Bringing Riak to the Mobile Platform

Kresten Krab Thorup
Hacker

@drkrab

TRIFORK.

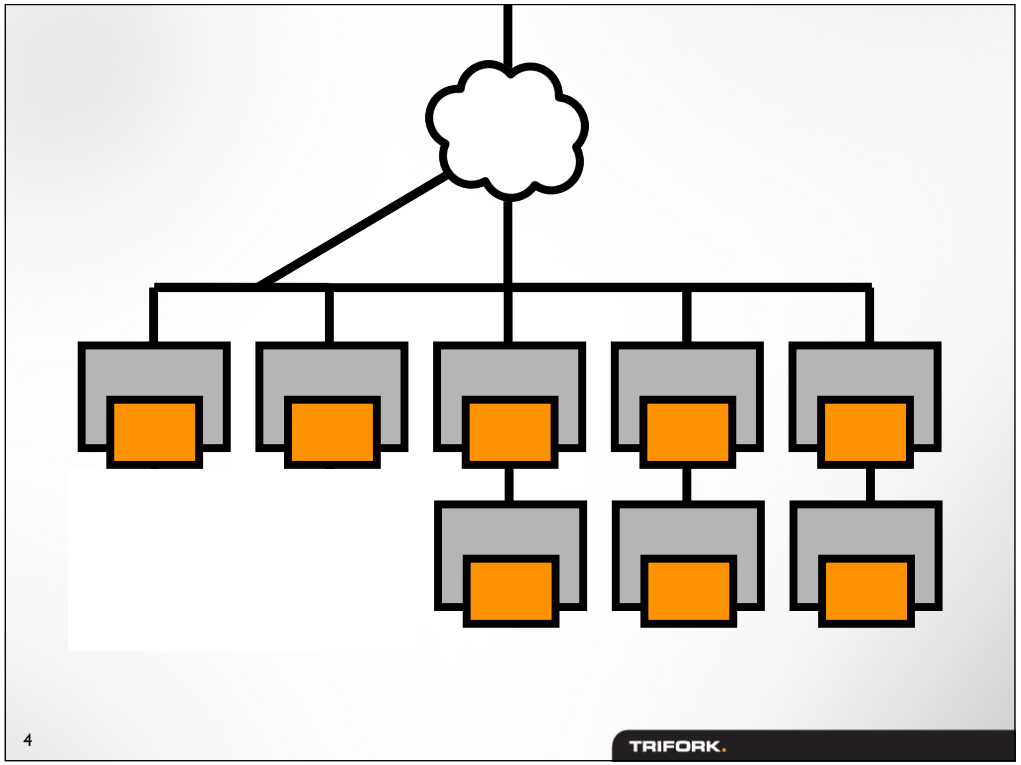
Outline

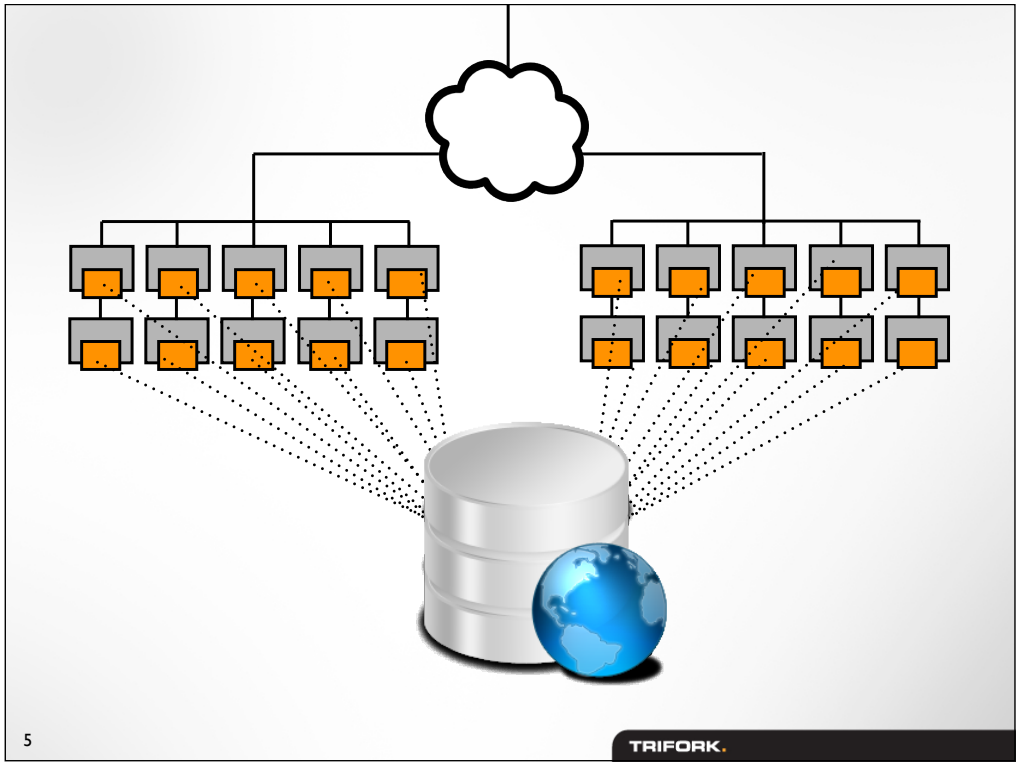
- Riak and it's Data Model
- RiakSync, a protocol for Key/Value synchronization
- RiakMobile, Riak clients for mobile

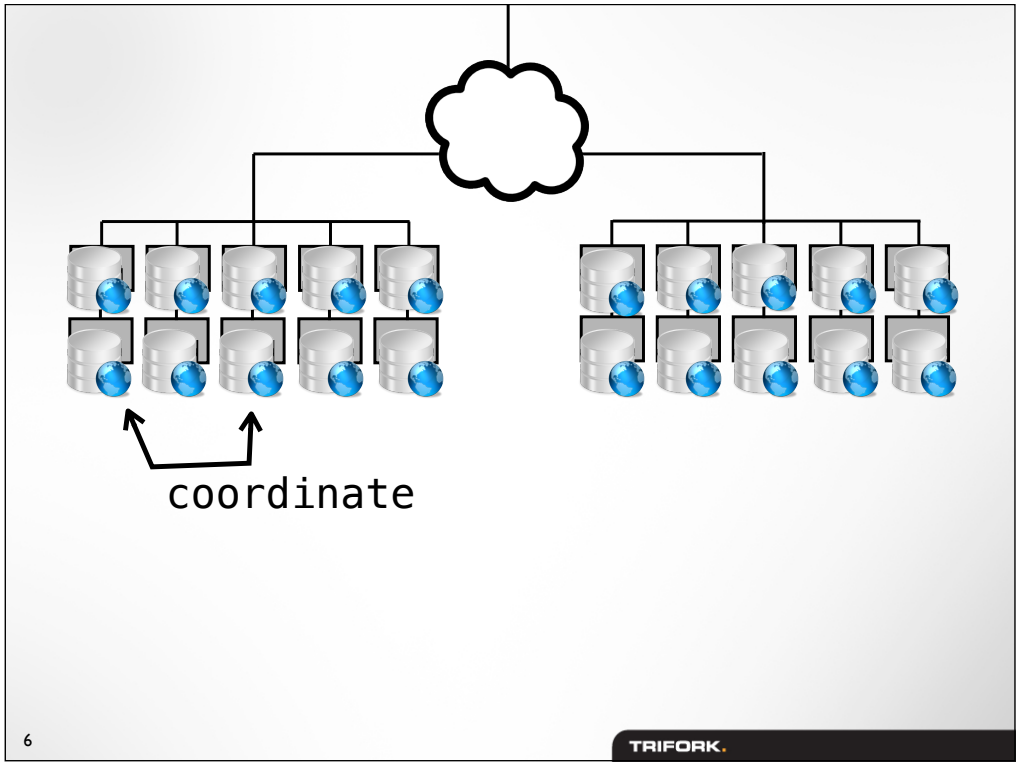
What is Riak, anyway?

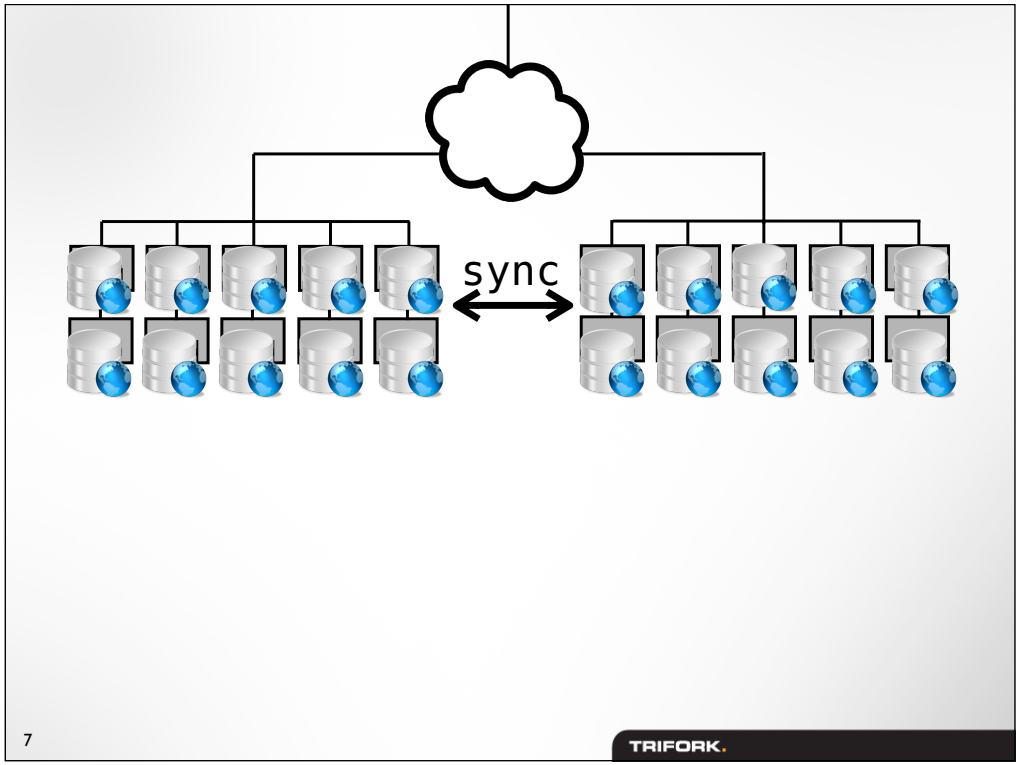
@drkrab

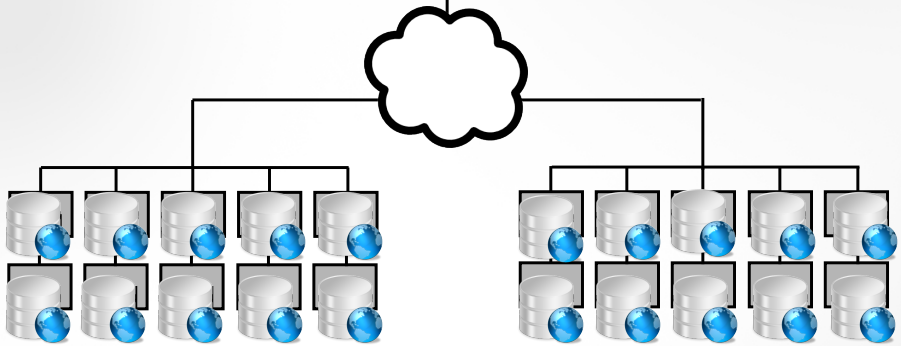
TRIFORK.






coordinate

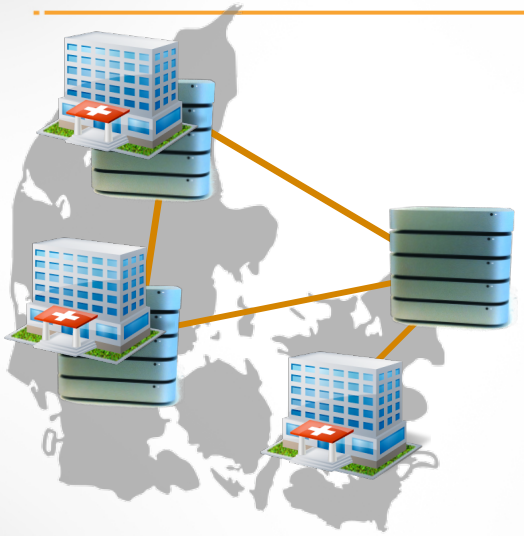

sync
7
TRIFORK.

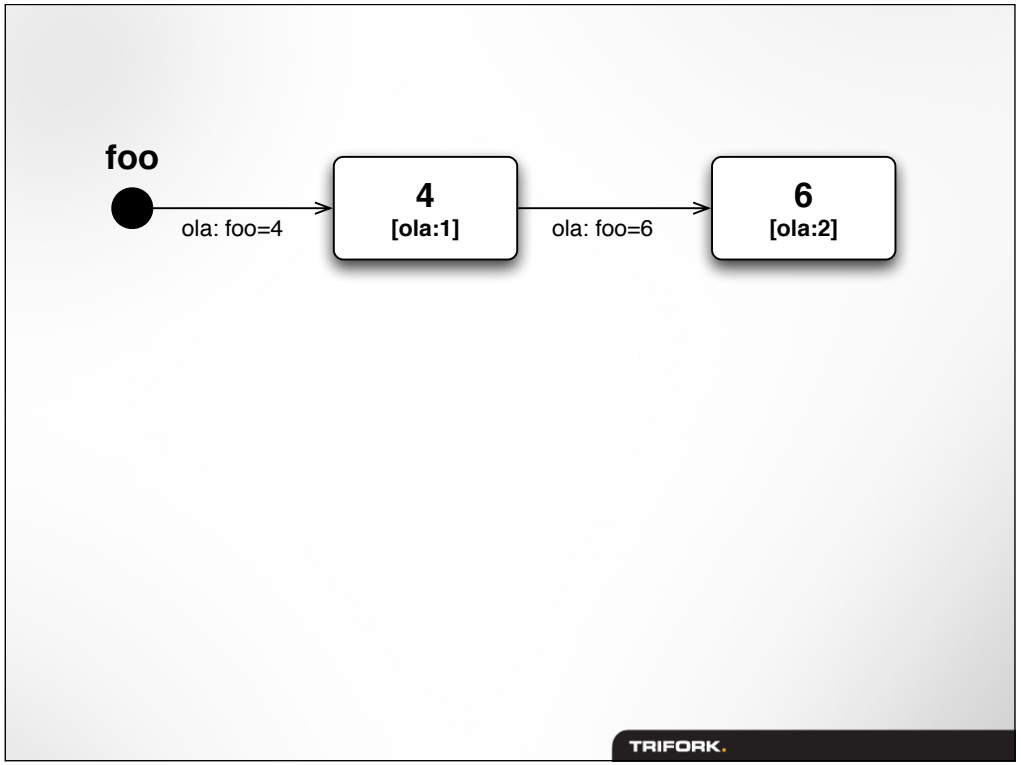


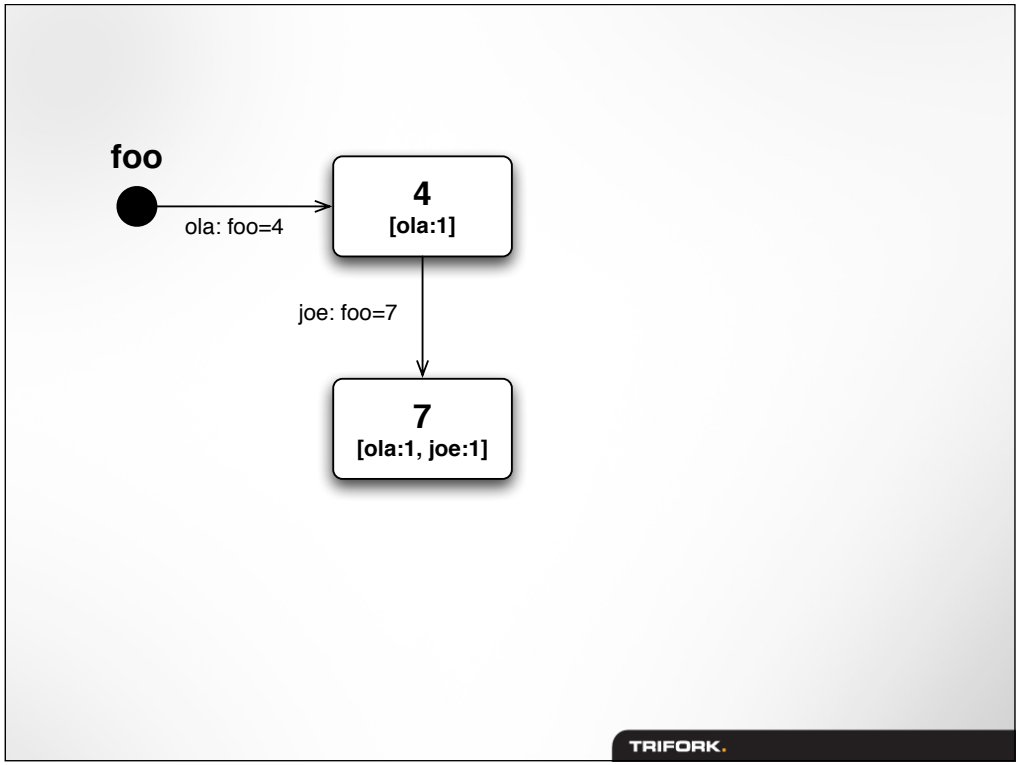
riak

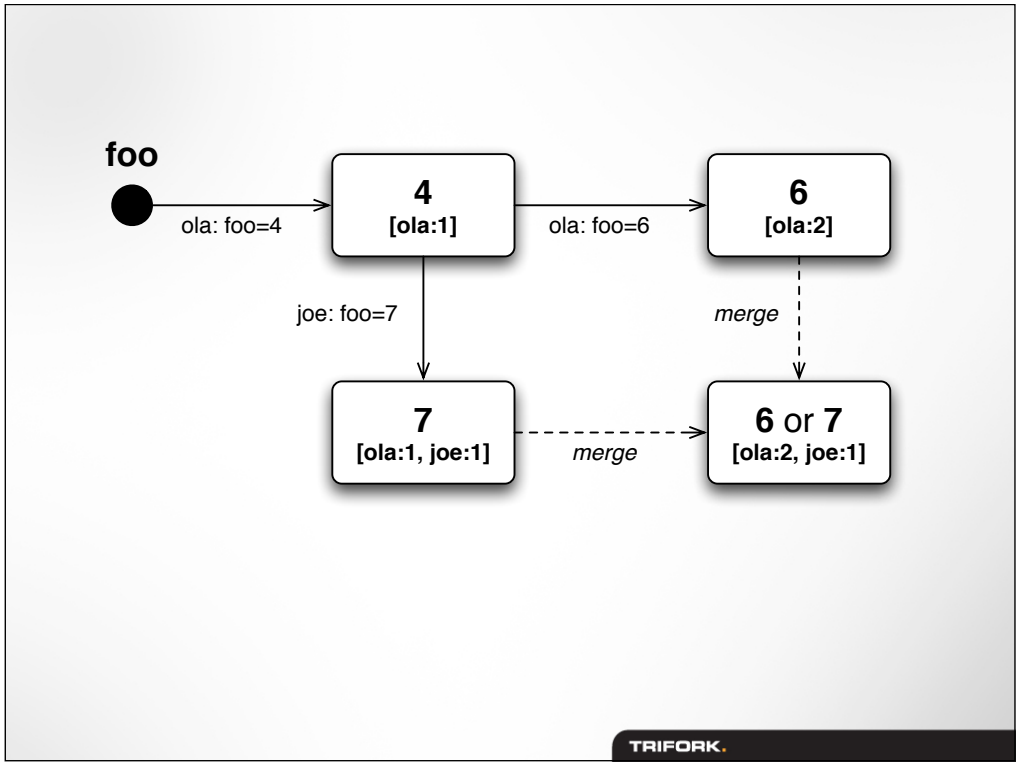
- write-availability
- system captures write conflicts
- resolve lazily (read repair)

Shared Medicine Card




foo
ola: foo=4
4
[ola:1]
ola: foo=6
6
[ola:2]
TRIFORK.


foo
ola: foo=4
4
[ola:1]
joe: foo=7
7
[ola:1, joe:1]
TRIFORK.


foo
ola: foo=4
4
[ola:1]
ola: foo=6
6
[ola:2]
joe: foo=7
merge
7
[ola:1, joe:1]
merge
6 or 7
[ola:2, joe:1]
TRIFORK.

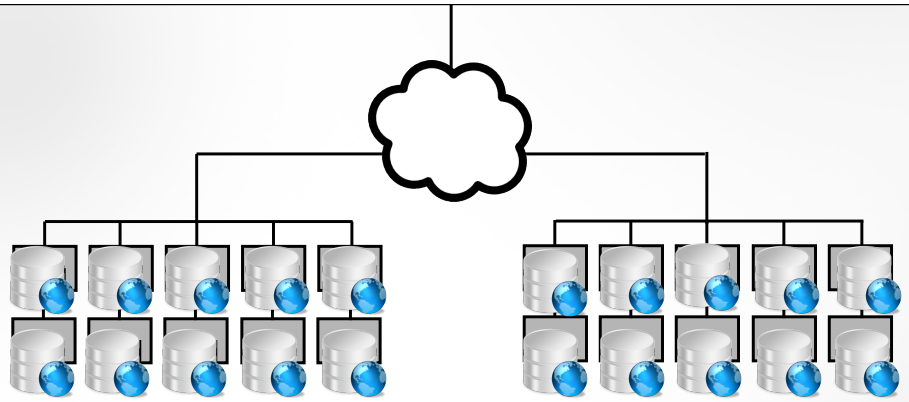
Pseudo Code!

Riak API review

```
connect(ClientID) -> Client
```

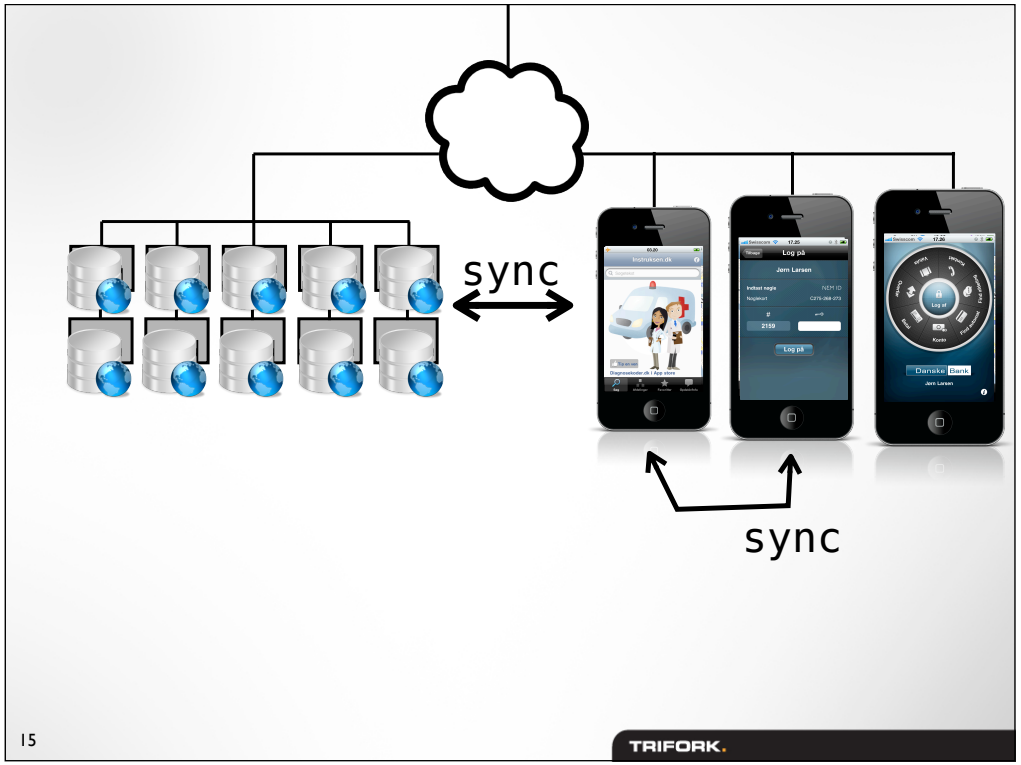
```
key()   :: {Bucket, Key}
datum() :: {ContentType, RawData}
```

```
Client:insert(key(), datum()) -> {ok, VClock} | ...
Client:lookup(key()) -> {ok, VClock, [datum()]} | ...
Client:update(key(), VClock, datum()) -> ok | ...
Client:delete(key(), VClock) -> ok | ...
```



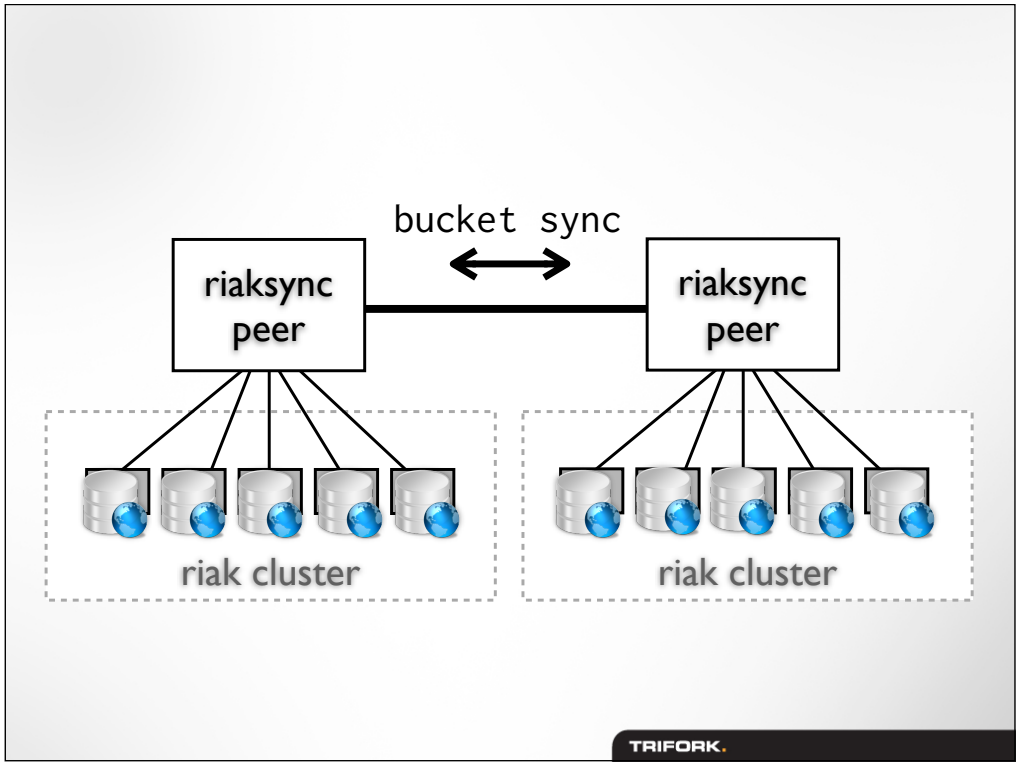
riak

- write-availability
- system captures write conflicts
- resolve lazily (read repair)


sync
sync

Today: Two Things

- **RiakSync**: Protocol for Riak bucket replication
- **RiakMobile**: Java & C++ based client implementations


bucket sync
riaksync peer
riaksync peer
riak cluster
riak cluster
TRIFORK.

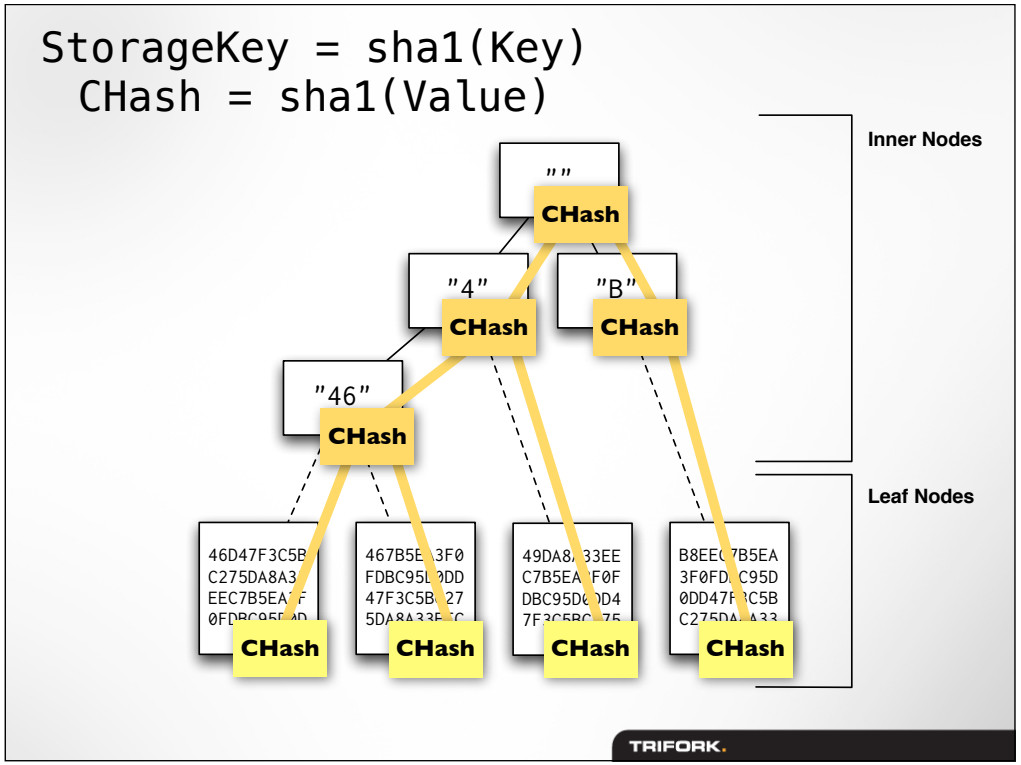
```
LocalStore = riak:client_connect('riak@127.0.0.1')

%% sync via protobuf socket
> riak_sync:sync(LocalStore, <<"bucket">>,
                 "172.1.35.204", 8082)


%% Other site has riak_sync running...
(riak@172.1.35.204)1> riak_sync:start()
```

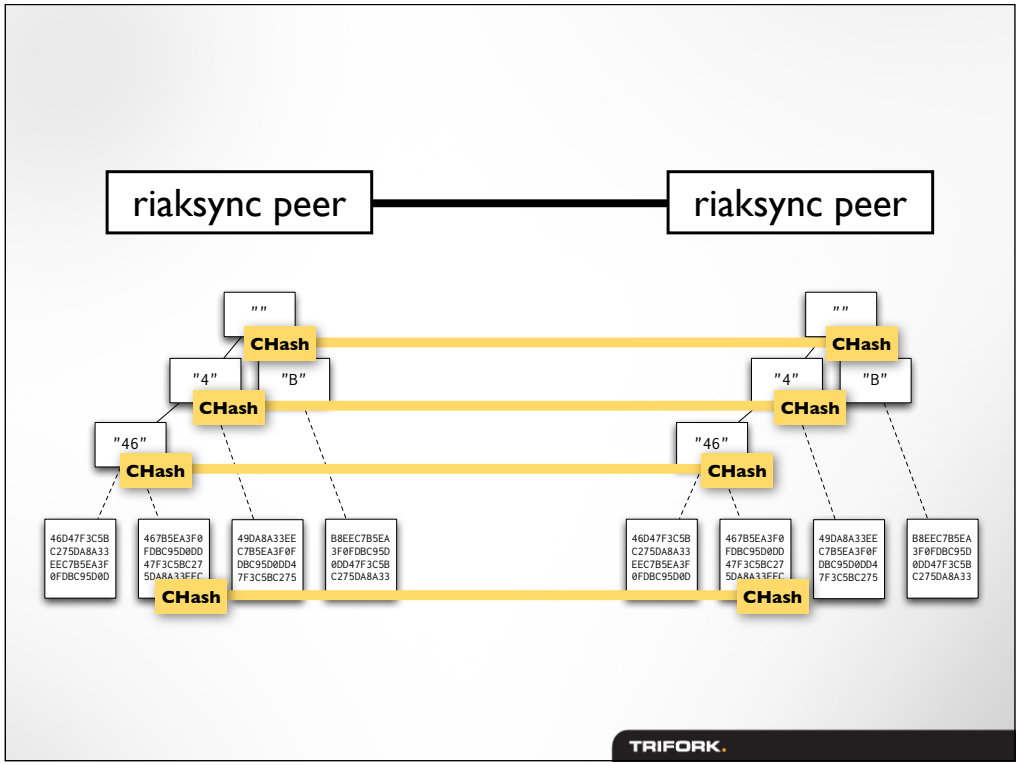
```
{riak_sync, [{pb_ip, "172.1.35.204"},
             {pb_port, 8082},
             {riak, 'dev@127.0.0.1'}]}
```

Riak Sync Protocol

- Riak bucket replication
- Suitable for up to ~100.000 objects/bucket
- Asymmetric work load
- Designed for high-latency networks

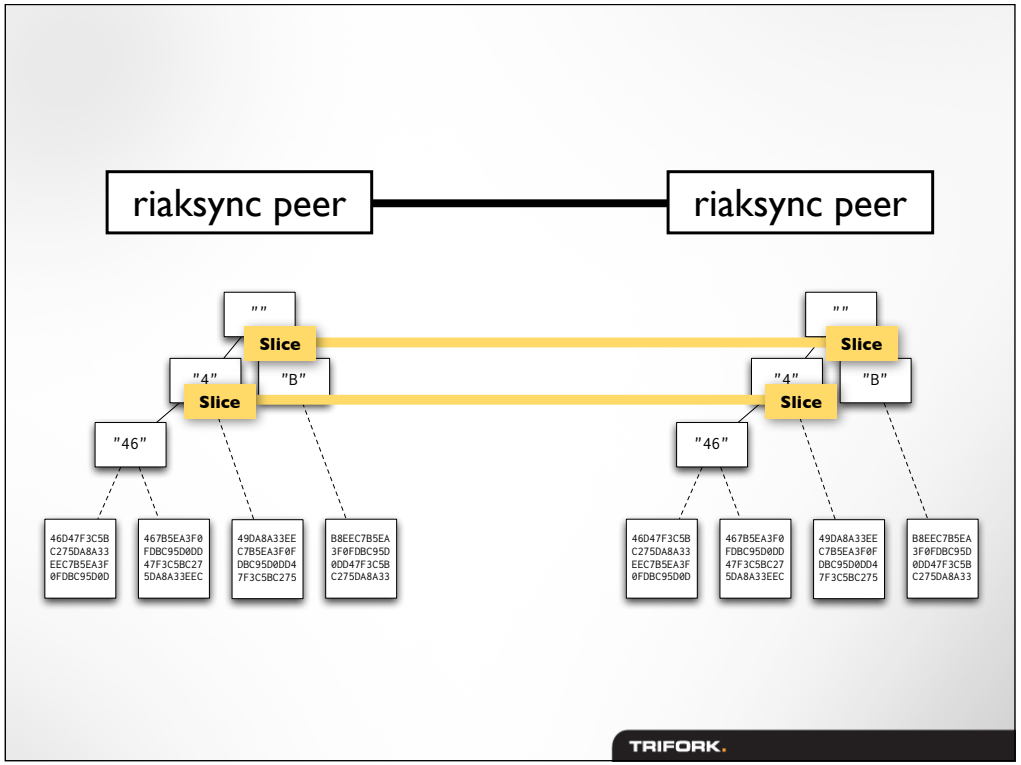
- **Very different design criteria than normal Riak cluster-cluster replication**


StorageKey = sha1(Key)
CHash = sha1(Value)
""
CHash
"4"
CHash
"B"
CHash
"46"
CHash
46D47F3C5B
C275DA8A3
EEC7B5EA
467B5E 3F0
FDBC95L DD
47F3C5B 27
49DA8 33EE
C7B5EA F0F
DBC95D DD4
B8EE 7B5EA
3F0FD C95D
0DD47F 3C5B
CHash
CHash
CHash
CHash
Inner Nodes
Leaf Nodes
TRIFORK.


riaksync peer
riaksync peer
""
CHash
"4"
"B"
CHash
"46"
CHash
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33FFC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
CHash
""
CHash
"4"
"B"
CHash
"46"
CHash
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33FFC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
CHash
TRIFORK.

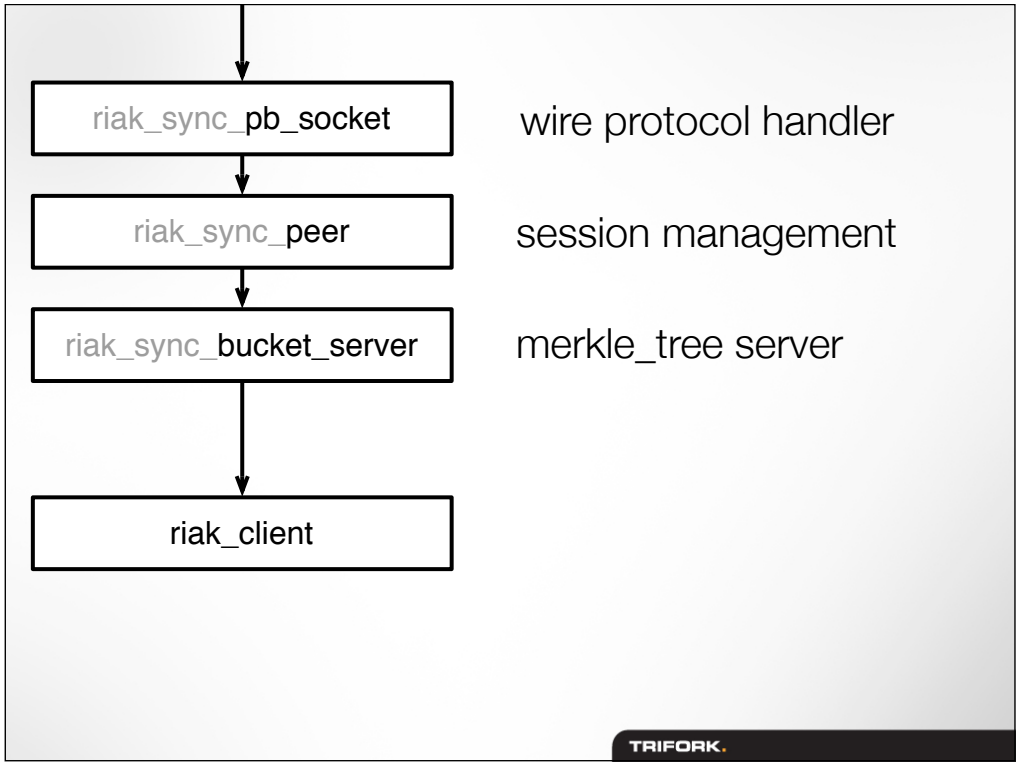
HashTree Slices

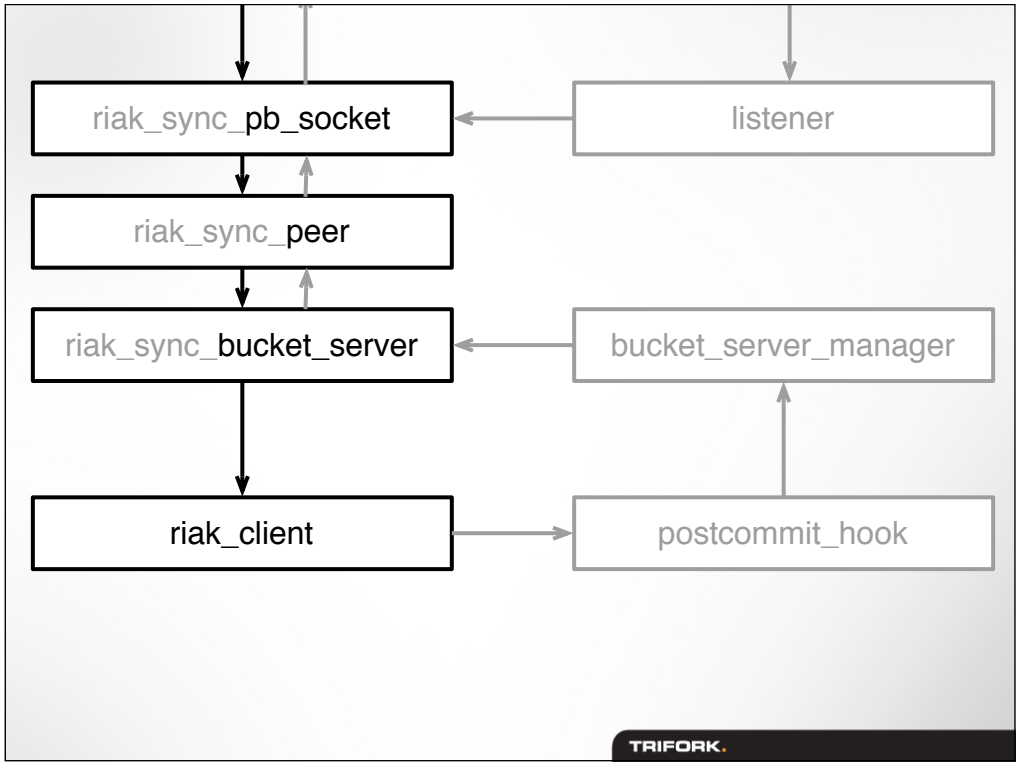
```
slice_msg() ::
   {slice, Path, CHash,
    [{Nibble :: 0..16, shallow_node()}]}.

shallow_node() ::
   {inner, CHash}
 | {leaf, [{Key, VClock, CHash}, ...] }.
```


riaksync peer
riaksync peer
""
Slice
"4"
Slice
"B"
"46"
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33EEC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
""
Slice
"4"
Slice
"B"
"46"
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33EEC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
TRIFORK.

RiakSync Protocol

- Asynchronous
- Parallelizes quickly (client does breadth first)
- Server peer reorders requests to reduce session state
  - **First do any put/get requests**
  - **The execute slice requests in depth first order**
- Packet size (slice messages) ~ 1000 bytes
- Client needs (almost) no session state

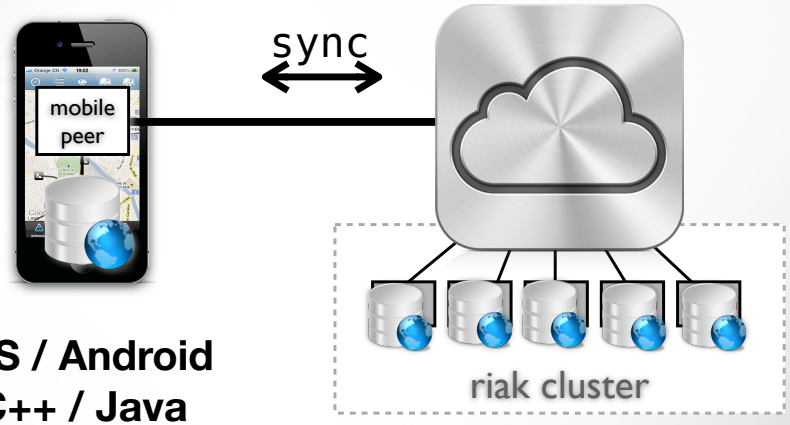

riak_sync_pb_socket
wire protocol handler
riak_sync_peer
session management
riak_sync_bucket_server
merkle_tree server
riak_client
TRIFORK.


riak_sync_pb_socket
listener
riak_sync_peer
riak_sync_bucket_server
bucket_server_manager
riak_client
postcommit_hook
TRIFORK.

Interesting Issues

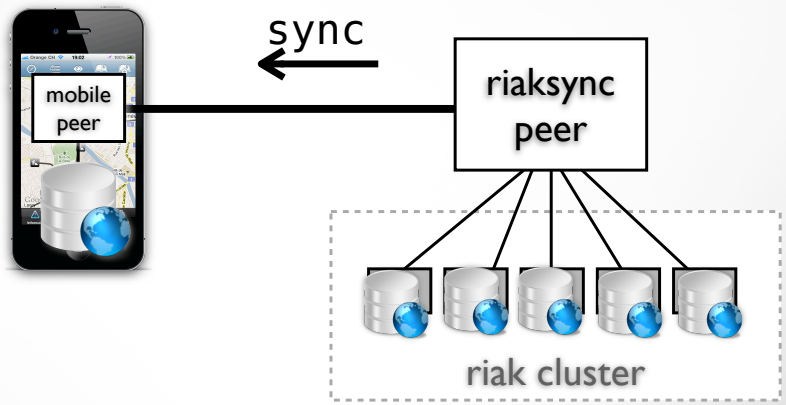
- Capturing deletes
  - **RiakSync's `postcommit` hook creates records of deleted data to maintain vector clocks for those**
  - **If a client comes along a week after the delete, it will thus not re-create the deleted record.**
- Cheating riak's vector clocks
  - **In Riak, you can only "put and increment vclock" but RiakSync needs "put sans vclock increment".**

Riak Mobile

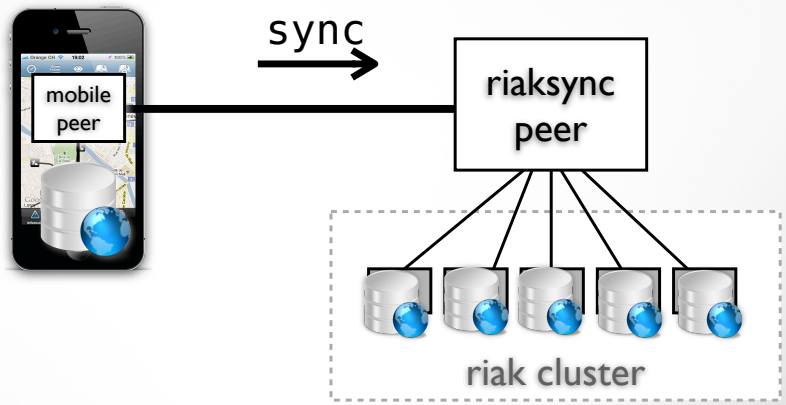


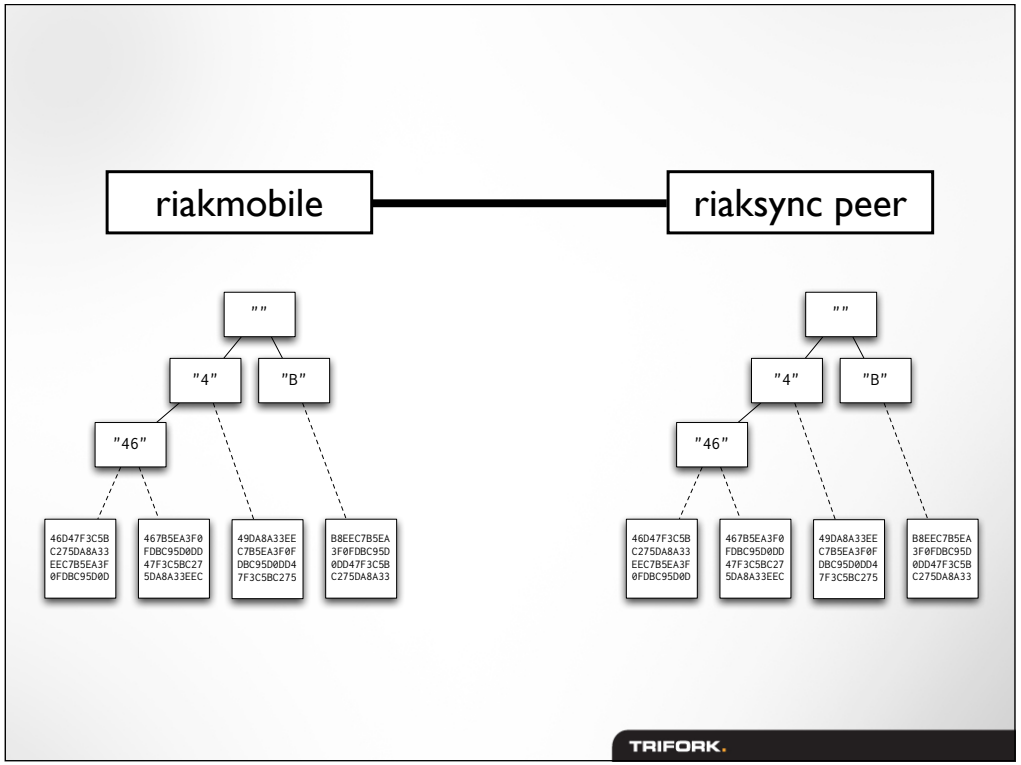
iOS / Android
C++ / Java

Content Distribution

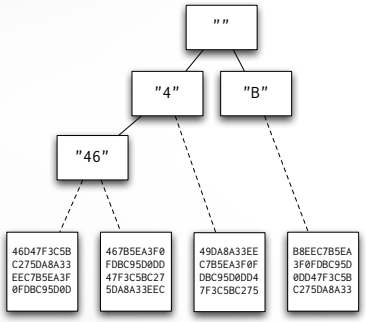


Reliable Queueing

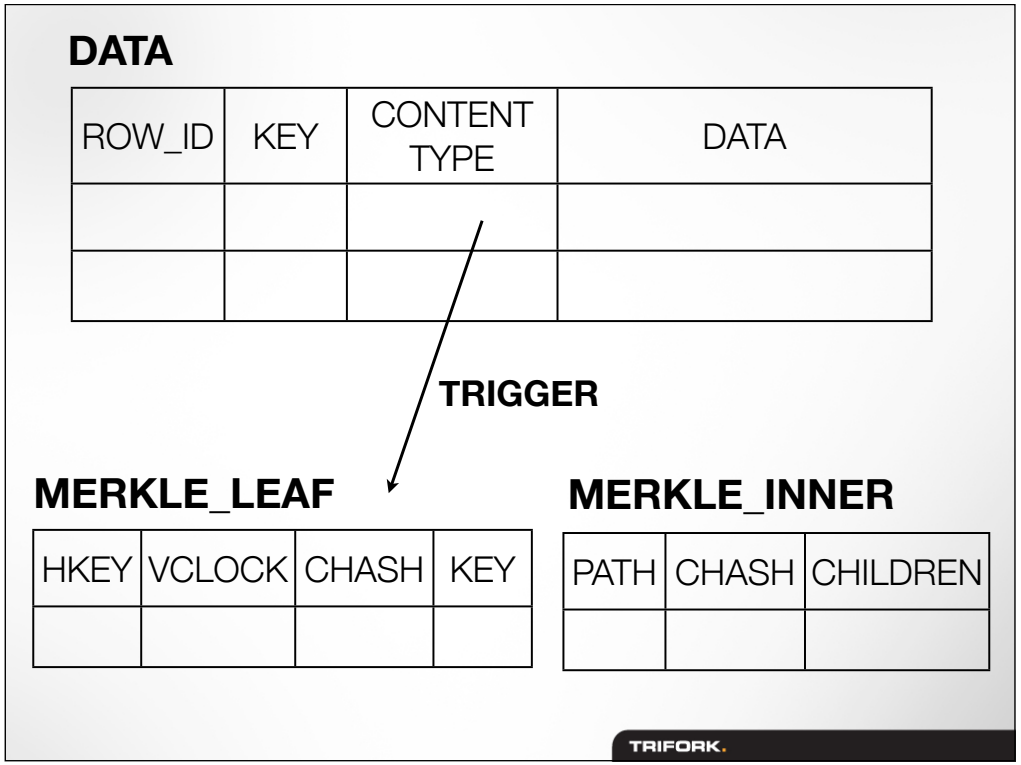



riakmobile
riaksync peer
""
"4"
"B"
"46"
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33EEC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
""
"4"
"B"
"46"
46D47F3C5B C275DA8A33 EEC7B5EA3F 0FDBC95D0D
467B5EA3F0 FDBC95D0DD 47F3C5BC27 5DA8A33EEC
49DA8A33EE C7B5EA3F0F DBC95D0DD4 7F3C5BC275
B8EEC7B5EA 3F0FDBC95D 0DD47F3C5B C275DA8A33
TRIFORK.

riakmobile



- Client stores data in local data store as a HASH TREE
- Always “ready to sync” (no session state, no cpu/memory intensive computation)


DATA
ROW_ID
KEY
CONTENT TYPE
DATA
TRIGGER
MERKLE_LEAF
HKEY
VCLOCK
CHASH
KEY
MERKLE_INNER
PATH
CHASH
CHILDREN
TRIFORK.

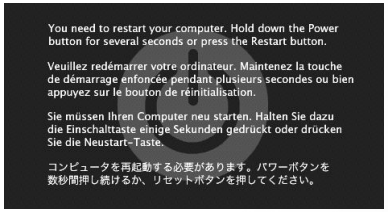
```
class RiakClient {

 bool insert(const string& key,
             /*out*/ VClock& version,
               const Datum&  value);

 bool lookup(const string& key,
             /*out*/ VClock& version,
             /*out*/ Data&   values);

 void update(const string& key,
               const VClock& version,
               const Datum&  value);
}

class Data {
  Datum[] datas;
  int count;
}
```

```
class Datum {
  string content_type;
  string body;
}
```

```
class RiakClient {  ...continued...

 // create client for bucket
 RiakClient(string& bucket);

 // initiate RiakSync, and receive live updates
 void connect(string& host, int port);

 // disconnect from RiakSync server
 void disconnect();

 // observe updates
 void set_commithook(update_hook callback);

}

typedef void (*update_hook)(const string& key,
                            const Data&   data);
```

Summary

- Riak Data Model
- RiakSync, a protocol for Key/Value synchronization
- RiakMobile, Riak clients for mobile



Thank You

@drkrab

Thank You

@drkrab