riak

basho

What am I ?

- Bryan Hunt
- Basho Client Services Engineer
- Erlang neophyte
- JVM refugee
- Be gentle

What are you?

- Developer
- Operations
- Other

Structure of this talk

- Introduction to Riak
- Introduction to Riak 2.0
- Riak 2.0 Features
- Example uses

Introduction to Riak

What is Riak?

Key	Value
Key	Value
Key	Value
Key	Value
Key	Value
Key	Value
Key	Value
Key	Value

Riak is the
ops-friendly
database

Runs on everything

FreeBSD®

debian

SmartOS

solaris

fedora

redhat.

ubuntu

Except Windows

Cluster of DISTRIBUTED nodes

Performance through concurrency

All nodes participate equally

MASTERLESS

No **single** point of failure

Easily add or remove nodes

SCALABLE

Linear scalability

Replicas of stored data

HIGHLY AVAILABLE

Erlang core

FAULT TOLERANT

self healing

So what?

- Simple deployment model
- Predictable performance
- Easy scaling
- Less tedium
- More sleep

Introduction to Riak 2.0

The Swiss Army Database

Nope. Not like this

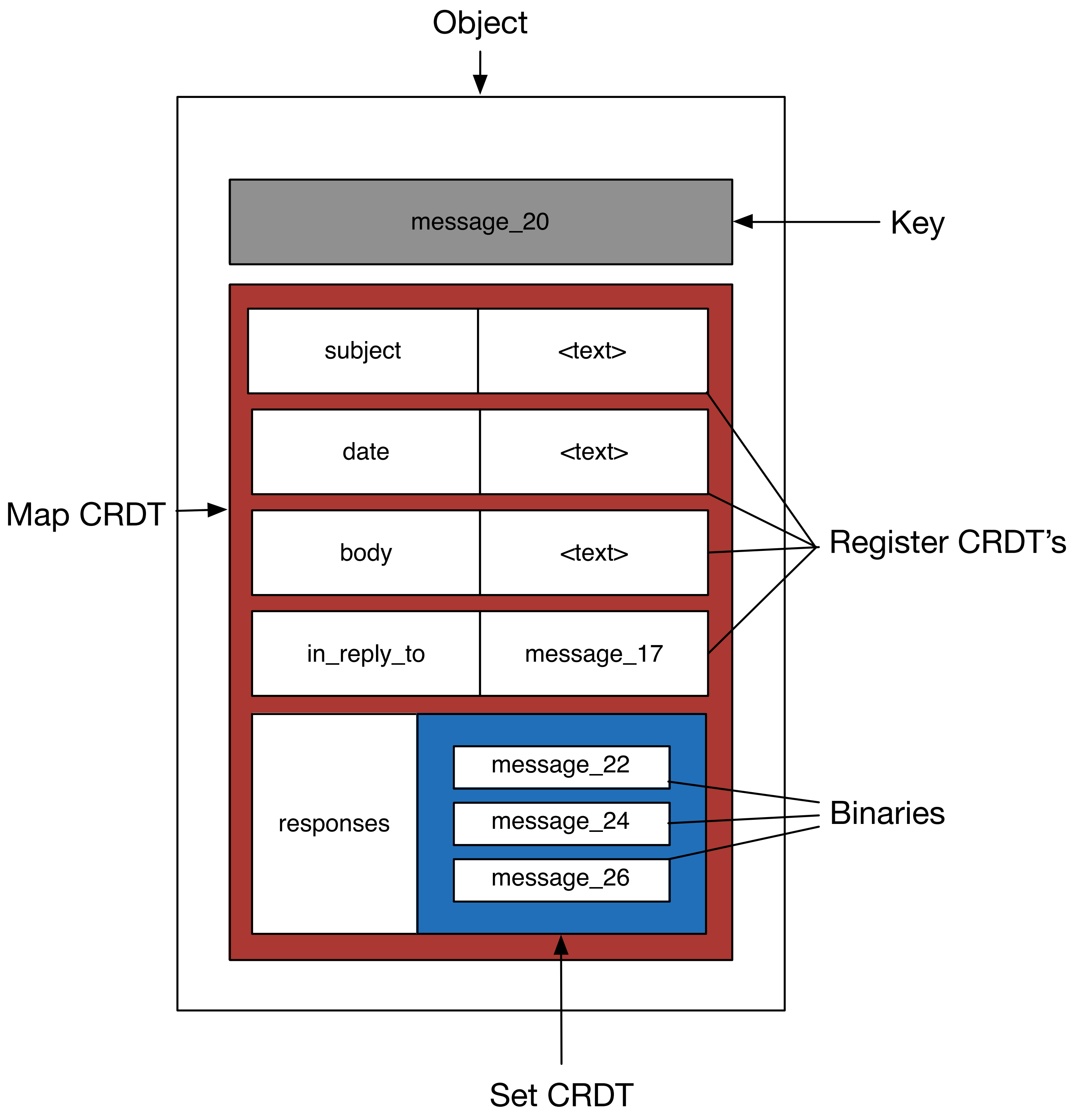
Actually, more like this..

Riak 2.0 Features

Riak 2.0 key features

- Riak Data Types (CRDT’s)
- Full-Text Search (Yokozuna)
- Security
- Simplified Configuration (Cuttlefish)
- Reduced Replicas for Multiple Data Centers
- Strong Consistency

Riak Data Types


Object
message_20
Key
subject
<text>
date
<text>
body
<text>
in_reply_to
message_17
Map CRDT
Register CRDT’s
responses
message_22
message_24
message_26
Binaries
Set CRDT

CRDT's - how was it before ?

- Client side conflict resolution (siblings)
- All objects was opaque to RIAK

CRDT’s - Simple use cases

- Increment a value
- Append values to an object
- Batch add or remove multiple associated objects

CRDT's - incrementing concurrently - before (1)

Client 1

- GET /riak/pints_sold
- Deserialize/increment/Serialize
- PUT /riak/pints_sold

CRDT's - incrementing concurrently - before (2)

Client 2

- GET /riak/pints_sold
- Deserialize/increment/Serialize
- PUT /riak/pints_sold - CONFLICT !!! BOOM !

CRDT's - incrementing concurrently - before (3)

Client 2

- GET /riak/pints_sold (both siblings)
- Deserialize/Merge/Serialize
- PUT /riak/pints_sold

Boring!

CRDT’s - incrementing concurrently - now

- Create a bucket-type with the data-type 'counter'
- Active the bucket-type
- Initialize the bucket
- Send increment or decrement commands to the server

CRDT's - how we used to append to an object

1. Fetch
2. Deserialize
3. Append
4. Store
5. Conflict GOTO 1

CRDT’s - how we now append to an object

- Create a bucket-type with the data-type ’set'
- Active the bucket-type
- Initialize the bucket
- Send add, remove, add_all, and remove_all commands to the server

CRDT’s - complex nested data

how we used to do it

CRDT's - complex nested data (now)

- Conflict resolution is handled on the server
- Manipulate remote data structures by sending update commands to Riak
- Avoids client-side roundtrip
- Reduces write contention
- It's just easier

Yokozuna

AKA Search 2.0

- Full-text search
- Integration with Apache Solr

Yokozuna

How did we search before ?

- Original Riak Search
- Secondary indexes (2i)
- Map-Reduce

Yokozuna

Original Riak search

- Implemented in Erlang
- Subset of Solr functionality
- Perpetually chasing feature parity

Yokozuna

2i search

- Two types of secondary attributes:
  integers and strings (aka binary).
- Querying by exact match or range on one index.
- Index is defined at object creation time

Yokozuna

Limitations of 2i

- No full-text (term based) query capability.
- Composite queries require multiple range queries
- Not supported on bitcask, only leveldb and memory

Yokozuna

MapReduce

- Not suitable for real-time querying
- Designed for scheduled analytics
- Not a search engine

Security

```
riak-admin security enable
```

Security Authentication

- Trust
- Password file
- PAM
- Certificate

basho

Security Authorization

- Per bucket
- Per operation

- GET
- PUT
- DELETE
- INDEX

Cuttlefish

Simplified Configuration Management

Cuttlefish - how it was

The old configuration file format was a huge list of terms.

```
 %% Riak Core config
 {riak_core, [
            %% Default location of ringstate
            {ring_state_dir, “./data/ring"},
           %% Default ring creation size.  Make sure it is
a power of 2,
            %% e.g. 16, 32, 64, 128, 256, 512 etc
            %{ring_creation_size, 64}, ad infinitum….
```

Cuttlefish - now

The new configuration file format

```
%% implicit scope
ring_size = 64
%% explicit scope
foo.bar.baz = “alice"
```

Cuttlefish - so what?

Cuttlefish payoff - UNIX admin

```
sed -i'' -e '/ring_size.*=/{s_.*_ring_size = 128_;}' ./**/etc/riak.conf
```

Cuttlefish payoff - Configuration Management

Let’s take Ansible as an example

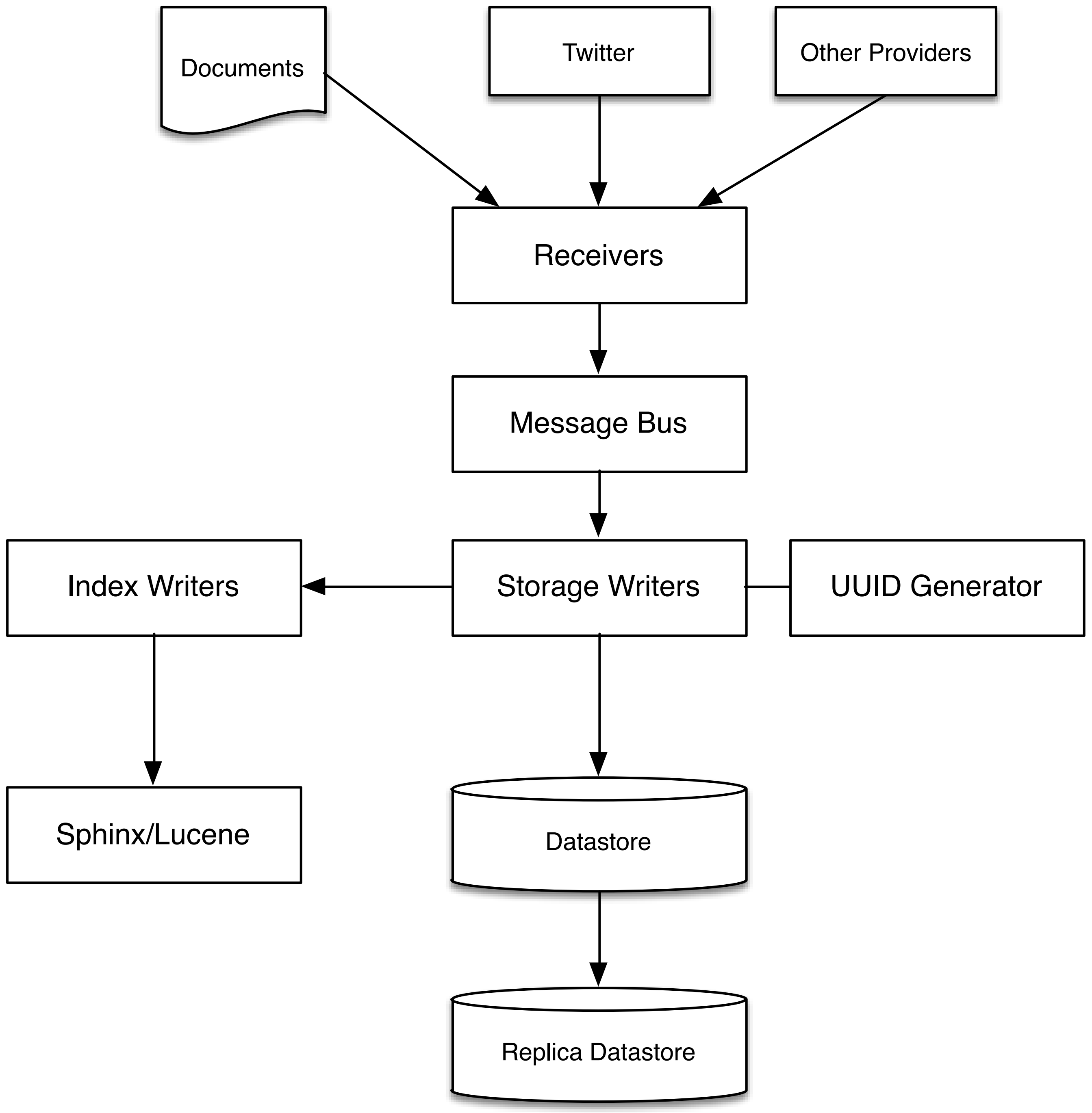
```
—
 - hosts: all
   tasks:
   - name: ensure ring size is 128
     lineinfile: dest=/etc/riak.conf
     line='ring_size = 128'
     regexp=‘ring_size[^=]*=.*'
     owner=root
     state=present create=False
```

Reduced Replicas for Multiple Data Centers

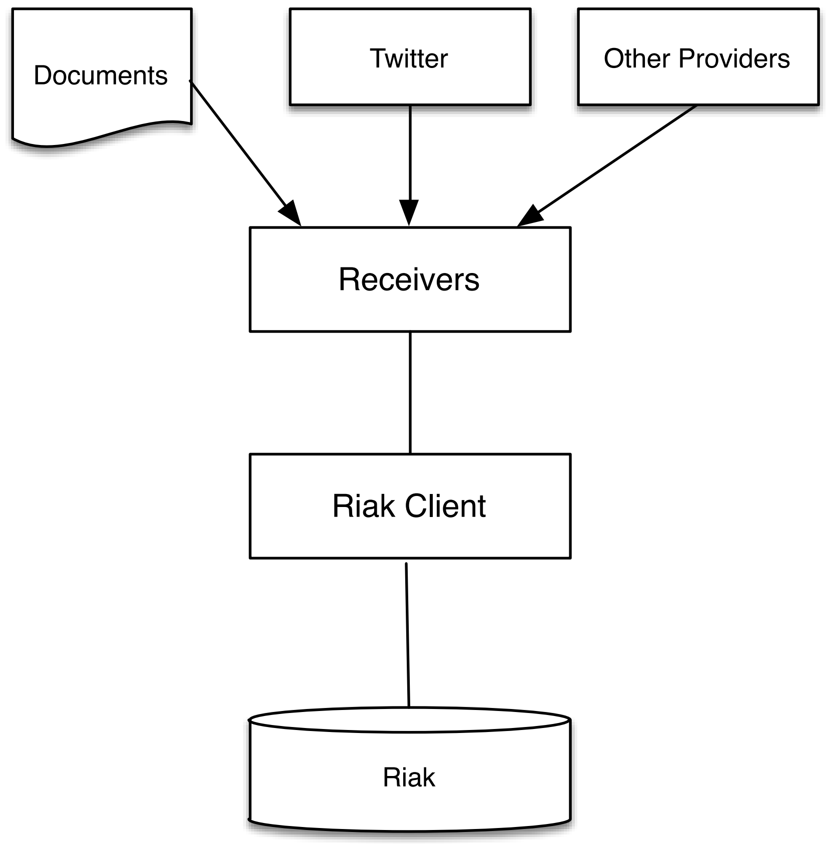
Strong Consistency

Example uses

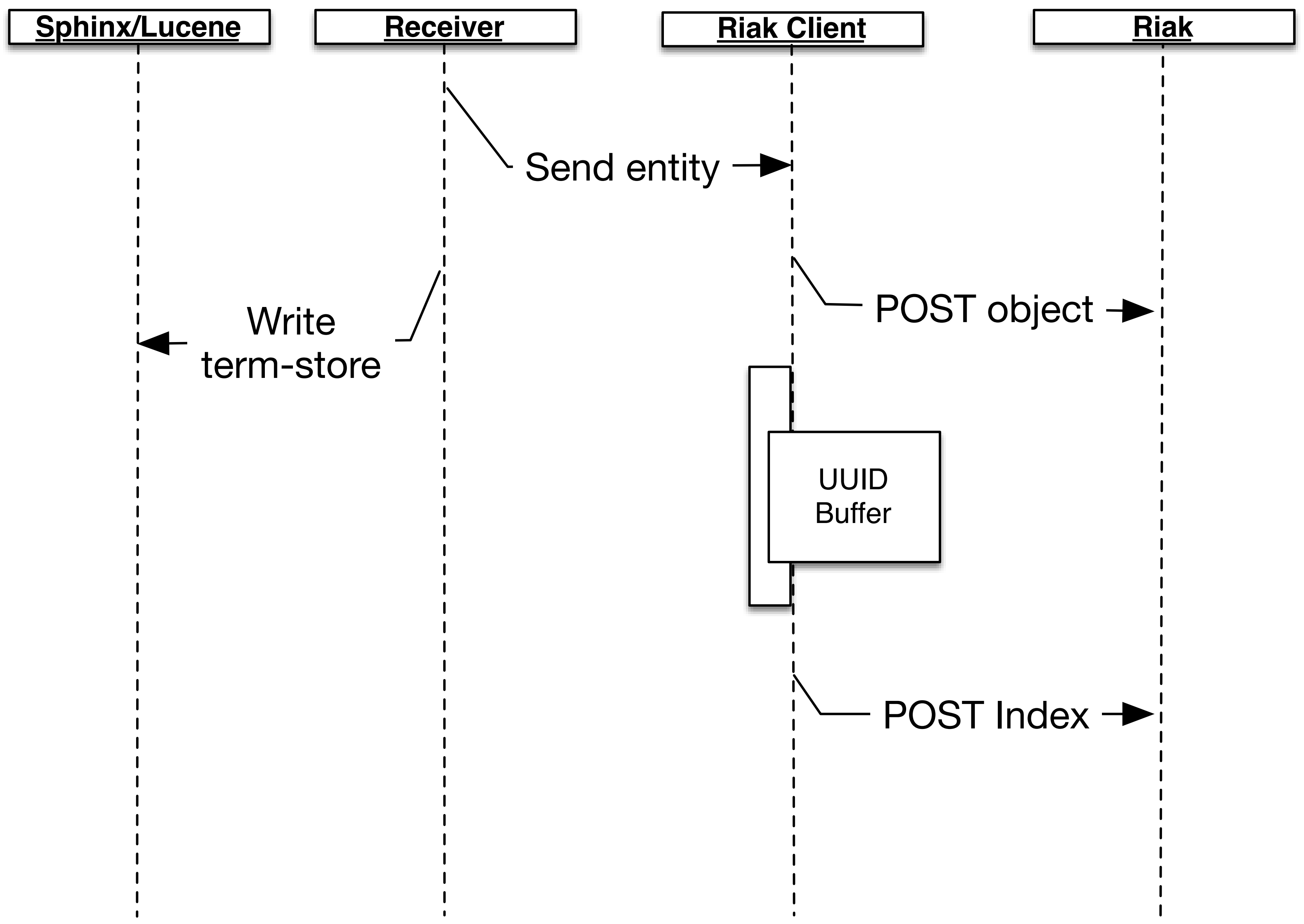
Social Media (old)



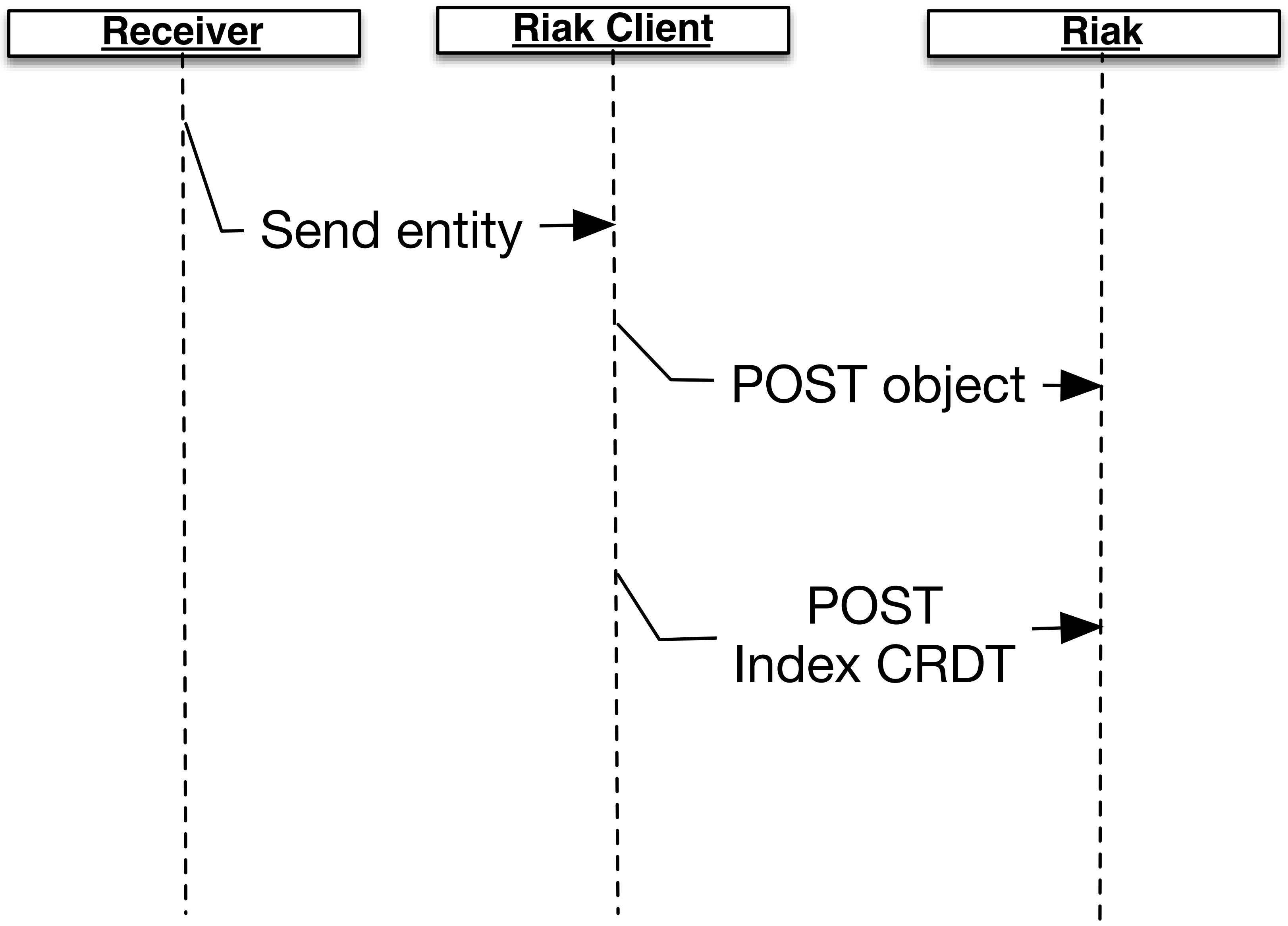
Social Media (new)



Social Media (old)



Social Media (new)



The End

- Questions ?