apigee

Apache Cassandra

Clients and Transports

Hi Folks!
I'm Nate
@zznate

apigee

API Management
API Analytics
API Tools

Clients and transports for Cassandra

But first...
some questions

But first:
Architectural stuff

Cassandra: “Sparsely Columnar”

An RDBMS table

ID	Name	Color	Age
9ae30...54ced2	Fluffy	Orange Tabby	7
9a4f8...54ced2	Darth	Black	NULL
9bd05...54ced2	Mr. Tibbs	Calico	3
954d7..54ced2	Scooter	Grey	NULL

An RDBMS table

ID	Name	Color	Age
9ae30...54ced2	Fluffy	Orange Tabby	7
9a4f8...54ced2	Darth	Black	NULL
9bd05...54ced2	Mr. Tibbs	Calico	3
954d7..54ced2	Scooter	Grey	NULL

Cassandra Style

ID	Name	Color	Age
9ae30...54ced2	Fluffy	Orange Tabby	7
9a4f8...54ced2	Darth	Black	
9bd05...54ced2	Mr. Tibbs	Calico	3
954d7..54ced2	Scooter	Grey	

Cassandra Style

ID	Name	Color	Age
9ae30...54ced2	Fluffy	Orange Tabby	7
9a4f8...54ced2	Darth	Black	
9bd05...54ced2	Mr. Tibbs	Calico	3
954d7..54ced2	Scooter	Grey	

ID	Name	Color	Age	Temperment
954d7..54ced2	Scooter	Grey		Cuddly

Cassandra data modelling

Four common patterns

Simple object to simple row

Sparse object to rows

Materialized view

Manual index

simple objects to simple row

“static” column family

ID	Name	Color	Age
9ae30...54ced2	Fluffy	Orange Tabby	7
9a4f8...54ced2	Darth	Black	
9bd05...54ced2	Mr. Tibbs	Calico	3
954d7..54ced2	Scooter	Grey	

Sparse Objects

“dynamic” column family

Pets

ID	Name	Color	Age	Temperment	Water temp	Type
954d7..54ced2	Scooter	Grey		Cuddly		Cat
...	...	...	...	...	...	...
962e1..54ced2		Gold	8		64	Fish

Materialized Views

Materialized view

KEY: **2013_02_27_10_03_05**

954d7..54ced2	127.0.0.1 - - [10/Oct/2012:12:16:39 -0700] /foo/bar GET HTTP/1.1" 200 1586
962e1..54ced2	127.0.0.1 - - [10/Oct/2012:12:16:39 -0700] /foo/bar GET HTTP/1.1" 200 2048
9bd01..54ced2	127.0.0.1 - - [10/Oct/2012:12:16:39 -0700] /foo/bar GET HTTP/1.1" 200 1022
9ed02..54ce2d	127.0.0.1 - - [10/Oct/2012:12:16:39 -0700] /foo/bar GET HTTP/1.1" 200 4506

KEY: **2013_02_27_10_03_10**

…

KEY: **2013_02_27_10_03_15**

…

Regardless of the approached used, there are four overall goals

1. Denormalize
2. Eliminate seeks
3. Design for read
4. Optimiza for blind writes

Now... let's talk about protocols

Thrift

Thrift
RPC-Based

Thrift
RPC-Based
Mature Apache Project

Thrift
RPC-Based
Mature Apache Project
Supports lots of languages

Thrift
RPC-Based
Mature Apache Project
Supports lots of languages
Extensible!

CQL

CQL
Well defined protocol

CQL
Well defined protocol
Supports Compression

CQL
Well defined protocol
Supports Compression
Netty/NIO-based

Storage Mechanics (but quickly)

get_slice

```
1   list<ColumnOrSuperColumn> get_slice(1:required binary key,
2                                       2:required ColumnParent column_parent,
3                                       3:required SlicePredicate predicate,
4                                       4:required ConsistencyLevel consistency_level=ConsistencyLevel.ONE)
5                             throws (1:InvalidRequestException ire, 2:UnavailableException ue,
6                                     3:TimedOutException te),
```

Workhorse of Cassandra selection methods

get_slice: key

```
1   list<ColumnOrSuperColumn> get_slic (1:required binary key,
2                                       2:required ColumnParent column_parent,
3                                       3:required SlicePredicate predicate,
4                                       4:required ConsistencyLevel consistency_level=ConsistencyLevel.ONE)
5                             throws (1:InvalidRequestException ire, 2:UnavailableException ue,
6                                     3:TimedOutException te),
```

The row key

get_slice: ColumnParent

```
1   list<ColumnOrSuperColumn> get_slice(1:required binary key,
2                                       2:required ColumnParent column_parent,
3                                       3:required SlicePredicate predicate,
4                                       4:required ConsistencyLevel consistency_level=ConsistencyLevel.ONE)
5                           throws (1:InvalidRequestException ire, 2:UnavailableException ue,
6                                   3:TimedOutException te),
```

The column family (a.k.a table)

get_slice: SlicePredicate

```
1  list<ColumnOrSuperColumn> get_slice(1:required binary key,
2                                      2:required ColumnParent column_parent,
3                                      3:required SlicePredicate predicate,
4                                      4:required ConsistencyLevel consistency_level=ConsistencyLevel.ONE)
5                              throws (1:InvalidRequestException ire, 2:UnavailableException ue,
6                                      3:TimedOutException te),
```

defines the column range, or specifically named columns

get_slice:ConsistencyLevel

```
1  list<ColumnOrSuperColumn> get_slice(1:required binary key,
2                                      2:required ColumnParent column_parent,
3                                      3:required SlicePredicate predicate,
4                                      4:required ConsistencyLevel consistency_level=ConsistencyLevel.ONE)
5                              throws (1:InvalidRequestException ire, 2:UnavailableException ue,
6                                      3:TimedOutException te),
```

The level of consistency we want for this read

Obtuse at first glance, but nothing is hidden

So...

But one person's abstraction leakage is another's preffered model

How closely do you want to interact with the underlying storage engine?

Client APIs

Benefits of thrift

Benefits of thrift

Mature selection of clients

Benefits of thrift

Mature selection of clients

Multiple languages

Benefits of thrift

Mature selection of clients

Multiple languages

Well documented

Benefits of thrift

Mature selection of clients

Multiple languages

Well documented

Can be used in other places

Drawbacks of thrift

Drawbacks of thrift

Several objects are required for any request

Drawbacks of thrift

Several objects are required for any request

Clients differs in implementation

Drawbacks of thrift

Several objects are required for any request

Clients differs in implementation

Upstream dependency issues

Drawbacks of thrift

Several objects are required for any request

Clients differs in implementation

Upstream dependency issues

Schema changes and cluster health done pro-actively

Benefits of cql api

Benefits of cql api

Stored procedures

Benefits of cql api

Stored procedures

Common operations are straight forward

Benefits of cql api

Stored procedures

Common operations are straight forward

Cluster health and schema change push-back

Benefits of cql api

Stored procedures

Common operations are straight forward

Cluster health and schema change push-back

Awesome client available

Drawbacks of CQL apis

Drawbacks of CQL apis

Still have idiomatic clients

Drawbacks of CQL apis

Still have idiomatic clients

Still a binary protocol

Drawbacks of CQL apis

Still have idiomatic clients

Still a binary protocol

Default storage model emposes substantial restrictions

** see gotchas section later

Considerations for your app

Stick with Thrift if...

Heavy update workloads

Large, dynamic batch insertions

Hadoop integration (CASSANDRA-4421)

Commonly deal with very wide rows (CASSANDRA-4176)

CASSANDRA-4176: “Pick your shard keys carefully”

Sharded Cluster Internals and Behaviors

This document introduces lower level sharding concepts for users who are familiar with *shardi*
about the internals. This document provides a more detailed understanding of your cluster's be
concepts, see *Sharded Cluster Overview*. For complete documentation of sharded clusters see

Shard Keys

Shard keys are the field in a collection that MongoDB uses to distribute *documents* within a sh
keys for an introduction to these topics.

Cardinality

Cardinality in the context of MongoDB, refers to the ability of the system to *partition* data ir
collection of data such as an "address book" that stores address records:

- Consider the use of a **`state`** field as a shard key:

  The state key's value holds the US state for a given address document. This field ha

Consider CQL if...

Static column family model: Take advantage of stored procedures for common reads

Despite the shard key jab, CQL makes good use of the storage model

You can replace some custom serialization with collections

Integration with JDBC and/or BI tools

Wire efficient: Does not return timestamp or TTL by default

Larger, potentially more transient evironments

But CQL is
- new
- an abstraction

In some cases, CQL might not do what you think

Most common CQL pitfalls

Collections can only be retrieved in their entirety

Can’t mix static and dynamic data in a column family

“keys only” range slices don’t work (CASSANDRA-4536)

Range ghosts will not be returned

Batch inserts are clunky (CASSANDRA-4693)

With non-compact storage the whole row must be read every time.

The take away is that you have options. Particularly good ones for Java.

```
# Whether to start the native transport server.
# Currently, only the thrift server is started by default because the native
# transport is considered beta.
# Please note that the address on which the native transport is bound is the
# same as the rpc address. The port however is different and specified below.
start_native_transport: true
# port for the CQL native transport to listen for clients on
native_transport_port: 9042
# The maximum of thread handling requests. The meaning is the same than
# rpc_max_threads. The default is unlimited.
#native_transport_max_threads: 2048

# Whether to start the thrift rpc server.
start_rpc: true
# The address to bind the Thrift RPC service to -- clients connect
# here. Unlike ListenAddress above, you *can* specify 0.0.0.0 here if
# you want Thrift to listen on all interfaces.
#
# Leaving this blank has the same effect it does for ListenAddress,
# (i.e. it will be based on the configured hostname of the node).
rpc_address: localhost
# port for Thrift to listen for clients on
rpc_port: 9160

# enable or disable keepalive on rpc connections
```

BUT

there is a larger, more fundamental problem to discuss

“If [they] think that CQL is the answer to usability then I just won. We at least know where our problems are.”

- 10gen exec.

The market has spoken and we missed the boat.

POST /endpoint {json}

A Cassandra-MVP actually maintains a REST front-end

So we've taken this and gone further

What if...

Coming soon...
Intravert.
Vert.x+Cassandra.
ASF-licensed.
Driven by real-world requirements.