Avoiding anti-patterns: Staying in love with Cassandra

Christopher Batey
Technical Evangelist for Apache Cassandra
@chbatey

Who am I?

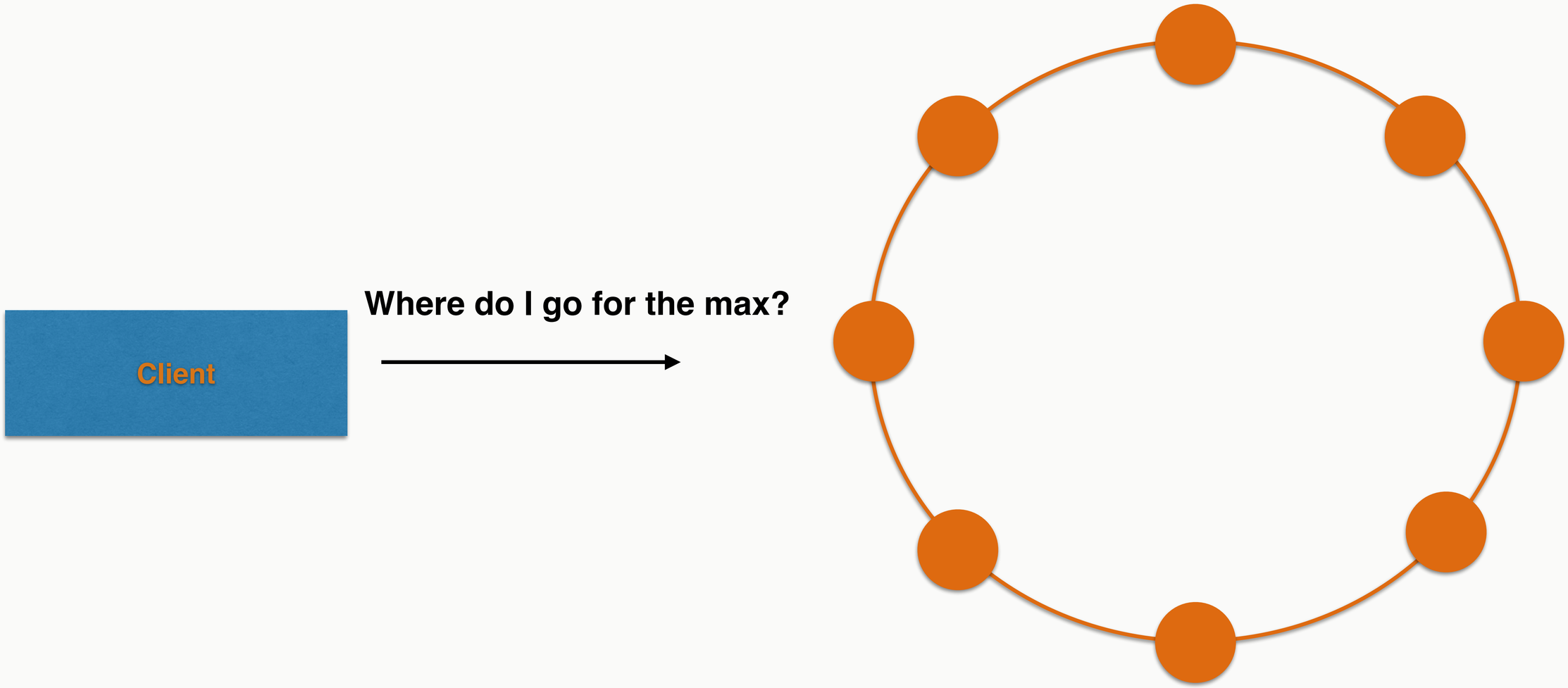
- Technical Evangelist for Apache Cassandra
  - Work on Stubbed Cassandra
  - Help out Apache Cassandra users
- Built systems using Java/Spring/Dropwizard with Cassandra @ Sky
- Follow me on twitter @chbatey

Anti patterns

- Client side joins
- Multi partition queries
- Batches
- Mutable data
- Retry policies
- Tombstones
- Secondary indices
  - Includes home work + prize

Distributed joins

Cassandra can not join or aggregate



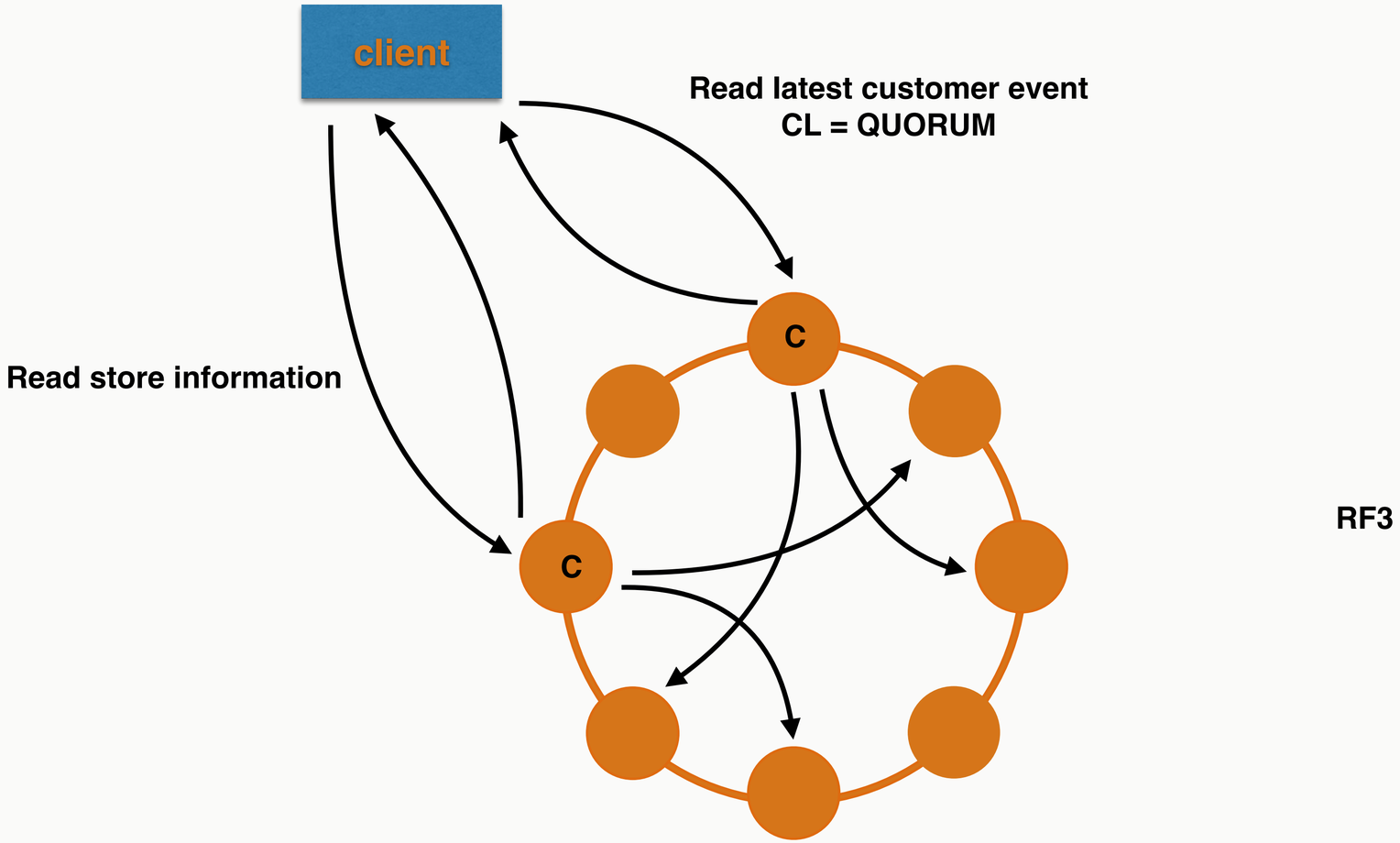
Storing customer events

- Customer event
  - customer_id - ChrisBatey
  - staff_id - Charlie
  - event_type - login, logout, add_to_basket, remove_from_basket
  - time
- Store
  - name
  - store_type - Website, PhoneApp, Phone, Retail
  - location

Model

```
CREATE TABLE customer_events(
     customer_id text,
     staff_id text,
     time timeuuid,
     event_type text,
     store_name text,
     PRIMARY KEY ((customer_id), time));

CREATE TABLE store(
     store_name text,
     location text,
     store_type text,
     PRIMARY KEY (store_name));
```

Reading this



One to one vs one to many

- This required possibly 6 out of 8 of our nodes to be up
- Imagine if there were multiple follow up queries

Multi-partition queries

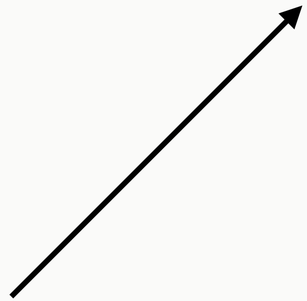
- Client side joins normally end up falling into the more general anti-pattern of multi partition queries

Adding staff link

```
CREATE TABLE customer_events(
      customer_id text,
      staff set<text>,
      time timeuuid,
      event_type text,
      store_name text,
      PRIMARY KEY ((customer_id), time));

CREATE TYPE staff(
      name text,
      favourite_colour text,
      job_title text);

CREATE TYPE store(
      store_name text,
      location text,
      store_type text);
```

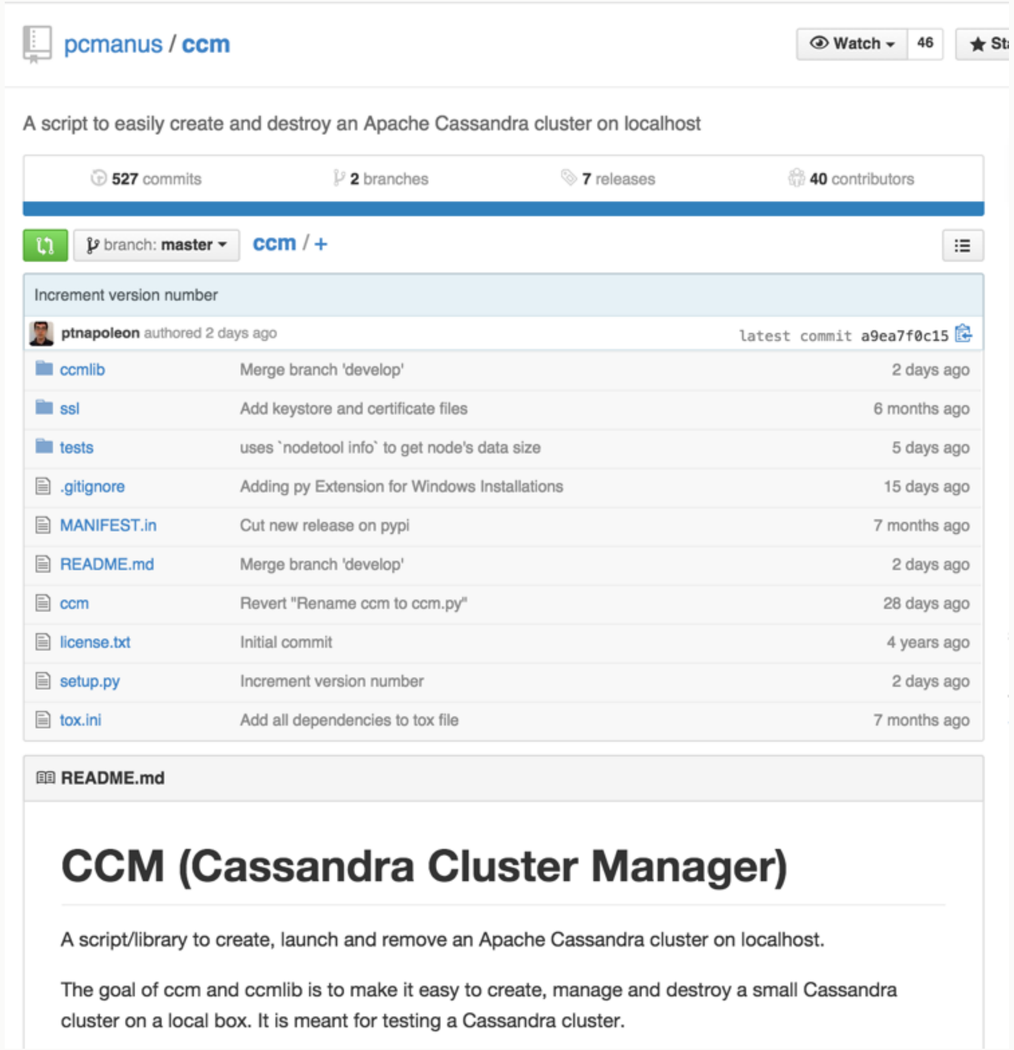
Queries

```
select * from customer_events where customer_id = ‘chbatey’
limit 1

select * from staff where name in (staff1, staff2, staff3,
staff4)
```



Any one of these fails the whole query fails

Use a multi node cluster locally



Use a multi node cluster locally

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns    Host ID
Rack
UN  127.0.0.1  102.27 KB  256     ?       15ad7694-3e76-4b74-aea0-fa3c0fa59532
rack1
UN  127.0.0.2  102.18 KB  256     ?       cca7d0bb-e884-49f9-b098-e38fbe895cbc
rack1
UN  127.0.0.3  93.16 KB   256     ?       1f9737d3-c1b8-4df1-be4c-d3b1cced8e30
rack1
UN  127.0.0.4  102.1 KB   256     ?       fe27b958-5d3a-4f78-9880-76cb7c9bead1
rack1
UN  127.0.0.5  93.18 KB   256     ?       66eb3f23-8889-44d6-a9e7-ecdd57ed61d0
rack1
UN  127.0.0.6  102.12 KB  256     ?       e2e99a7b-c1fb-4f2a-9e4f-7a4666f8245e
rack1
```

Let’s see with with a 6 node cluster

```
INSERT INTO staff (name, favourite_colour , job_title ) VALUES ( 'chbatey', 'red',
'Technical Evangelist' );
INSERT INTO staff (name, favourite_colour , job_title ) VALUES ( 'luket', 'red',
'Technical Evangelist' );
INSERT INTO staff (name, favourite_colour , job_title ) VALUES ( 'jonh', 'blue',
'Technical Evangelist' );

select * from staff where name in ('chbatey', 'luket', 'jonh');
```

Trace with CL ONE = 4 nodes used

```
Execute CQL3 query | 2015-02-02 06:39:58.759000 | 127.0.0.1 |              0
Parsing select * from staff where name in ('chbatey', 'luket', 'jonh'); [SharedPool-Worker-1] | 2015-02-02
06:39:58.766000 | 127.0.0.1 |           7553
Preparing statement [SharedPool-Worker-1] | 2015-02-02 06:39:58.768000 | 127.0.0.1 |           9249
Executing single-partition query on staff [SharedPool-Worker-3] | 2015-02-02 06:39:58.773000 | 127.0.0.1 |          14255
Sending message to /127.0.0.3 [WRITE-/127.0.0.3] | 2015-02-02 06:39:58.773001                   | 127.0.0.1 |          14756
Sending message to /127.0.0.5 [WRITE-/127.0.0.5] | 2015-02-02 06:39:58.773001                   | 127.0.0.1 |          14928
Sending message to /127.0.0.3 [WRITE-/127.0.0.3] | 2015-02-02 06:39:58.774000                   | 127.0.0.1 |          16035
Executing single-partition query on staff [SharedPool-Worker-1] | 2015-02-02 06:39:58.777000 | 127.0.0.5 |           1156
Enqueuing response to /127.0.0.1 [SharedPool-Worker-1] | 2015-02-02 06:39:58.777001           | 127.0.0.5 |           1681
Sending message to /127.0.0.1 [WRITE-/127.0.0.1] | 2015-02-02 06:39:58.778000                   | 127.0.0.5 |           1944
Executing single-partition query on staff [SharedPool-Worker-1] | 2015-02-02 06:39:58.778000 | 127.0.0.3 |           1554
Processing response from /127.0.0.5 [SharedPool-Worker-3] | 2015-02-02 06:39:58.779000         | 127.0.0.1 |          20762
Enqueuing response to /127.0.0.1 [SharedPool-Worker-1] | 2015-02-02 06:39:58.779000           | 127.0.0.3 |           2425
Sending message to /127.0.0.5 [WRITE-/127.0.0.5] | 2015-02-02 06:39:58.779000                   | 127.0.0.1 |          21198
Sending message to /127.0.0.1 [WRITE-/127.0.0.1] | 2015-02-02 06:39:58.779000                   | 127.0.0.3 |           2639
Sending message to /127.0.0.6 [WRITE-/127.0.0.6] | 2015-02-02 06:39:58.779000                   | 127.0.0.1 |          21208
Executing single-partition query on staff [SharedPool-Worker-1] | 2015-02-02 06:39:58.780000 | 127.0.0.5 |            304
Enqueuing response to /127.0.0.1 [SharedPool-Worker-1] | 2015-02-02 06:39:58.780001           | 127.0.0.5 |            574
Executing single-partition query on staff [SharedPool-Worker-2] | 2015-02-02 06:39:58.781000 | 127.0.0.3 |           4075
Sending message to /127.0.0.1 [WRITE-/127.0.0.1] | 2015-02-02 06:39:58.781000                   | 127.0.0.5 |            708
Enqueuing response to /127.0.0.1 [SharedPool-Worker-2] | 2015-02-02 06:39:58.781001           | 127.0.0.3 |           4348
Sending message to /127.0.0.1 [WRITE-/127.0.0.1] | 2015-02-02 06:39:58.782000                   | 127.0.0.3 |           5371
Executing single-partition query on staff [SharedPool-Worker-1] | 2015-02-02 06:39:58.783000 | 127.0.0.6 |           2463
Enqueuing response to /127.0.0.1 [SharedPool-Worker-1] | 2015-02-02 06:39:58.784000           | 127.0.0.6 |           2905
Sending message to /127.0.0.1 [WRITE-/127.0.0.1] | 2015-02-02 06:39:58.784001                   | 127.0.0.6 |           3160
Processing response from /127.0.0.6 [SharedPool-Worker-2] | 2015-02-02 06:39:58.785000         | 127.0.0.1 |             --
Request complete | 2015-02-02 06:39:58.782995                                                     | 127.0.0.1 |          23995
```

Denormalise with UDTs

```
CREATE TYPE store (name text, type text, postcode text)

CREATE TYPE staff (name text, fav_colour text, job_title text)


CREATE TABLE customer_events(
     customer_id text,
     time timeuuid,
     event_type text,
     store store,
     staff set<staff>,
     PRIMARY KEY ((customer_id), time));
```

Less obvious example

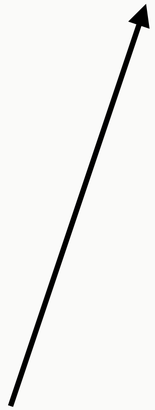
- A good pattern for time series: **Bucketing**

Adding a time bucket

```
CREATE TABLE customer_events_bucket(
    customer_id text,
    time_bucket text
    time timeuuid,
    event_type text,
    store store,
    staff set<staff>,
    PRIMARY KEY ((customer_id, time_bucket), time));
```

Queries

```
select * from customer_events_bucket where customer_id =
‘chbatey’ and time_bucket IN (‘2015-01-01|0910:01’,
‘2015-01-01|0910:02’, ‘2015-01-01|0910:03’, ‘2015-01-01|
0910:04’)
```



Often better as multiple async queries

Tips for avoiding joins & multi gets

- Say no to client side joins by denormalising
  - Much easier with UDTs
- When bucketing aim for at most two buckets for a query
- Get in the habit of reading trace + using a multi node cluster locally

Unlogged Batches

Unlogged Batches

- Unlogged batches
  - Send many statements as one

Customer events table

```
CREATE TABLE if NOT EXISTS customer_events (
    customer_id text,
    statff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (customer_id, time))
```

Inserting events

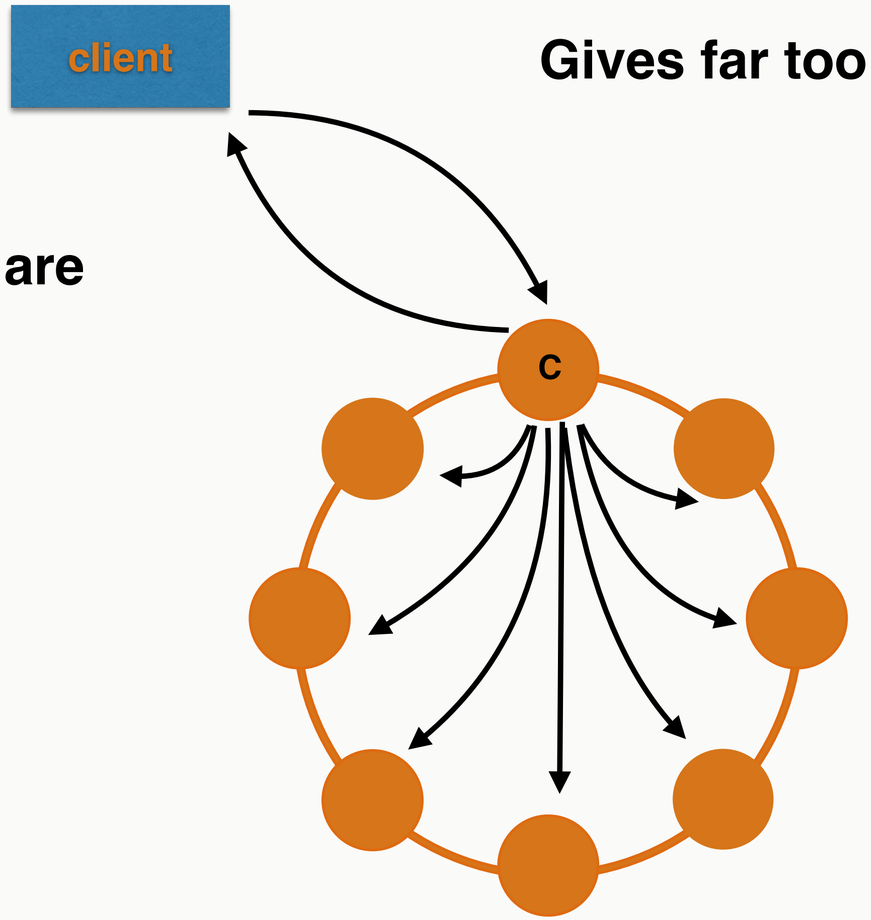
```
INSERT INTO events.customer_events (customer_id, time , event_type , statff_id , store_type ) VALUES
( ?, ?, ?, ?, ?)
```

```
public void storeEvent(ConsistencyLevel consistencyLevel, CustomerEvent customerEvent) {
    BoundStatement boundInsert = insertStatement.bind(
         customerEvent.getCustomerId(),
         customerEvent.getTime(),
         customerEvent.getEventType(),
         customerEvent.getStaffId(),
         customerEvent.getStaffId());


    boundInsert.setConsistencyLevel(consistencyLevel);
    session.execute(boundInsert);
}
```

Batch insert - Good idea?

```
public void storeEvents(ConsistencyLevel consistencyLevel, CustomerEvent... events) {
    BatchStatement batchStatement = new BatchStatement(BatchStatement.Type.UNLOGGED);

    for (CustomerEvent event : events) {
         BoundStatement boundInsert = insertStatement.bind(
         customerEvent.getCustomerId(),
         customerEvent.getTime(),
         customerEvent.getEventType(),
         customerEvent.getStaffId(),
         customerEvent.getStaffId());

         batchStatement.add(boundInsert);
    }
    session.execute(batchStatement);
}
```

Not so much

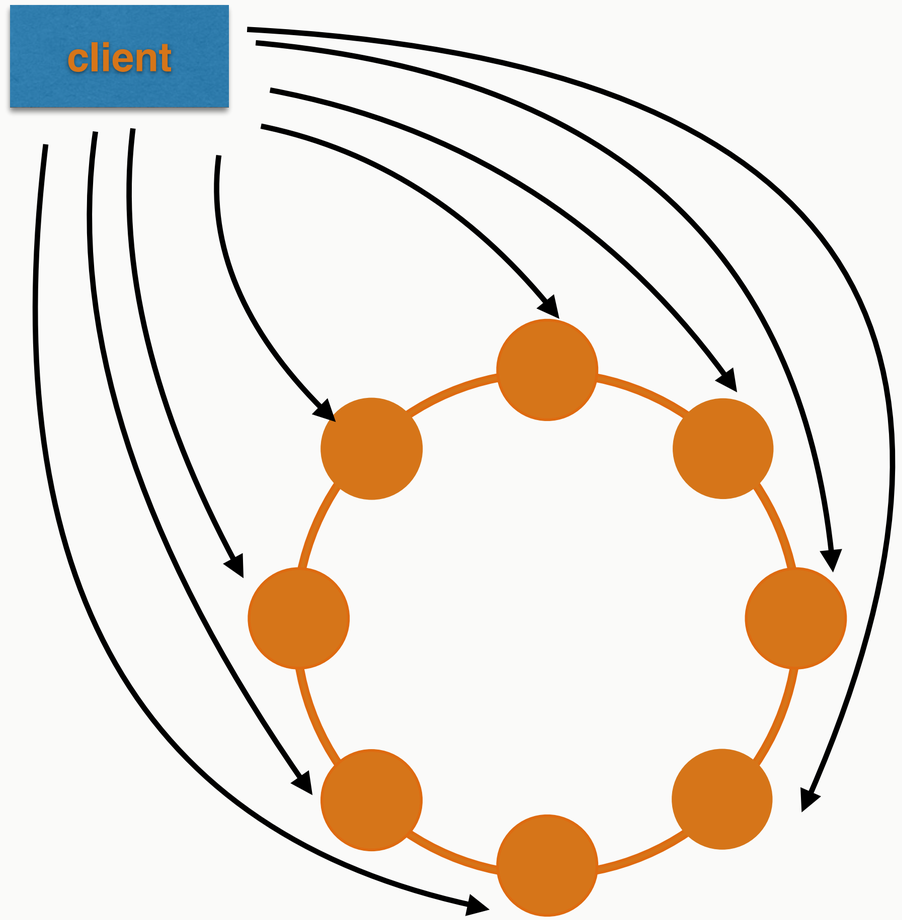


Gives far too much work for the coordinator

Very likely to fail if nodes are down / over loaded

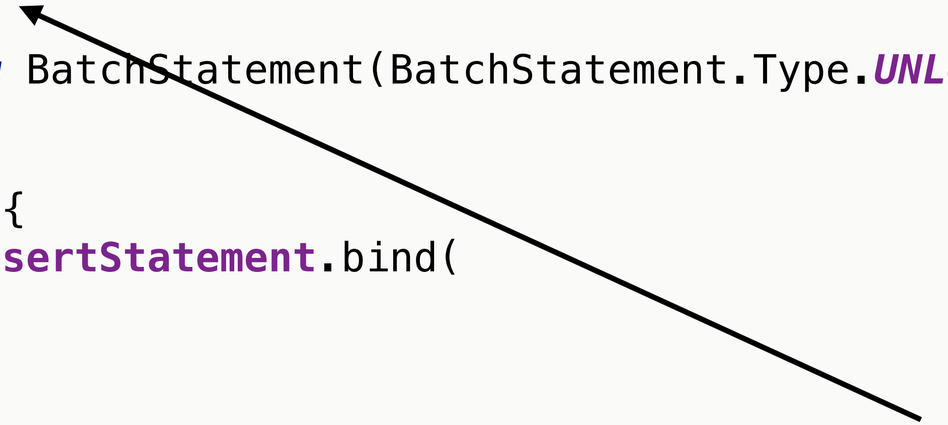
Ruins performance gains of token aware driver

Individual queries

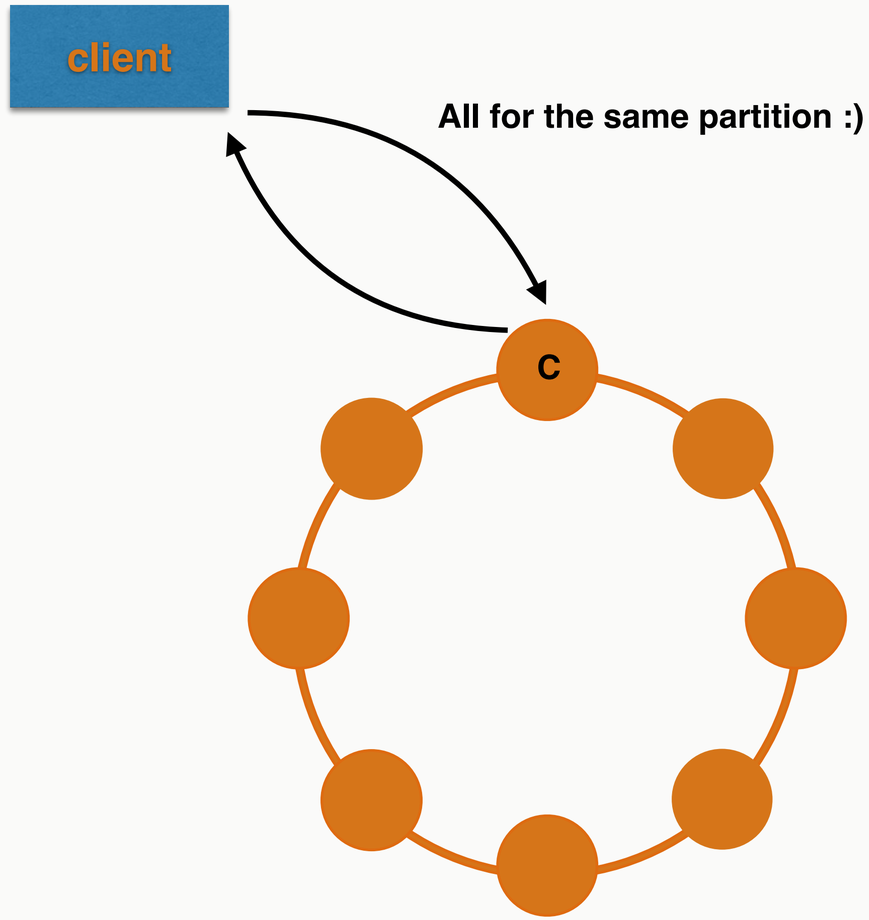


Unlogged batch use case

```
public void storeEvents(String customerId, ConsistencyLevel consistencyLevel, CustomerEvent...
events) {
    BatchStatement batchStatement = new BatchStatement(BatchStatement.Type.UNLOGGED);
    batchStatement.enableTracing();

    for (CustomerEvent event : events) {
        BoundStatement boundInsert = insertStatement.bind(
                customerId,
                event.getTime(),
                event.getEventType(),
                event.getStaffId(),
                event.getStaffId());
        boundInsert.enableTracing();
        boundInsert.setConsistencyLevel(consistencyLevel);
        batchStatement.add(boundInsert);
    }

    ResultSet execute = session.execute(batchStatement);
    logTraceInfo(execute.getExecutionInfo());
}
```



Writing this



Logged Batches

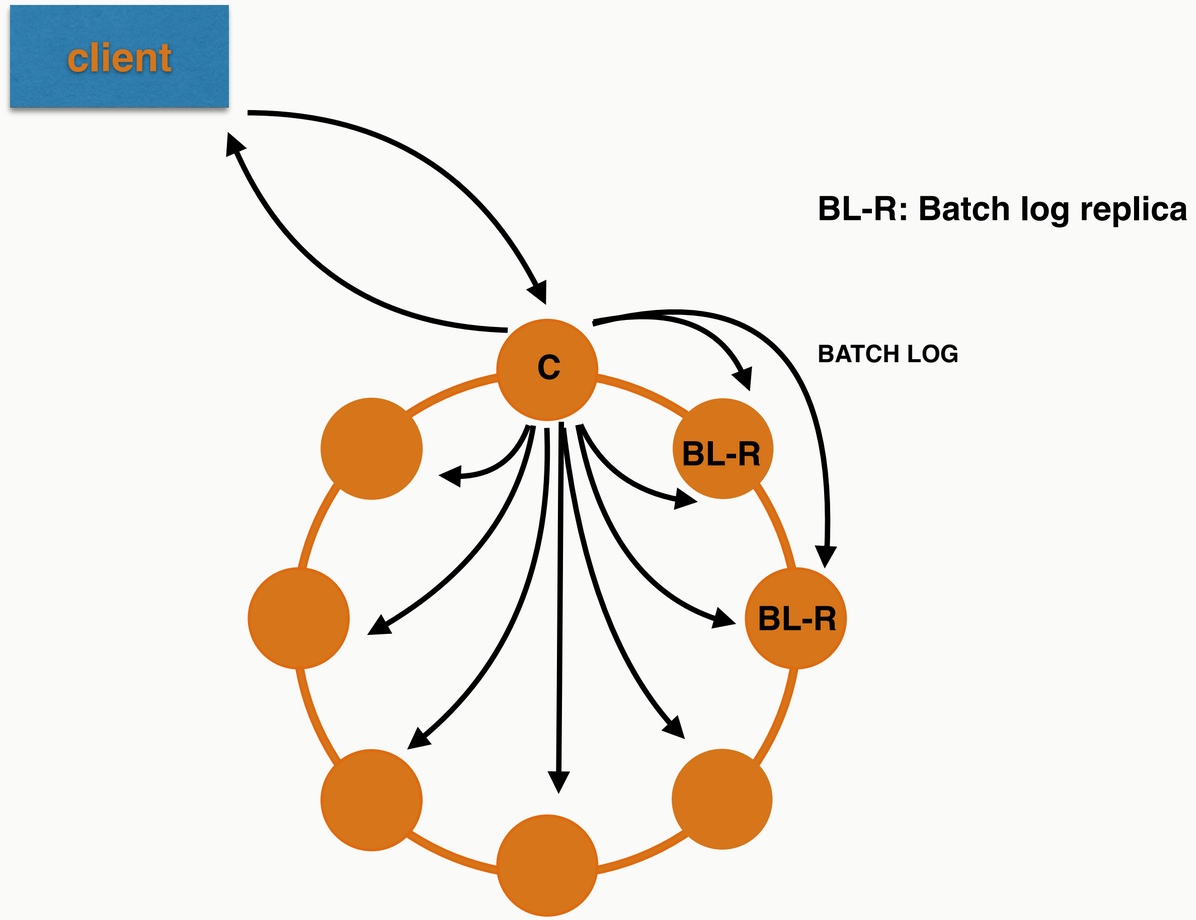
Logged Batches

- Once accepted the statements will eventually succeed
- Achieved by writing them to a distributed batchlog
- 30% slower than unlogged batches

Batch insert - Good idea?

```
public void storeEvents(ConsistencyLevel consistencyLevel, CustomerEvent... events) {
    BatchStatement batchStatement = new BatchStatement(BatchStatement.Type.LOGGED);

    for (CustomerEvent event : events) {
         BoundStatement boundInsert = insertStatement.bind(
         customerEvent.getCustomerId(),
         customerEvent.getTime(),
         customerEvent.getEventType(),
         customerEvent.getStaffId(),
         customerEvent.getStaffId());

         batchStatement.add(boundInsert);
    }
    session.execute(batchStatement);
}
```

Not so much



Use case?

```
CREATE TABLE if NOT EXISTS customer_events (
    customer_id text,
    statff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (customer_id, time))

CREATE TABLE if NOT EXISTS customer_events_by_staff (
    customer_id text,
    statff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (staff_id, time))
```

Storing events to both tables in a batch

```
public void storeEventLogged(ConsistencyLevel consistencyLevel, CustomerEvent customerEvent) {
    BoundStatement boundInsertForCustomerId = insertByCustomerId.bind(customerEvent.getCustomerId(),
       customerEvent.getTime(),
       customerEvent.getEventType(),
       customerEvent.getStaffId(),
       customerEvent.getStaffId());

    BoundStatement boundInsertForStaffId = insertByStaffId.bind(customerEvent.getCustomerId(),
       customerEvent.getTime(),
       customerEvent.getEventType(),
       customerEvent.getStaffId(),
       customerEvent.getStaffId());

    BatchStatement batchStatement = new BatchStatement(BatchStatement.Type.LOGGED);
    batchStatement.enableTracing();
    batchStatement.setConsistencyLevel(consistencyLevel);
    batchStatement.add(boundInsertForCustomerId);
    batchStatement.add(boundInsertForStaffId);


    ResultSet execute = session.execute(batchStatement);
}
```

Mutable data

Distributed mutable state :-(

```
create TABLE accounts(customer text PRIMARY KEY, balance_in_pence int);
```

customer	balanace_in_pence
luket	-10000
chbatey	10000
patrick	500

Overly naive example

- Cassandra uses last write wins (no Vector clocks)
- Read before write is an anti-pattern - race conditions and latency

Take a page from Event sourcing

```
create table accounts_log(
  customer text,
  time timeuuid,
  delta int,
  primary KEY (customer, time));
```

All records for the same customer in the same storage row

Transactions ordered by TIMEUUID - UUIDs are your friends in distributed systems

customer	time	delta
chbatey	4344c720-c1c0-11e4-81a7-1f0a3e4ebeaa	100
chbatey	44ea12b0-c1c0-11e4-81a7-1f0a3e4ebeaa	200
chbatey	47688030-c1c0-11e4-81a7-1f0a3e4ebeaa	-50
patrick	4d326f30-c1c0-11e4-81a7-1f0a3e4ebeaa	25
patrick	4f811a20-c1c0-11e4-81a7-1f0a3e4ebeaa	125

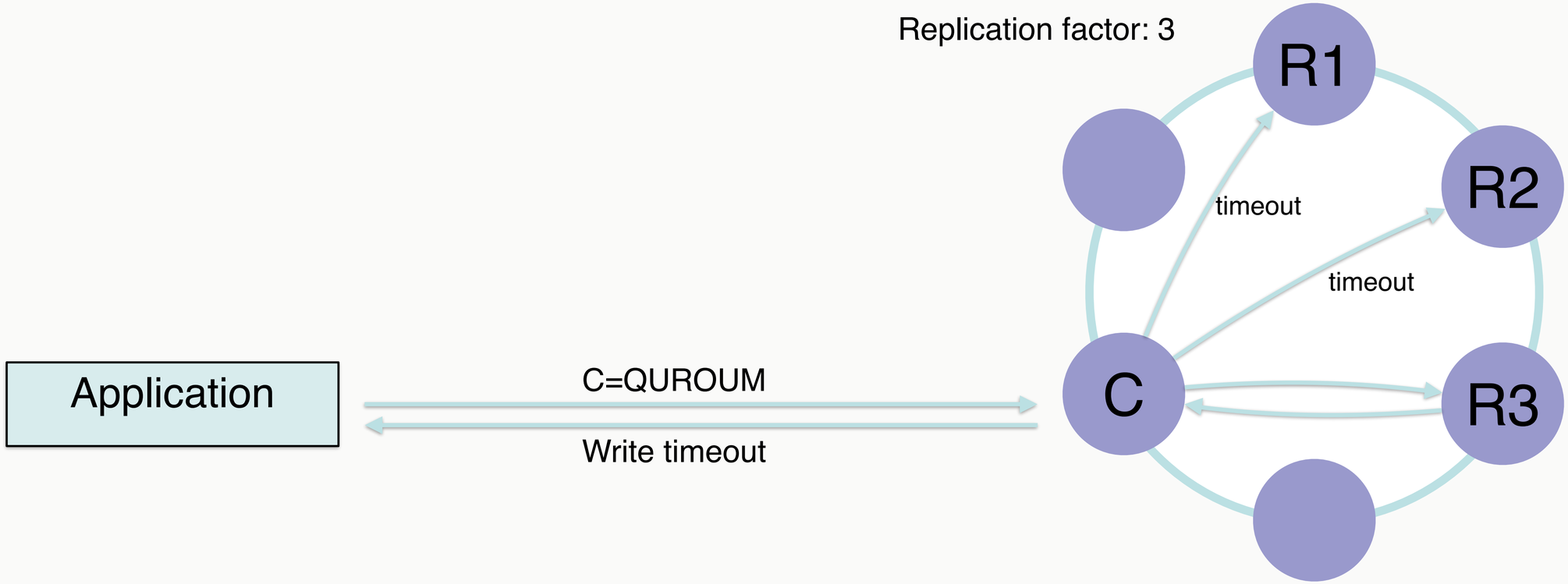
Tips

- Avoid mutable state in distributed system, favour immutable log
- Can roll up and snapshot to avoid calculation getting too big
- If you really have to then checkout LWTs and do via CAS - this will be slower

DATASTAX

Understanding failures (crazy retry policies)

Write timeout



Retrying writes

- Cassandra does not roll back!
- R3 has the data and the coordinator has hinted for R1 and R2

Write timeout

- Received acknowledgements
- Required acknowledgements
- Consistency level
- CAS and Batches are more complicated:
  - WriteType: SIMPLE, BATCH, UNLOGGED_BATCH, BATCH_LOG, CAS

Batches

- BATCH_LOG
  - Timed out waiting for batch log replicas
- BATCH
  - Written to batch log but timed out waiting for actual replica
  - Will eventually be committed
- UNLOGGED_BATCH

Idempotent writes

- All writes are idempotent with the following exceptions:
  - Counters
  - lists

DATASTAX

Cassandra as a Queue (tombstones)

Well documented anti-pattern

- http://www.datastax.com/dev/blog/cassandra-anti-patterns-queues-and-queue-like-datasets
- http://lostechies.com/ryansvihla/2014/10/20/domain-modeling-around-deletes-or-using-cassandra-as-a-queue-even-when-you-know-better/

Requirements

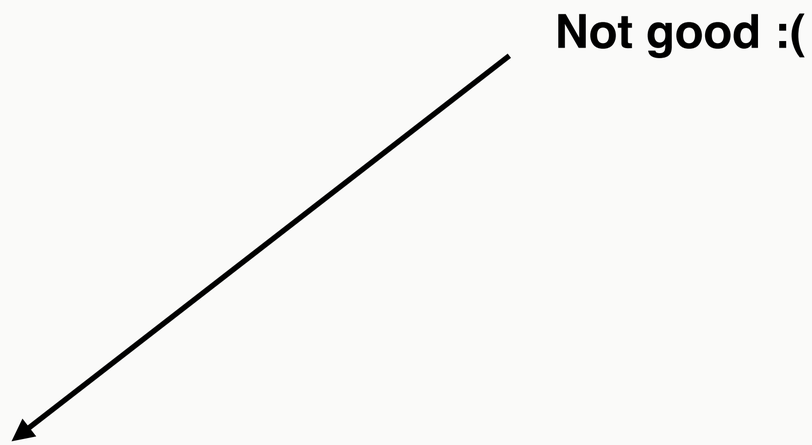
- Produce data in one DC
- Consume it once and delete from another DC
- Use Cassandra?

Datamodel

```
CREATE TABLE queues (
    name text,
    enqueued_at timeuuid,
    payload blob,
    PRIMARY KEY (name, enqueued_at)
);

SELECT enqueued_at, payload
  FROM queues
 WHERE name = 'queue-1'
 LIMIT 1;
```

Trace

```
activity                                       | source    | elapsed
-----------------------------------------------+-----------+--------
                           execute_cql3_query | 127.0.0.3 |       0
                            Parsing statement | 127.0.0.3 |      48
                           Peparing statement | 127.0.0.3 |     362
             Message received from /127.0.0.3 | 127.0.0.1 |      42
                Sending message to /127.0.0.1 | 127.0.0.3 |     718
Executing single-partition query on queues | 127.0.0.1 |     145
                 Acquiring sstable references | 127.0.0.1 |     158
                    Merging memtable contents | 127.0.0.1 |     189
Merging data from memtables and 0 sstables | 127.0.0.1 |     235
    Read 1 live and 19998 tombstoned cells | 127.0.0.1 |  251102
             Enqueuing response to /127.0.0.3 | 127.0.0.1 |  252976
                Sending message to /127.0.0.3 | 127.0.0.1 |  253052
             Message received from /127.0.0.1 | 127.0.0.3 |  324314
```



Tips

- Lots of deletes + range queries in the same partition
- Avoid it with data modelling / query pattern:
  - Move partition
  - Add a start for your range: e.g. enqueued_at > 9d1cb818-9d7a-11b6-96ba-60c5470cbf0e

Secondary Indices

Customer events table

```
CREATE TABLE if NOT EXISTS customer_events (
    customer_id text,
    staff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (customer_id, time))
```

```
cqlsh:anti_patterns> select * from customer_events where staff_id = 'chbatey';
Bad Request: No indexed columns present in by-columns clause with Equal operator
```

```
create INDEX on customer_events (staff_id) ;
```

Indexes to the rescue?

customer_id	time	staff_id
chbatey	2015-03-03 08:52:45	trevor
chbatey	2015-03-03 08:52:54	trevor
chbatey	2015-03-03 08:53:11	bill
chbatey	2015-03-03 08:53:18	bill
rusty	2015-03-03 08:56:57	bill
rusty	2015-03-03 08:57:02	bill
rusty	2015-03-03 08:57:20	trevor

staff_id	customer_id
trevor	chbatey
trevor	chbatey
bill	chbatey
bill	chbatey
bill	rusty
bill	rusty
trevor	rusty

Inserts

```
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'rusty', 42730bf0-c183-11e4-a4c6-1971740a12cd, 'SELL', 'trevor', 'WEB');
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'rusty', 3f07cd70-c183-11e4-a4c6-1971740a12cd, 'BUY', 'trevor', 'WEB');
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'chbatey', a9550c70-c182-11e4-a4c6-1971740a12cd, 'BUY', 'trevor', 'WEB');
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'chbatey', aebbf3e0-c182-11e4-a4c6-1971740a12cd, 'SELL', 'trevor', 'WEB');
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'chbatey', b8c2f050-c182-11e4-a4c6-1971740a12cd, 'VIEW', 'bill', 'WEB');
INSERT INTO customer_events (customer_id, time , event_type , staff_id, store_type ) VALUES
( 'chbatey', bcb5ae50-c182-11e4-a4c6-1971740a12cd, 'BUY', 'bill', 'WEB');
```

Secondary index are local

- The staff_id partition in the secondary index is not distributed like a normal table
- The secondary index entries are only stored on the node that contains the customer_id partition

Indexes to the rescue?

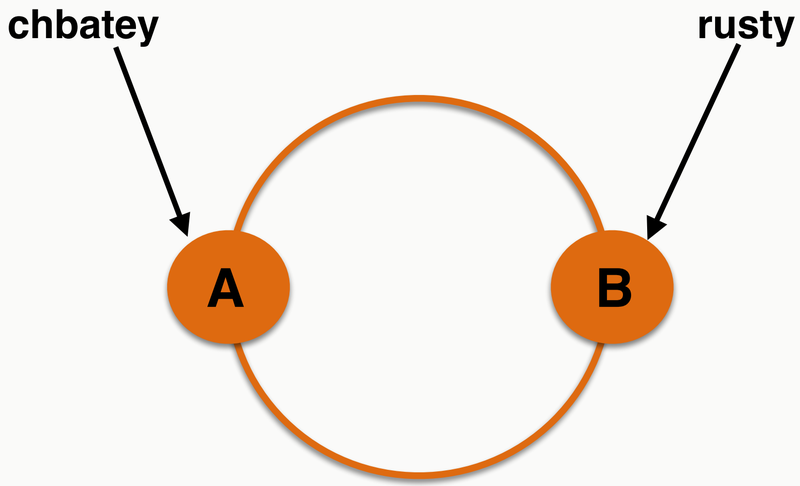
customer_events table

customer_id	time	staff_id
chbatey	2015-03-03 08:52:45	trevor
chbatey	2015-03-03 08:52:54	trevor
chbatey	2015-03-03 08:53:11	bill
chbatey	2015-03-03 08:53:18	bill
rusty	2015-03-03 08:56:57	bill
rusty	2015-03-03 08:57:02	bill
rusty	2015-03-03 08:57:20	trevor

staff_id index

staff_id	customer_id
trevor	chbatey
trevor	chbatey
bill	chbatey
bill	chbatey
bill	rusty
bill	rusty
trevor	rusty

staff_id	customer_id
trevor	chbatey
trevor	chbatey
bill	chbatey
bill	chbatey



staff_id	customer_id
bill	rusty
bill	rusty
trevor	rusty

Homework

1. Use a cluster with 6 nodes (CCM)
2. Create the customer events table and add the secondary index
3. Insert the data (insert statements are in a hidden slide I'll distribute)
4. Turn tracing on and execute the following queries:
   A. select * from customer_events where staff_id = 'trevor' ;
   B. select * from customer_events where staff_id = 'trevor' and customer_id = 'chbatey'
5. How many partitions queried and nodes were used for each query?
6. Send me the answer on twitter, first couple get SWAG!

Do it your self index

```
CREATE TABLE if NOT EXISTS customer_events (
    customer_id text,
    statff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (customer_id, time))

CREATE TABLE if NOT EXISTS customer_events_by_staff (
    customer_id text,
    statff_id text,
    store_type text,
    time timeuuid ,
    event_type text,
    PRIMARY KEY (staff_id, time))
```

Possibly in 3.0 - Global indexes

Cassandra / CASSANDRA-6477

Global indexes

Agile Board

Details

Type:	New Feature	Status:	OPEN
Priority:	Major	Resolution:	Unresolved
Component/s:	API, Core	Fix Version/s:	3.0
Labels:	cql		

Description

Local indexes are suitable for low-cardinality data, where spreading the index across the cluster is a Good Thing. However, for high-cardinality data, local indexes require querying most nodes in the cluster even if only a handful of rows is returned.

Issue Links

is related to CASSANDRA-8517 Make client-side Token Aware Balancing possible with Globa… OPEN

Activity

Anti patterns summary

- Most anti patterns are very obvious in trace output
- Don’t test on small clusters, most problems only occur at scale

Thanks for listening

- Check out my blog: http://christopher-batey.blogspot.co.uk/
  - Answers for all questions
  - Detailed posts on most of the anti patterns + full working code
- More questions? Contact me on twitter: @chbatey
- Learn more about read / write path, CCM checkout DataStax academy
  - https://academy.datastax.com/


KEEP
CALM
AND
ASK
QUESTIONS