Cassandra & Akka

IndyScala, August 5, 2013

What is Cassandra?

A persistent Map[SortedMap[Any, Any]] that is:

- distributed
- fault tolerant
- tunably consistent
- FAST!

Let’s build a system that...

1. Ingests the Twitter Sample API*
2. Detects the language of the tweets
3. Create a searchable index, by language
4. Records info about users:
    a. Their interests (#hashtags)
    b. People who talk to them (@mentions)
    c. The languages they tweet in

* - I’m not worthy of the firehose

Our schema

tweets	
tweet_id (PK)	bigint
text	text
username	text
lang	map<text, text>

tweet_terms	
term (PK)	text
lang (PK)	text
tweet_id (PK)	bigint
username	text
text	text

user_profiles	
username	text
languages	set<text>
hashtags	set<text>
mentions	set<text>

counters	
name	count

How do you define it?

CQL3 DDL looks a lot like SQL DDL:

```
create table tweets (
  tweet_id bigint primary key,
  text text,
  username text,
  lang map<text, text>
);
```

How do you query it?

CQL3 DML looks a lot like SQL DML:

```
INSERT INTO tweets (tweet_id, text,
username) VALUES (?, ?, ?);

SELECT * FROM tweets WHERE tweet_id = ?;
```

But it’s not relational

- No check constraints on columns
- No foreign keys to other tables
- Not normalized.
- Not even in 1NF!!!

So you can’t do...

- joins
- subqueries
- like queries
- most forms of aggregation
- filtering by non-PK columns*
- range queries on PK**
- ordering by column value***

* - without a secondary index

** - you can, but it’s inefficient

*** - unless the second column of the PK

That sucks!

- Yeah, at first, if you're a relational guy, it does suck.
- Think in terms of the data structure.
  - Review: Map[SortedMap[Any, Any]]
- Think in terms of how you'll query that data structure.
- Don't fear denormalization.
- You don't have to give up your other DBMS.

Seems limiting...

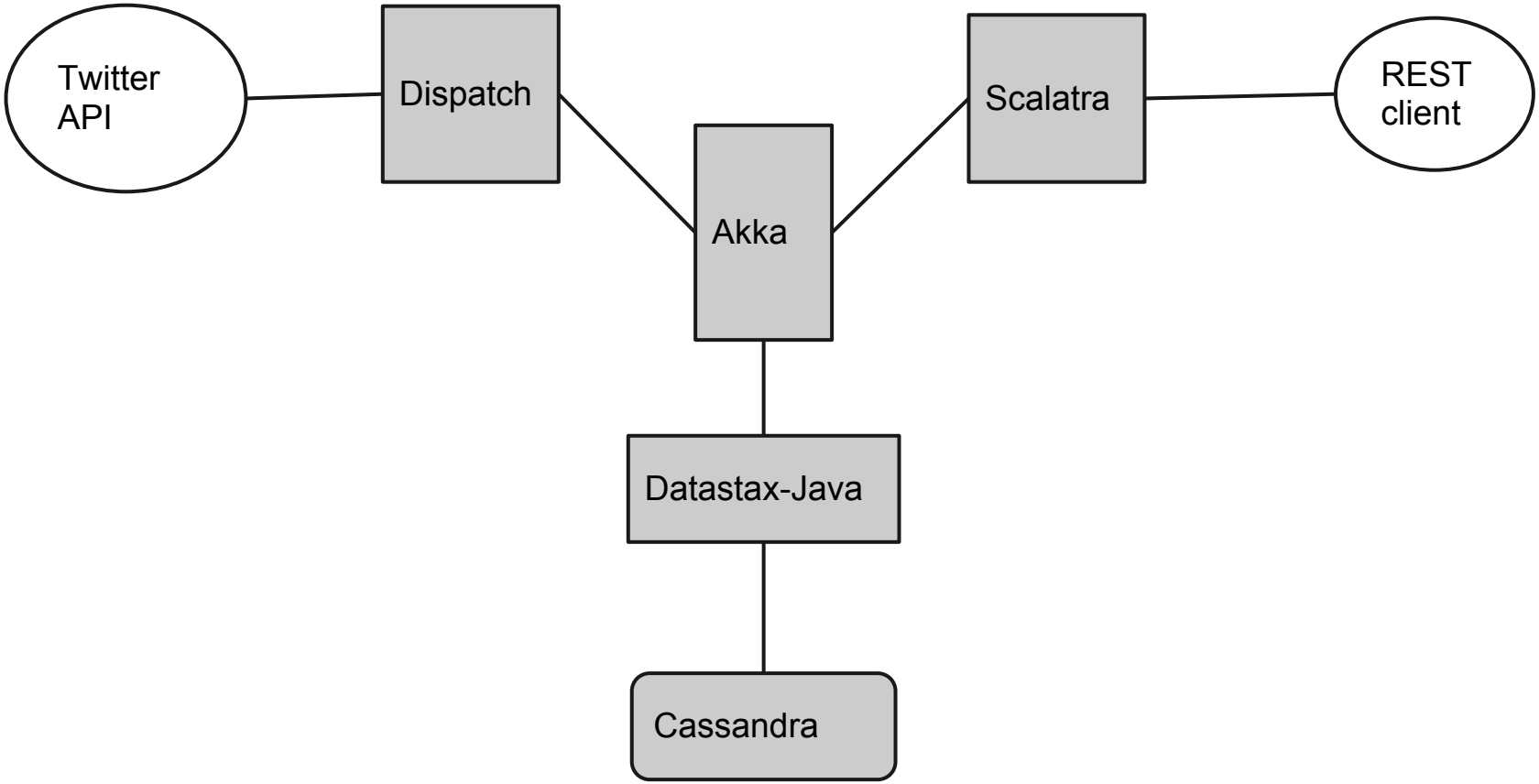
But it’s also freeing.

- Leave the sharding to it.
- Leave the replication to it.
- Add nodes on the fly.
- Available if some nodes fail. - *
- Available if Godzilla rampages through one of your data centers. - **

* - how many? depends how you tuned it.

** - as long as he doesn’t get them all

Architecture



Walk through the code

Open your browsers and follow along as we discuss the important classes.

https://github.com/indyscala/cassandra-demo/tree/master/src/main/scala/indyscala/aug2013

source.TwitterSample

- Uses Dispatch to connect to the Twitter Sample API.
- Parses each line of the streaming response with json4s.
- Executes a callback function. In our case, we send it to the Twitter Service.

service.TwitterService

- Our central nervous system.
- Supervises our subservices:
  - tweet parser actor
  - data store actor
  - language detector actor
  - indexer actor
- Knows how to route:
  - incoming JSON messages from the TwitterSample
  - parsed tweets from the
  - incoming queries from Scalatra
  - results from the language detection actor
  - results from the index actor

service.JsonParser

- Knows how to construct a tweet from a JSON value.
- Actually, the smarts are built into extractors in the model package.
- Important point: feeds recognized Tweets back into the TwitterService.

service.LangDetector

- Detects languages from two sources
  - Twitter gives us its best guess
  - Analyzes with Cybozu Labs' LangDetect, an open source project.
- Does the detections in parallel, and reports back to the TwitterService to store the results and trigger indexing.
- Adding an HTTP services for a third opinion via Dispatch would be easy.
  - 10% of my speaker fee of $0 bought us 0 such services.

service.Indexer

- Triggered when:
  - a tweet is parsed (for any-language searches)
  - a language is detected
- Yes, yes, caching parsed tweets may have been more efficient. But this approach was simple.
- Yes, yes, the index table is repetitious. But it’s efficient in Cassandra. Forget your relational instincts!

service.DataStore

- Finally... this actor is why we're all gathered here tonight.
- All Cassandra interaction happens in this actor.
  - That's not necessary. I have lots of actors that talk to Cassandra at work. This is a simple app.
- It's all non-blocking. Great fit for Akka.
- DataStax's Java driver is nice, but you deal with some types you left behind when you left Java behind...

util.DataStaxImplicits

- Implicits to the rescue!
- Adapt Guava's Listenable Future to scala.concurrent.Future, makes working with Cassandra feel like a first-class citizen in Akka.
- Map ResultSets over an arbitrary function, to feel more like a Scala collection.
  - Map straight to json with json4s, because that's what we usually do.

util.DataStaxExtension

- Provides a way for multiple actors to share a Cassandra connection pool.
  - Only one actor type talks to Cassandra in this demo, but we have more than one instance of it.
- It’s convenient like a global, but it’s scoped to the ActorSystem.
  - We only have one ActorSystem, but you’ll really appreciate this when you start writing tests.

servlet.TwitterServlet

- A simple Scalatra service to define a few REST endpoints and query the service.
- Sends query messages to the TwitterService, and renders the returned JSON.
- Yes, Virginia, there is asynchronous support in the Servlet API.
  - Are you the same person you were in 1997? Neither are Java Servlets.

What still hurts

- Typing
  - Known invalid CQL3 types can be bound.
  - Collections are Java collections.
- Positional parameters

Anorm did a nice job applying a Scala sheen to JDBC. Something similar for talking to Cassandra would be a big win for Scala developers.

Immaturity

- Don’t panic. I’ve not lost any data yet.
  - But you keep running into things that aren’t implemented yet. Like prepared statements with a map key, so I had to drop into String concat.
- Some of the immaturity is in me.
  - A dozen years of RDBMS wisdom was hard earned. So shall it be with Cassandra.
- Some of the immaturity is in us all.
  - Nobody has a dozen years of Cassandra wisdom, because it’s not a dozen years old. We’re all learning on the fly here.

In conclusion

- I work on Big Data. Cassandra is doing things for us that PostgreSQL simply can't.
- But sometimes I need ad hoc queries on smaller data sets, and PostgreSQL is still really great at that.
- It's okay to use them both. NoSQL is not NoSQLAnywhere.
  - And with Akka and future comprehensions, you can join them together at the application level where necessary.

Thanks and good night

- http://github.com/indyscala/cassandra-demo
- I’m Ross A. Baker (@rossabaker), and I work on problems like this at CrowdStrike.
- Thanks to E-gineering for hosting.
- Thanks to Brad Fritz (@bfritz) of Fewer Hassles for organizing.