goto; conference aarhus

RIAK ON DRUGS (AND THE OTHER WAY AROUND)

Kresten Krab Thorup
CTO, Trifork

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

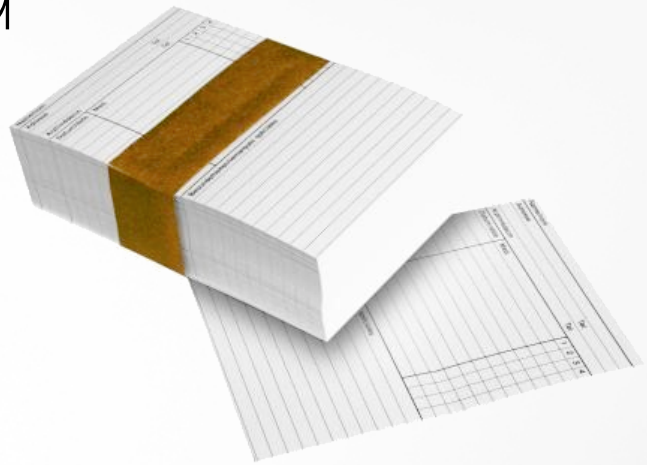
About the Speaker

- **Language/Runtime Geek** Emacs/TeX Hacker, Objective C, NeXT, GNU Compiled Java, Java Generics, J2EE, Erlang Hacker, Ph.D.
- **Trifork CTO** Conference “Editor”, Technology Adoption [Erlang / Riak]

In this talk…

- About Common Medicine Card
- Building a Decentralized Architecture
- Mapping different “shapes of data” to a Key/Value store
- Experiences with Riak along the way

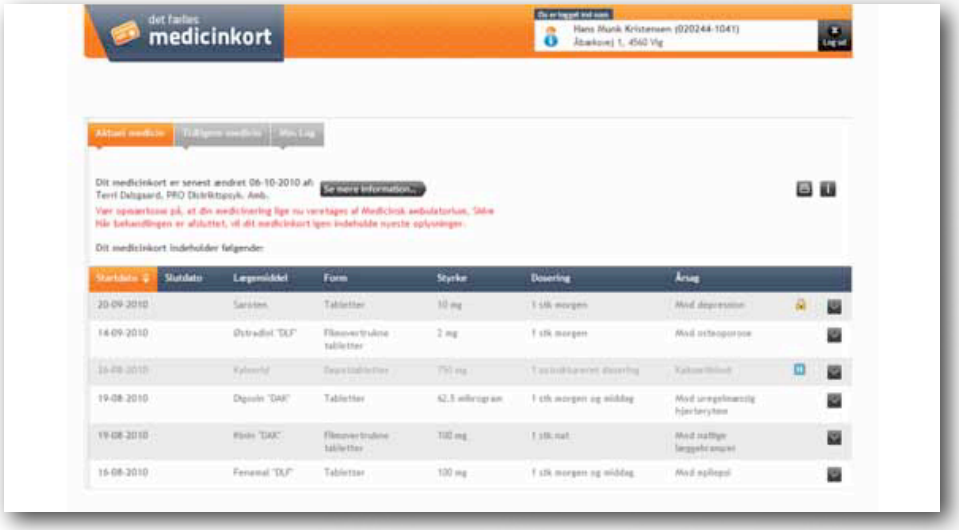
A Medicine Card



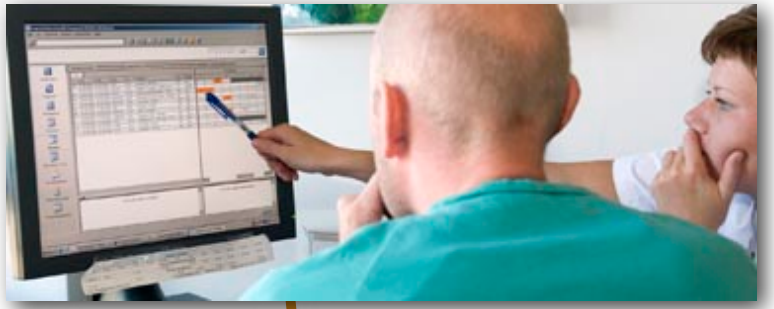
- For a person
- List of current drug treatments
- With prescriptions and related events

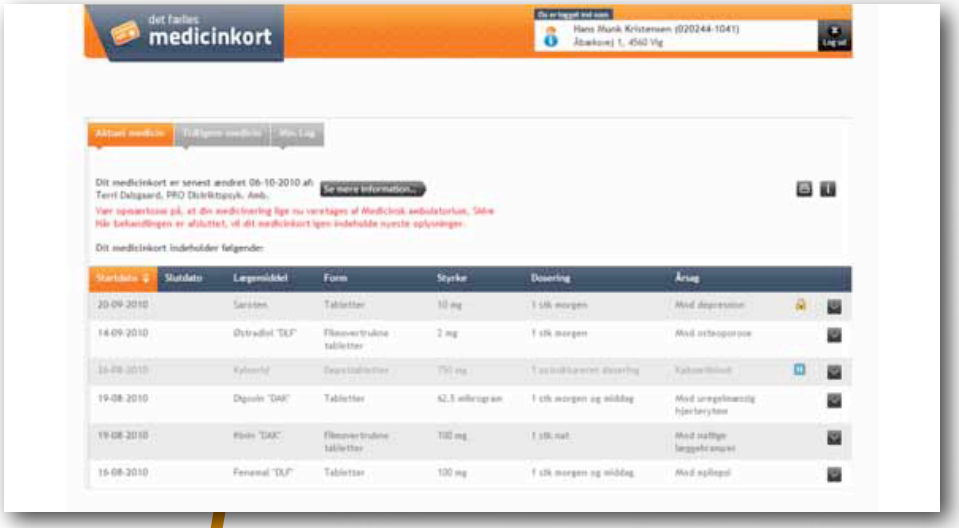
Common Medicine Card

Common Medicine Card



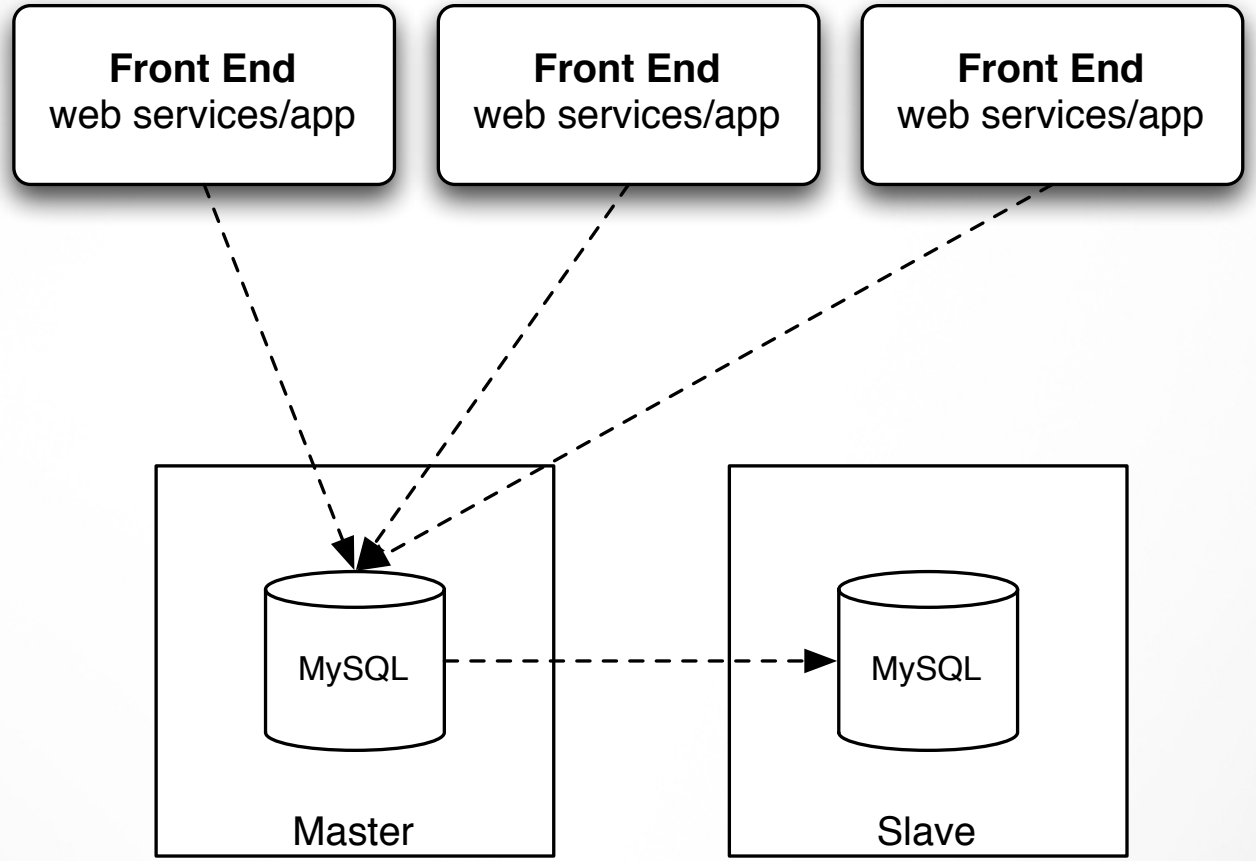
15-30 existing systems
+150k users

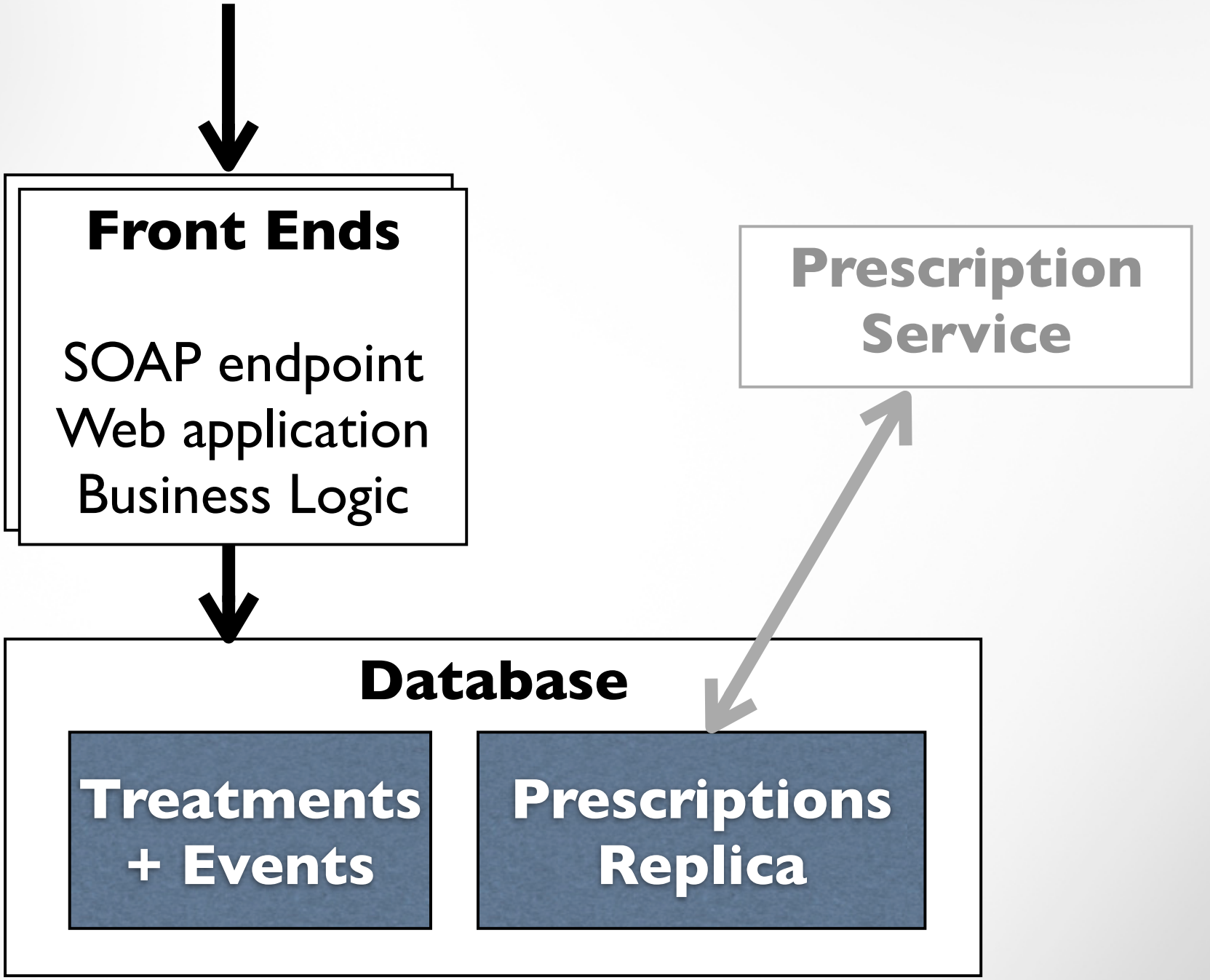




SOAP

"Old" Architecture



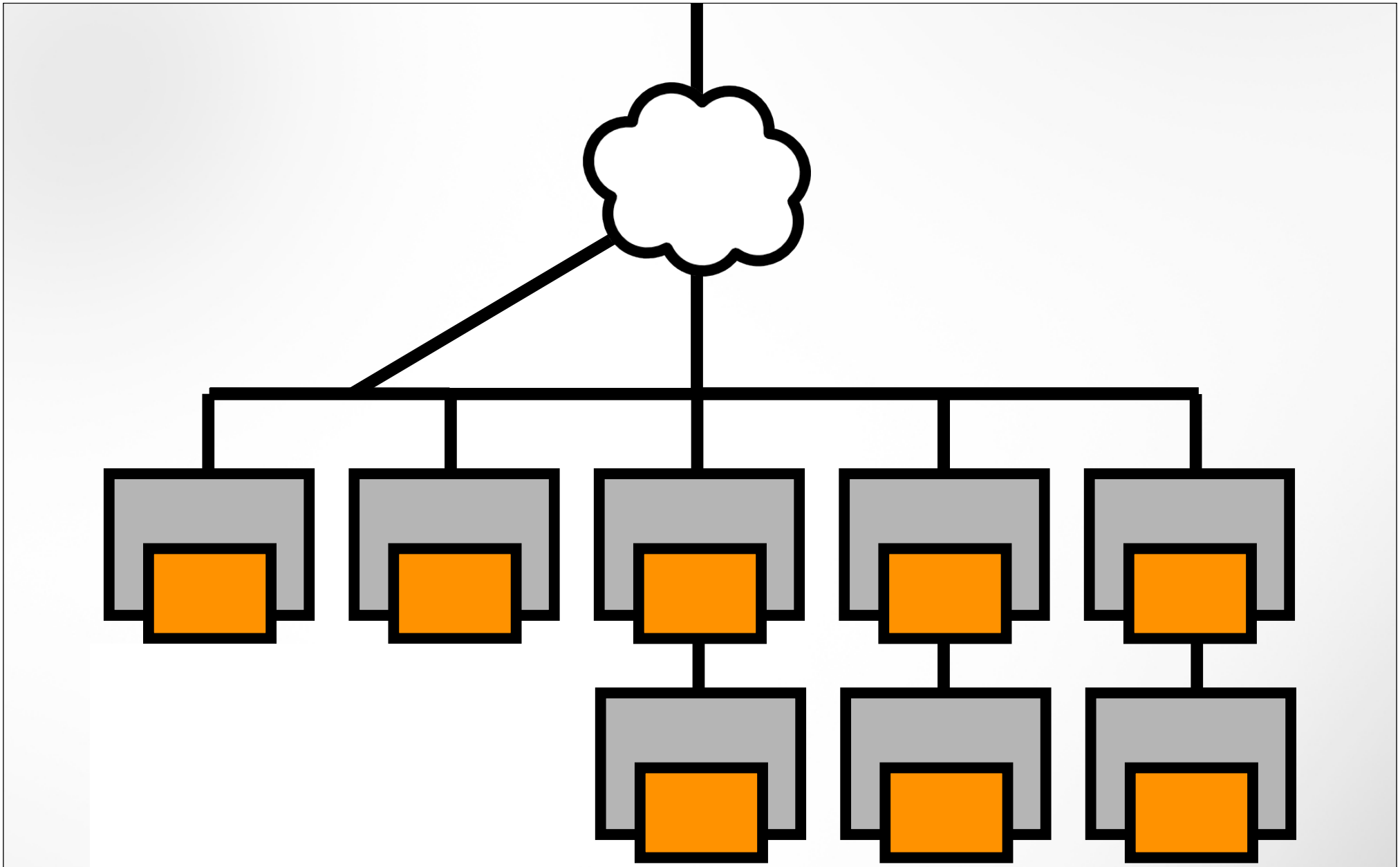

Front Ends
SOAP endpoint
Web application
Business Logic
Prescription Service
Database
Treatments + Events
Prescriptions Replica

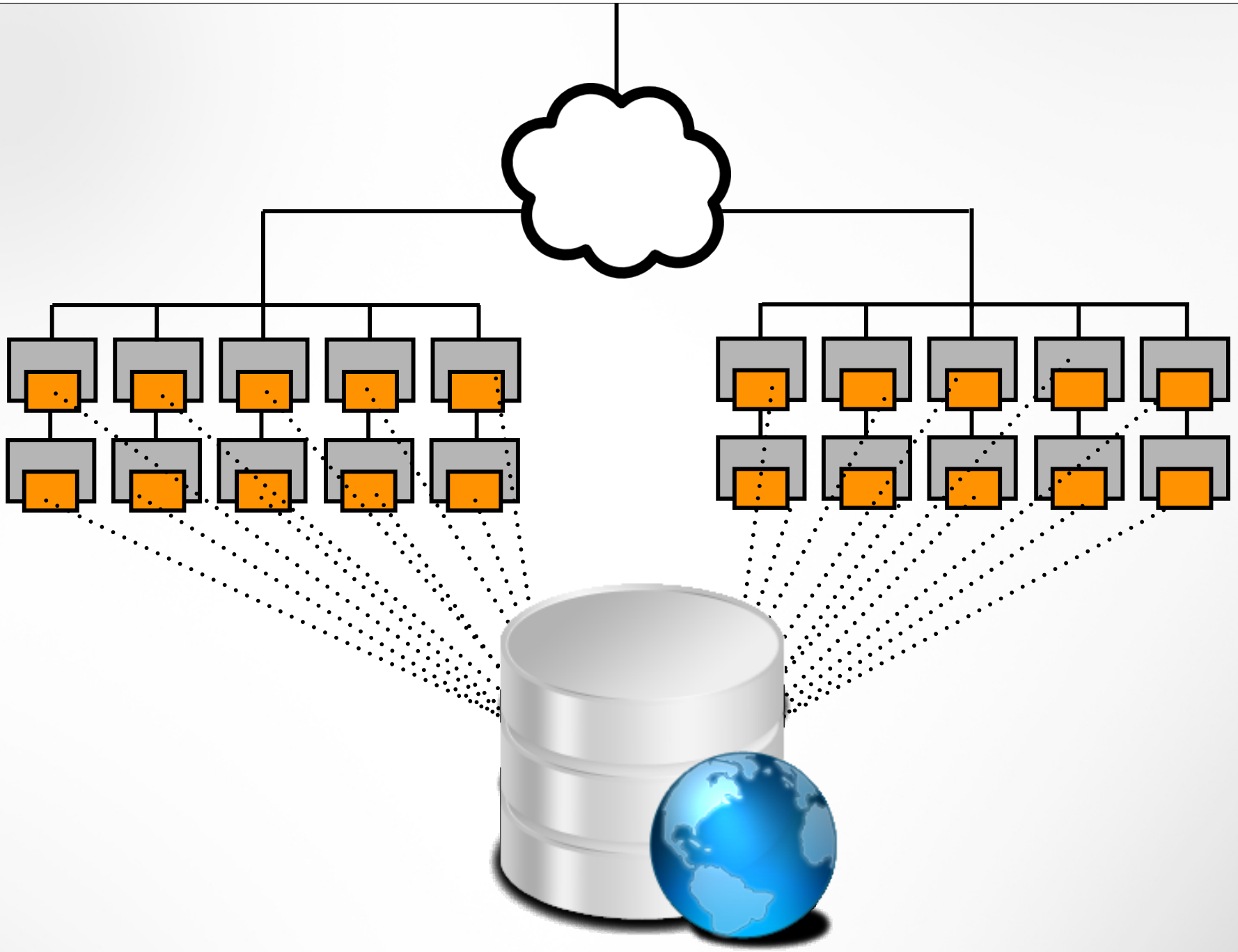
Distributed Architecture

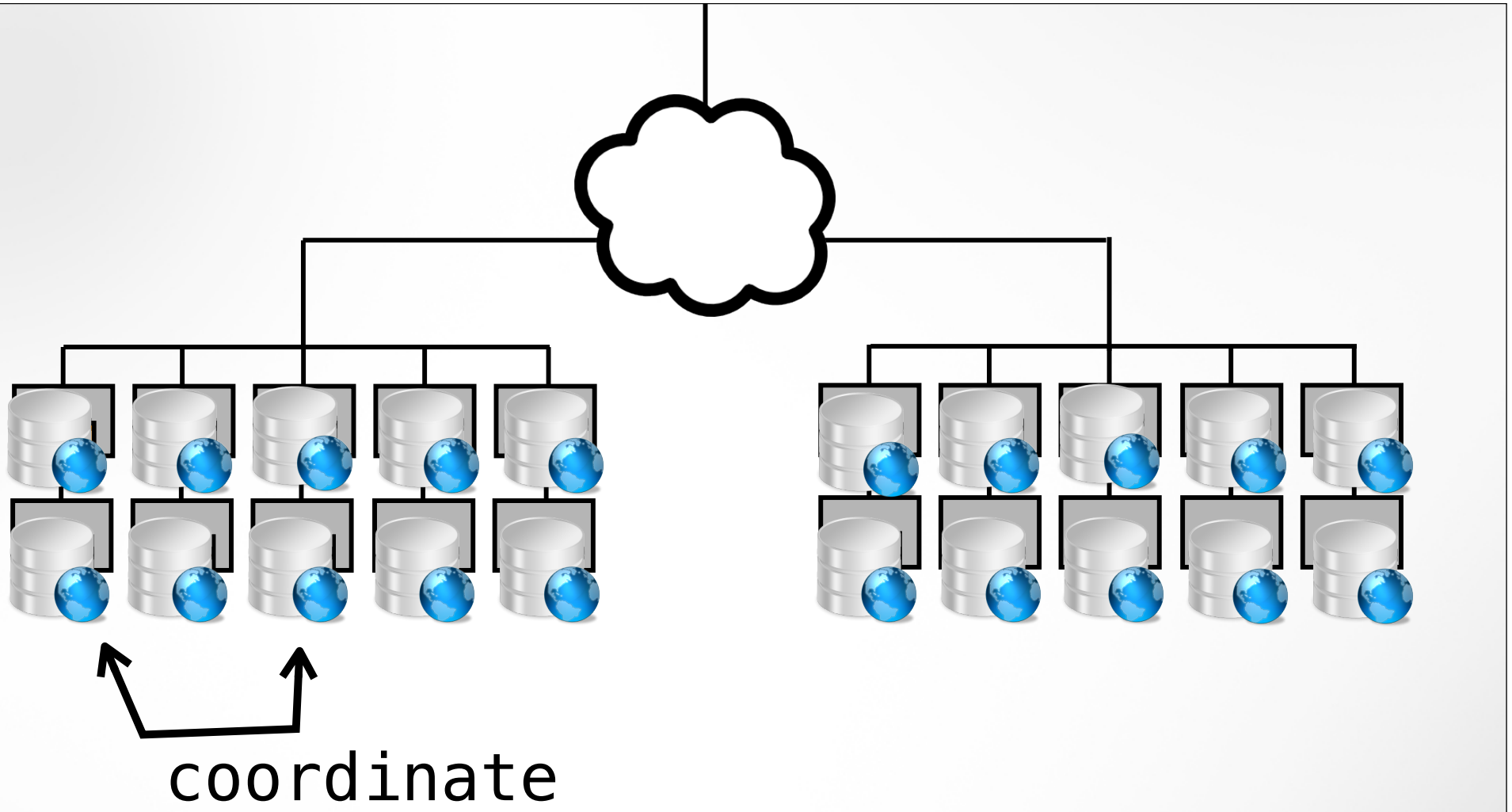
- Availability: Run in multiple data centers
- Scalability: Prepare the system for expected growth

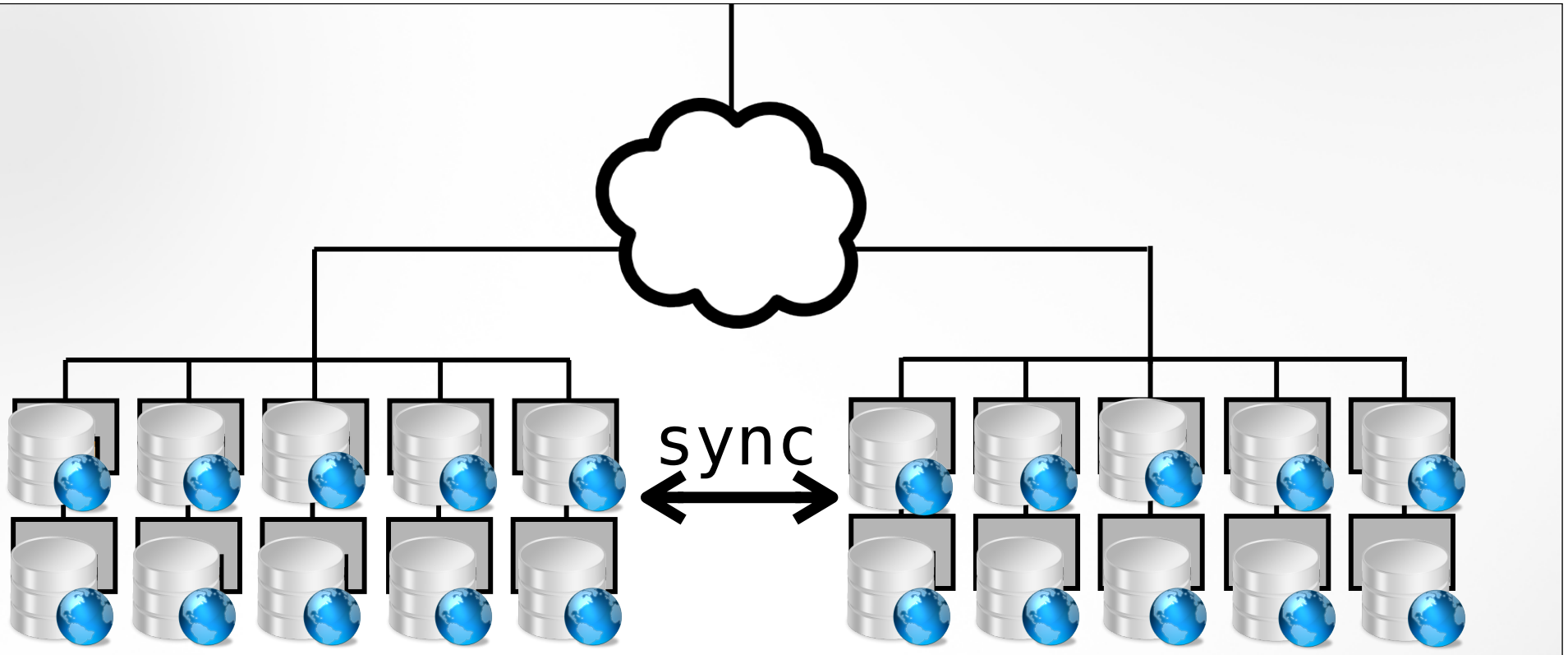
Riak Data Store

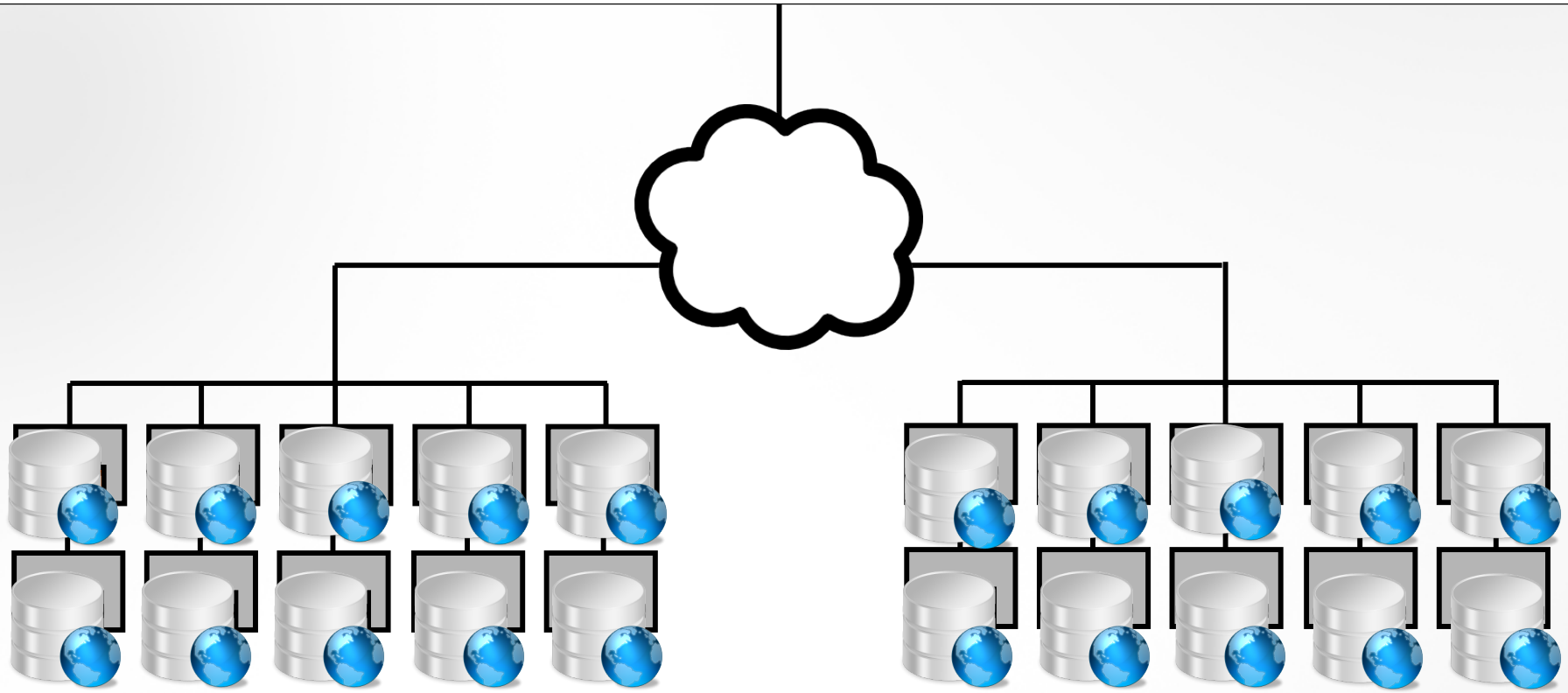
- Fit the general requirements
  - Availability + Scalability
  - Operational improvements
- Challenges
  - Key/Value Store, vs Relational Model
  - New technology, many unknowns





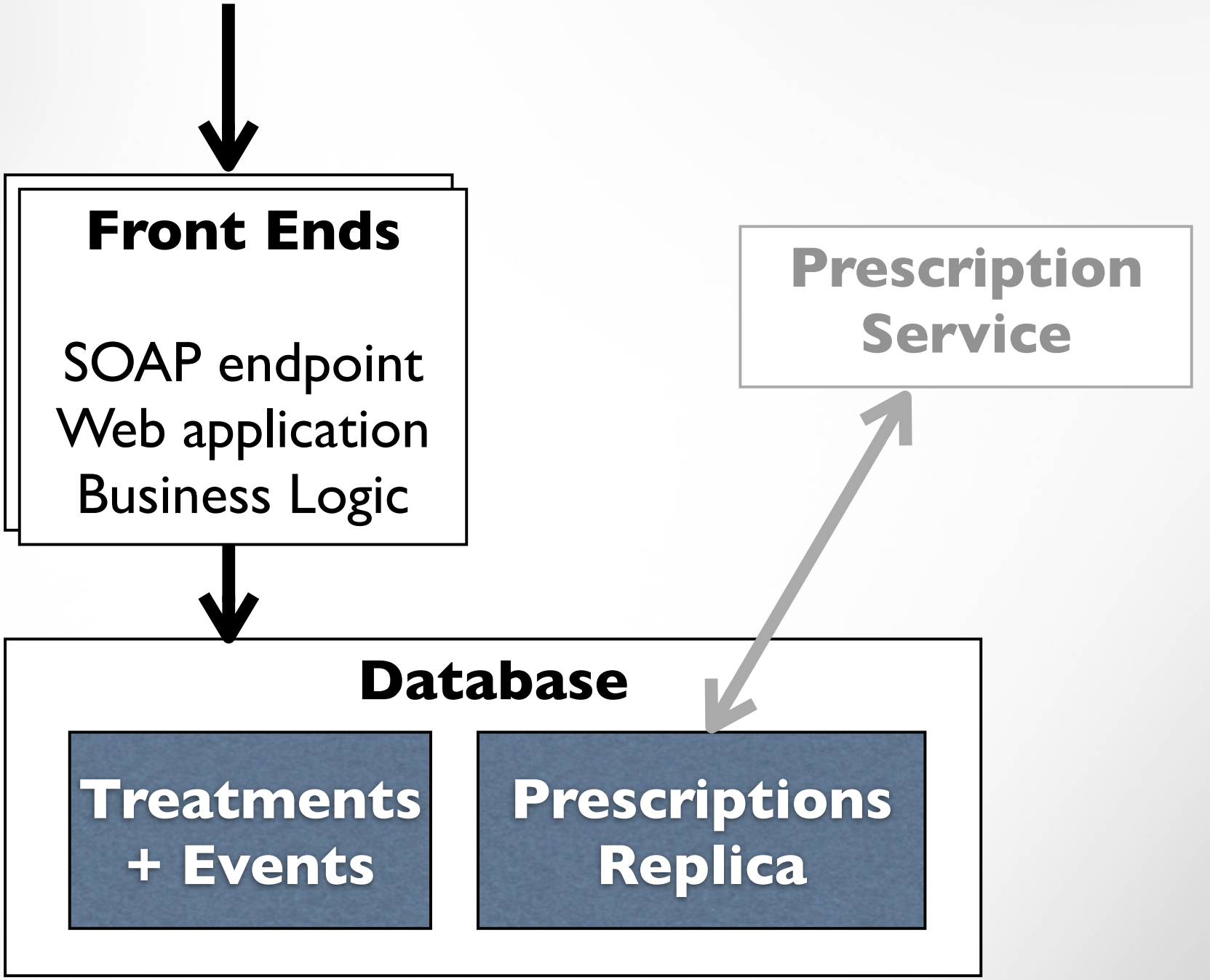

coordinate

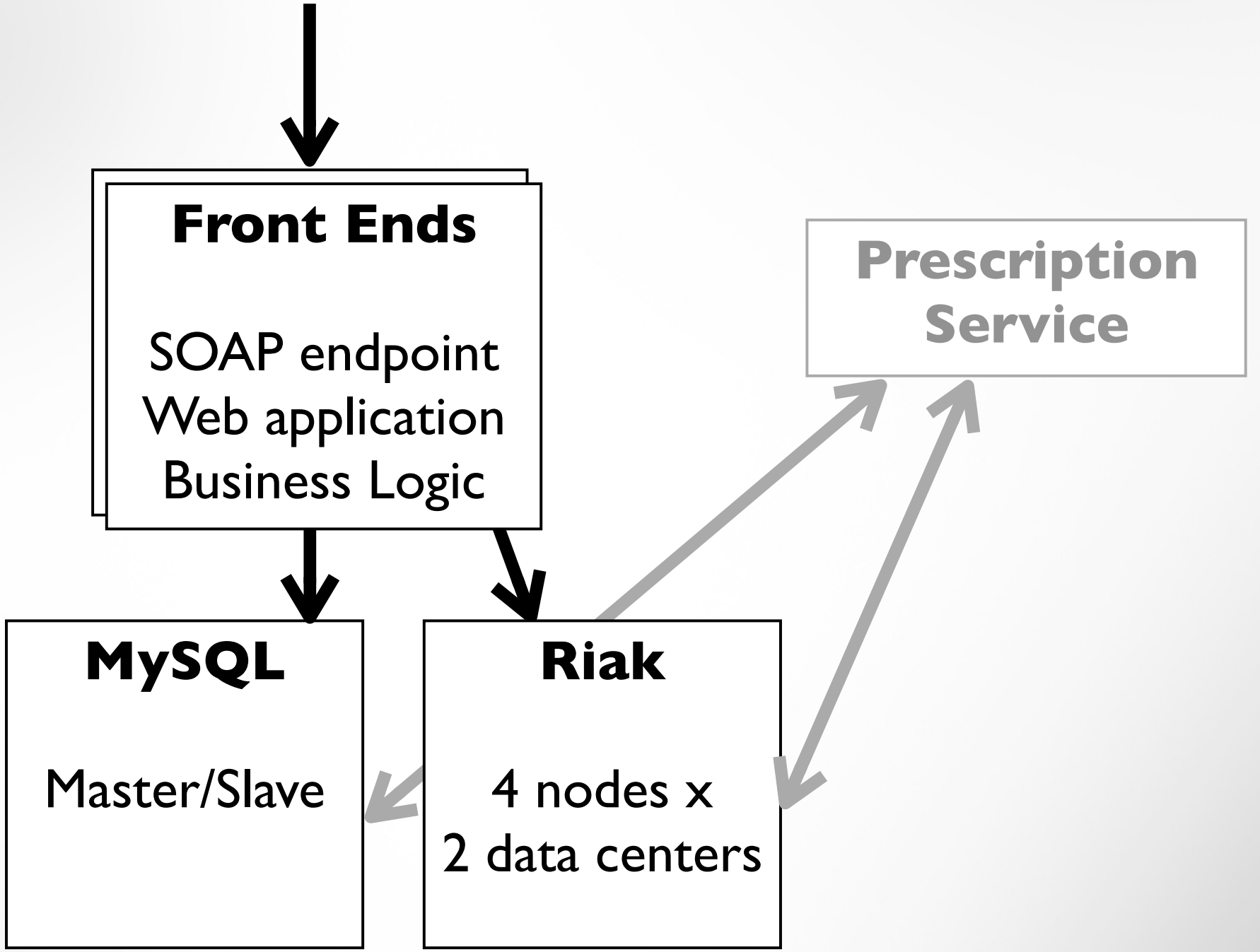

sync



riak

- scalable and available
- system captures write conflicts
- resolve lazily (read repair)


Front Ends
SOAP endpoint
Web application
Business Logic
Prescription Service
Database
Treatments + Events
Prescriptions Replica


Front Ends
SOAP endpoint
Web application
Business Logic
Prescription Service
MySQL
Master/Slave
Riak
4 nodes x
2 data centers

Challenges

- Data model: how to go from Relational model to Key/Value model
- Experiences with Riak's backends
- How to keep version history
- A true war story

Data Model

- Integrity without ACID transactions
- Riak's default storage keeps *all* keys in memory
- Dealing with Write Conflicts

Phase I

- To validate the architecture, we built a system where these are kept in Riak:
  - Prescription Replicas
  - Audit-log
  - Request cache

First Attempt: Using Links

~5 million

Person
Key: Person-ID
Links: Prescription-ID*

~200 million

Prescription
Key: Prescription-ID
Content: Protobuf+GZip

- Allows reading of **N** record in one roundtrip
- Performance suffered: **1+N** disk access
- Too many keys in memory

First Attempt: Using Links

~5 million

Person
Key: Person-ID
Links: Prescription-ID*

~200 million

Prescription
Key: Prescription-ID
Content: Protobuf+GZip

- Ran poorly on Virtual Hardware
- Trying to figure out how to handle conflicts

Second Take

~5 million

Prescriptions
Key: Person-ID
Content: Protobuf+GZip

- Very simple: read - resolve - modify - write
- Integrity: 1 person ↔ 1 record
- Performance good: **1** disk access
- All keys fit in memory

Read Repair

- On every read, we handle write conflicts
- If so, auto-merge[*], store and re-read
- Resolve: Merging is *business logic*; some merge actions need user attention, others don't.
- Forward: This is also the hook for schema evolution

The Audit Log

- ~1 billion log entries per year
  - Average 33/sec, peak 200/sec
- Stores generic JSON documents
- Need some search capability
- Bitcask backend was not an option

The Audit Log, Take I

- InnoDB backend [basically MySQL]
- Increasing keys for B-tree backend “YYYYMMDDhhmmss:<random-bits>”
- Indexing in SQL store

The Audit Log, Take II

- LevelDB Backend (SSTable)
- Riak Secondary Indexing
- Store JSON

The Audit Log, Take III

- HanoiDB / Log Structured B-Tree
  - On-disk, key-sorted, low memory
  - Predictable access times [merge/cleanup is incremental]
  - Secondary indexing, Auto expiry, Compression

Request Cache

- Makes SOAP-endpoints idempotent
- Keep Request/Response for 14 days
- Perfect fit for default Bitcask backend

A Real War Story...

- First production launch with Riak
- Strange data corruption started to appear
- Also spontaneous I/O errors sometimes
- ... we installed checksum hooks

```
0002280: 1e5d a8f6 5c18 7fac 468a 8e55 9851 1f6f  .]..\...F..U.Q.o
0002290: 617b 05ce 4a73 ba3d 29fc b034 396c 90c3  a{..Js.=)..49l..
00022a0: a7ea ff11 14f9 efcc 34e2 d80c 0834 c8d8  ........4....4..
00022b0: fb1f 5529 76bc 43cf 5cc6 b654 428d 2f29  ..U)v.C.\..TB./)
00022c0: b554 a2d3 5e98 a88f 928c c212 a177 9220  .T..^........w.
00022d0: c10b 06e6 d894 9d85 9266 3cfb fb6d 73ef  .........f<..ms.
00022e0: 4109 36fd d83d 0018 73d6 fb00 0050 56b9  A.6..=..s....PV.
00022f0: 002d 0800 4500 058c 3364 4000 4006 011e  .-..E...3d@.@...
0002300: 4df3 33ce 4df3 3136 d24a 1fa2 1341 ce84  M.3.M.16.J...A..
0002310: 6987 4397 5018 c210 c7ed 0000 c8c5 60f0  i.C.P.........`.
0002320: 9aba 0dfc cae6 70bb a06f 36c8 1c3b 00b2  ......p..o6..;..
0002330: 1a9e 1c62 87ce 8f3d c509 5ed3 f686 f1c7  ...b...=..^.....
0002340: 4784 f531 761b 3070 f0e0 4f12 d93f 00d9  G..1v.0p..O..?..
0002350: b9d3 f92f f2d8 faf5 ec31 9cff c3f2 5494  .../.....1....T.
0002360: 0f3b 3c18 ffcd b441 799a 90bc 9454 f25b  .;<....Ay....T.[
0002370: 1820 67d6 24b8 5a91 c0a8 d9a2 df0c 7b5e  . g.$.Z.......{^
```

```
0002280: 1e5d a8f6 5c18 7fac 468a 8e55 9851 1f6f  .]..\...F..U.Q.o
0002290: 617b 05ce 4a73 ba3d 29fc b034 396c 90c3  a{..Js.=)..49l..
00022a0: a7ea ff11 14f9 efcc 34e2 d80c 0834 c8d8  ........4....4..
00022b0: fb1f 5529 76bc 43cf 5cc6 b654 428d 2f29  ..U)v.C.\..TB./)
00022c0: b554 a2d3 5e98 a88f 928c c212 a177 9220  .T..^........w.
00022d0: c10b 06e6 d894 9d85 9266 3cfb fb6d 73ef  .........f<..ms.
00022e0: 4109 36fd d83d 0018 73d6 fb00 0050 56b9  A.6..=..s....PV.
00022f0: 002d 0800 4500 058c 3364 4000 4006 011e  .-..E...3d@.@...
0002300: 4df3 33ce 4df3 3136 d24a 1fa2 1341 ce84  M.3.M.16.J...A..
0002310: 6987 4397 5018 c210 c7ed 0000 c8c5 60f0  i.C.P.........`.
0002320: 9aba 0dfc cae6 70bb a06f 36c8 1c3b 00b2  ......p..o6..;..
0002330: 1a9e 1c62 87ce 8f3d c509 5ed3 f686 f1c7  ...b...=..^.....
0002340: 4784 f531 761b 3070 f0e0 4f12 d93f 00d9  G..1v.0p..O..?..
0002350: b9d3 f92f f2d8 faf5 ec31 9cff c3f2 5494  .../.....1....T.
0002360: 0f3b 3c18 ffcd b441 799a 90bc 9454 f25b  .;<....Ay....T.[
0002370: 1820 67d6 24b8 5a91 c0a8 d9a2 df0c 7b5e  . g.$.Z.......{^
```

A Real War Story

- The problem was a buggy Solaris/VMWare network driver [client machines]
- TCP checksumming is very simple
- $1/2^{16}$ packets was let thru - MD5 caught it
- Also the reason for I/O dropped connections

Phase I: Conclusions

- 3 data sets - 3 different solutions
- Availability & Scalability
- Response times are better and more predictable
- Before: Locked at max # ops/sec
- Now: 4 x ops/sec ... and can scale more