Enterprise Problems Internet Scale

Alex Lloyd
Senior Staff Software Engineer

Big Data, Big Systems

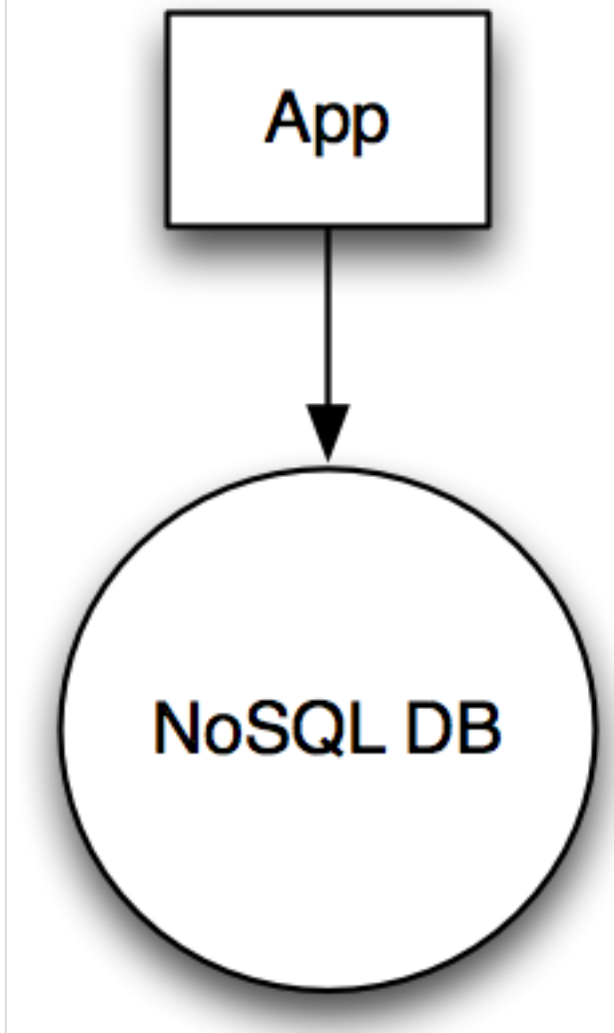
Time to move past false dichotomy

- Little, complicated databases
- Huge, scalable, simple ones

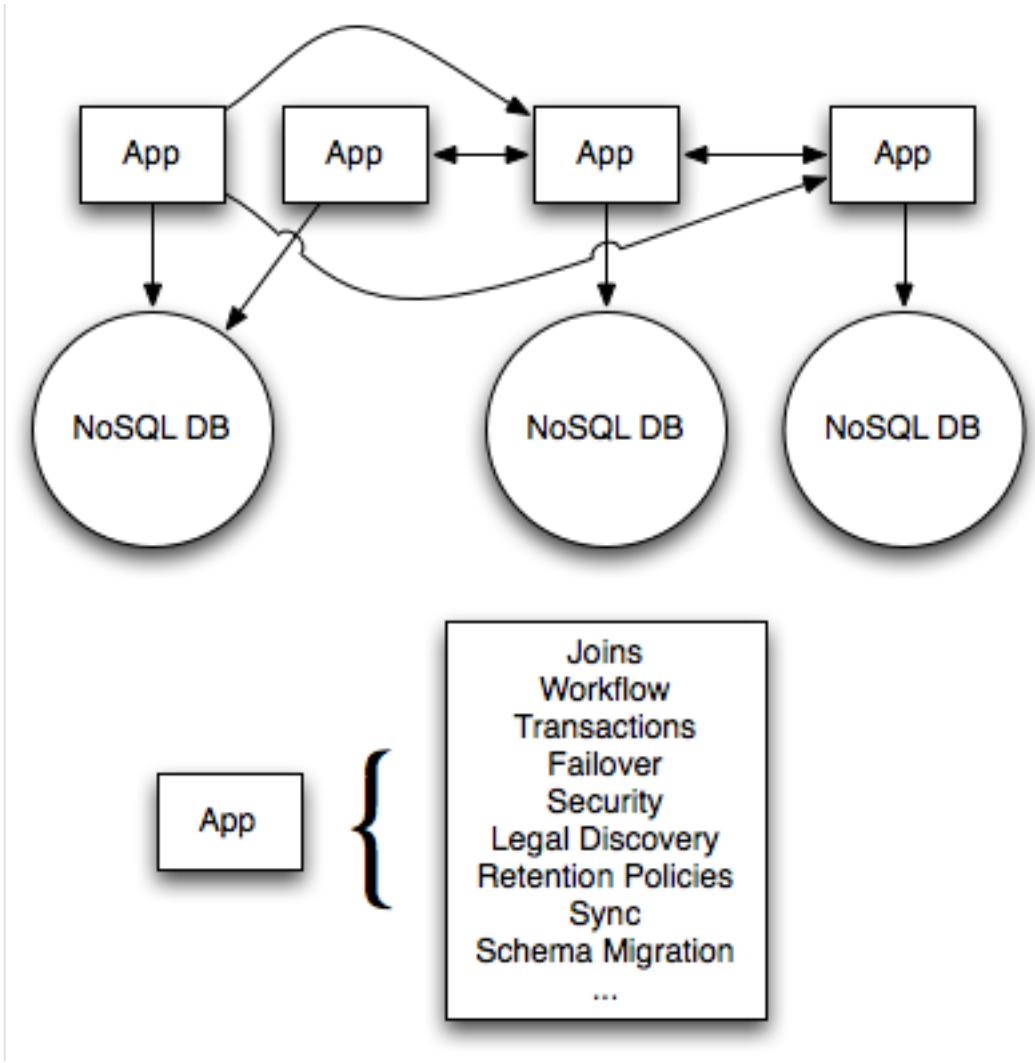
Complexity exists for a reason and ends up somewhere.

How do we push complexity down so app developers build features, not databases?

Starting Point



Entropy Increases



Big Data, Big Systems

How do we push complexity down so app developers build features, not databases?

- Reimplement best of SQL world at NoSQL scale
  - ACID, distributed (occasionally) transactions
  - Joins
  - SQL (with trees)
- And fix some holes
  - Standardized sync
  - Predictable performance
  - Scalable programming model

Anatomy of a Fast-Moving App Team

- Spends time on features, not concurrency anomalies:
  ACID transactions and Global Serializability
  - Key to users' trust in the cloud

- Codes 1-step transactions, not 10-step workflows:
  Distributed Transactions
  - Great for low-frequency, high-complexity operations
  - Job of concurrency control to isolate impact

- Writes UI code, not data loops:
  SQL
  - Extended for hierarchical input and output

Fast-Moving Team: Joins

Queries what the user wants:

- "upcoming concerts by bands whose members are in bands my friends listen to"
- "'see spot run' in documents I have access to"

Brutal to scale → fun questions:

- Leverage RDMA (implies CoW data structures?)
- Automatic pre-joins / partial materialized views
- Millisecond-scale congestion control

Fast-Moving Team

- Polishes mobile app UI, not sync protocol: Standardized Sync
  - Log of deltas + Operational Transformation avoids unimplementable merge functions

- Sleeps blissfully while pager lies quiet: Predictable Performance
  - Pay as you go: deferred compactions, SSD slowdown
  - Get what you paid for:
    - Seeks fairly scheduled by exactly one layer
    - Switch buffers fairly shared

Many Harmonious Fast-Moving Teams

Need a scalable programming model

- Join and sync datasets owned by disparate teams
- Isolate from each others' schema changes
- How embed modular ACLs, retention policies, and business logic in a system that still looks like a DB?
  - Executes *underneath* queries, MapReduce, etc.

Ultimate goal: localize risk for fast iteration with shared data.

We can have our features and scale them too.

Thanks,
Alex Lloyd (alloyd@google.com)