NoDB
A Database For Ships At Lightspeed

Malcolm Matalka

Klarna AB

June 2014
About

- Klarna is an ecommerce payments service from Sweden trying to take over the world
- Heavy users of Erlang
- On second-generation purchase taking system
NoDB

- Eventually consistent key-value database overlay
- Favors write-availability
- Synchronizes data between multiple systems
- Allows for large latencies
Eventual Consistency

- “guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value” - Some ACM Article
- Popular EC DBs - Riak, Cassandra
- Generally quite different from what developers are used to
Why would anyone want EC???

- Scalability
- Availability
- Geolocality
- Great for caching!
Examples of EC Systems

- DNS
- Amazon
- Financial transactions
Interplanetary Internet

- End-to-end information flow across the solar system
- "IP-like" protocol suite tailored to operate over long round trip light times
- Layered open architecture supports evolution and international interoperability
Borg - That sounds Swedish

WE ARE THE BORG KITTENS
Resistance iz futile
First Contact
Write-availability
Really, what is NoDB?

- More of a model than an implementation
- Allows separate systems to share data
- Works across transactional stores and EC stores (read the fine print)
- Not a database but a layer joining them
- Allows for out-of-order replication of data
- Inbetween step for going from monolith to SOA
Vector Clocks!!!!!!
NoDB: The Implementation

- Implemented in Erlang
- Riak ⇔ Mnesia
- Mnesia is system of record
- Uses RabbitMQ as transport
- Realtime replication and manual exports
NoDB: Mnesia Side

- Have a layer on top with a transaction log
- Listen in on transaction log
- Can iterate all keys in a table for export
- System of record
- Can replay history, great for catching up during downtime
- Vector clocks in shadow table
NoDB: Riak Side

- PUT/GET/DEL through a NoDB API, writes to Riak & replicates
- Importer pulls off Rabbit and pushes directly to Riak
- Layer to locally queue exports when Rabbit is down
- Vector clock stored with data
What's an Update Look Like?

- Get your data
- Modify it and bump the vector clock
- Write to DB + replicate
- Mnesia wrapped up in an abstraction and we sneak the vector clock into the transaction for the user (so kind!)
- On Mnesia, handle resolving possible siblings on write
- On Riak, handle resolving siblings on next update
Retrospective

- People hate using it. But they’re wrong
- Out of order events are hard to handle
- Being able to replay writes is really awesome (saved us from losing money multiple times)
Future Work

- Unify codebases, they were written in haste by different people
- Remove RabbitMQ, it’s not actually doing anything for us here
- Make it easier for non-Erlang systems
- Postgres support??
Questions?