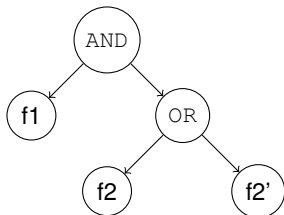


Figure: AND with OR child. Consider if index is $\{f1, f2\}$? $\{f2, f1\}$?

Multi-Master architectures in MongoDB

06/20/17 1530–1610: Pavel DUCHOVNY, TSE, MongoDB

- Key to geographic colocation.
- Zone sharding + replica sets
 - Zone sharding: shards per region.
 - Replica sets are `mongod` processes that share the same data.
- Configuration is:
 - One primary in each region, each a separate zone shard.
 - One secondary in its own region (prio 3), two in another (prio 2), and two more in a third (prio 1, 0) for symmetry across all regions, hidden secondary.
 - Spread across multiple regions, odd number of voting members, primary DC members should have higher priorities.
- Can specify region on read/write.
- Upshot is that can do multi-region writes while guaranteeing local availability on read.
- Configurable to write to secondary especially if primary lives in a different region.
- Can configure with `MAX_STALENESS` parameter for when a cluster can be read from.

Globally Distributed RESTful Object Storage

06/20/17 1620-1700: Julio Viera, Backend VP, Fuze

- Built an object store for internal communications, chat + attachment retention, Mongo backbone.
- RESTful so easy to expose HTTP link as a db.
- Nested schema corresponding to URL:
`/users/:id/chat/convs/X/messages/Y`
- Storage (chat), collection (convs), sub-collection (messages), documentIds
- Pubsub (user is online) can be done by consuming the oplog on the db primary.
- To shard and hide it to the user, just need some lookups userId → sharding keys.

- Saska Mojsilovic, IBM
 - Need for more data in health for precision health service distribution.
 - All sorts of orgs estimating and predicting from sparse data.
- Claudio Gosiker (Florida Blue) & Alan Chhabra (MongoDB)
 - Use data for healthcare outreach, personalizable views for customer reps.
- Matt Parker, Stand-up Mathematician!

- Bjorn Freeman-Benson, CTO, InVisionApp
 - Via microservices, can stand up new cluster in 10m!
 - Also has a `bailey stage it` etc.
 - QA against EA customers, automatically rolls out to rest of customers afterwards (24h).
- Cisco moved eCommerce to MongoDB, 40b connections?
- Justin Moses, Lead Software Engineer, MongoDB
 - Data auralization vs. visualization!
 - `npmjs link`
 - Just turns numbers into music.
- Jane McGonigal, Game Designer, AvantGame
 - 2.1b gamers > 1h/day, more stats.
 - 72% workforce not engaged, vs. 80+% of schoolchildren engaged.
 - Consider: *"Opposite of play is not work but depression."*
 - Video games overstimulate brain regions exactly what depression suppresses.
 - Pokemon Go fitness lol.
 - Reality's obligation to engage the way video games do, AR > VR!

Migrating from EC2 to Atlas

06/20/17 1050–1130: Jesse Dearing, SRE, InVision

- Mongo at InVision
 - 28 replica sets 4 env
 - 2000 rps, 600 wps
 - Chef to manage EC2, Mongo
- Old stack:
 - EC2 instance, deploy, manually configure/shard
 - Manual: backups, monitoring, alerting, security, updates
- Atlas:
 - All above, REST API, dashboards
- Transition Preparation
 - SSL (Atlas mandatory)
 - AWS VPC Peering
 - VPN + security setup, Amazon DNS
 - MongoDB 3.x + WiredTiger
- Transition
 - UI Live Migrator (<1m downtime for oplog)
 - `mongomirror` for full ZDT: Initial sync, streams oplog, point to new instance *before* fully synced, continues syncing.
 - Full ZDT but momentary inconsistency (< 1s).
 - In case of rolling ZDT deploys when re-pointing, inconsistency is most noticable; graceful degradation!
- Epilogue
 - Alerts, new playbooks, backup restores.
 - Automatic provisioning for new services that need MongoDB.

MONGODB
WORLD'17

blend

Design Patterns Resilient to Infrastructure Failure

06/20/17 1140-1220: Feng Qu, Senior MTS, eBay

- Availability = $\frac{MTTF}{MTTF + MTTR}$,
mean time to failure/recovery.
- High write (?) 3+2+2, 2 slaves each
except maybe 2 arbiters (only used for
quorum voting).
 - 2 slaves on master so if one slave
fails, master still has a slave
 - Master fail → slave
 - Primary datacenter fail → re-elect in
second datacenter
- High read goes to 3+3+3
- Shard for overall read/write scaling.
- For latency, `readPreference=nearest` or
`readPreference=secondaryPreferred`.
- `WriteConcern majority`.
- Tagged reads: read only from nodes
that made it into the write majority, get
updated data w/o paying remote read
penalty.
- tl;dr many patterns given different
read/write patterns latency vs.
throughput questions.
- Mongo supports causal consistency.

Write and Read Concern

06/20/17 1400–1440: Alex Komyagin, Senior Consulting Engineer, MongoDB

- How Mongo writes (3-replica)
 - App sends doc to primary, primary writes to in-memory structures and oplog, sends ack back to app.
 - In no particular order: journal, replicated to secondaries, then eventually (~ seconds) data files.
- How writes disappear
 - What if primary fails right before replication + data files?
 - One secondary is promoted, primary reboots.
 - Once primary reboots as secondary, produces rollback containing in-transit write `bson`.
 - App cannot find in-transit write!
- Many people disable rollback ☹☹☹
- RCA: Ack before replicated.
- Solution: `writeConcern`, # replicas or `majority`.
- Rec: majority for important writes, else 2 (lower latency, but at least one other server knows).
- Dirty reads: detect when read is of unstable state. Modes:
 - Local
 - Majority
 - Linearizable (blocks until last write is replicated)
- Linearizable can block e.g. example above, when master goes down, since the write never gets replicated.

MONGODB
WORLD'17

blend

Write and Read Concern

06/20/17 1400–1440: Alex Komyagin, Senior Consulting Engineer, MongoDB

- Majority vs. Linearizable
 - Majority returns the majority that a particular node knows about.
 - Linearizable guarantees recency, but can only work on primary, very slow.
- Retryable writes can handle write concern timeout.