MariaDB Cassandra interoperability

Cassandra Storage Engine in MariaDB

Sergei Petrunia
Colin Charles

PERCONA LIVE

Who are we

- Sergei Petrunia
  - Principal developer of CassandraSE, optimizer developer, formerly from MySQL
  - psergey@mariadb.org
- Colin Charles
  - Chief Evangelist, MariaDB, formerly from MySQL
  - colin@mariadb.org

Agenda

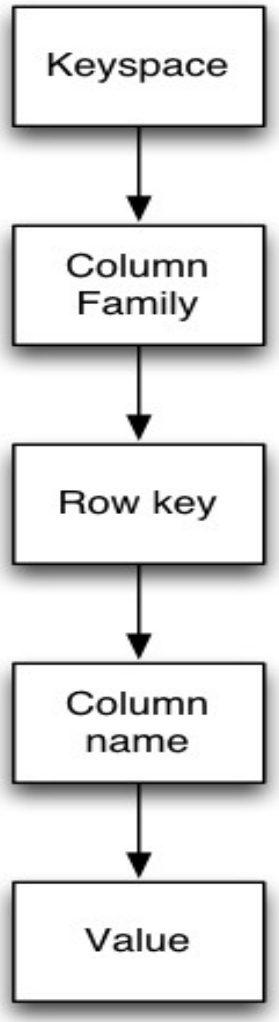
- An introduction to Cassandra
- The Cassandra Storage Engine (Cassandra SE)
- Data mapping
- Use cases
- Benchmarks
- Conclusion

Background: what is Cassandra

- A distributed NoSQL database
  - Key-Value store
    - Limited range scan suppor
  - Optionally flexible schema
    - Pre-defined “static” columns
    - Ad-hoc “dynamic” columns
  - Automatic sharding / replication
  - Eventual consistency

Background: Cassandra's data model

- “Column families” like tables
- Row key → columns
- Somewhat similar to SQL but some important differences.
- Supercolumns are not supported



CQL – Cassandra Query Language

Looks like SQL at first glance

```
bash$ cqlsh -3
cqlsh> CREATE KEYSPACE mariadbtest
   ...   WITH REPLICATION ={'class':'SimpleStrategy','replication_factor':1};

cqlsh> use mariadbtest;

cqlsh:mariadbtest> create columnfamily cf1 ( pk varchar primary key,
               ...   data1 varchar, data2 bigint
               ...  ) with compact storage;

cqlsh:mariadbtest> insert into cf1 (pk, data1,data2)
               ...   values ('row1', 'data-in-cassandra', 1234);

cqlsh:mariadbtest> select * from cf1;

 pk    | data1              | data2
------+--------------------+-------
 row1 | data-in-cassandra |  1234
```

CQL is not SQL

Similarity with SQL is superficial

```
cqlsh:mariadbtest> select * from cf1 where pk='row1';

 pk    | data1               | data2
------+---------------------+-------
 row1 | data-in-cassandra |  1234

cqlsh:mariadbtest> select * from cf1 where data2=1234;
Bad Request: No indexed columns present in by-columns clause with Equal
operator

cqlsh:mariadbtest> select * from cf1 where pk='row1' or pk='row2';
Bad Request: line 1:34 missing EOF at 'or'
```

- No joins or subqueries
- No GROUP BY, ORDER BY must be able to use available indexes
- WHERE clause must represent an index lookup.

Cassandra Storage Engine

~~Starts a NoCQL movement~~

Provides a “view” of Cassandra's data from MariaDB.

1. Load the Cassandra SE plugin

- Get MariaDB 10.0.1+
- Load the Cassandra plugin
  - From SQL:

```
MariaDB [(none)]> install plugin cassandra soname 'ha_cassandra.so';
```

  - Or, add a line to my.cnf:

```
[mysqld]
...
plugin-load=ha_cassandra.so
```

- Check it is loaded

```
MariaDB [(none)]> show plugins;
+--------------------+--------+----------------+-----------------+---------+
| Name               | Status | Type           | Library         | License |
+--------------------+--------+----------------+-----------------+---------+
...
| CASSANDRA          | ACTIVE | STORAGE ENGINE | ha_cassandra.so | GPL     |
+--------------------+--------+----------------+-----------------+---------+
```

2. Connect to Cassandra

- Create an SQL table which is a view of a column family

```
MariaDB [test]> set global cassandra_default_thrift_host='10.196.2.113';
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> create table t2 (pk varchar(36) primary key,
    ->               data1 varchar(60),
    ->               data2 bigint
    ->             ) engine=cassandra
    ->               keyspace='mariadbtest'
    ->              thrift_host='10.196.2.113'
    ->              column_family='cf1';
Query OK, 0 rows affected (0.01 sec)
```

  - thrift_host can be set per-table
  - @@cassandra_default_thrift_host allows to
    - Re-point the table to different node dynamically
    - Not change table DDL when Cassandra IP changes.

Possible gotchas

- SELinux blocks the connection

```
MariaDB [test]> create table t1 ( ... ) engine=cassandra  ... ;
ERROR 1429 (HY000): Unable to connect to foreign data source: connect()
failed: Permission denied [1]
```

  - Packaging bug
  - To get running quickly: `echo 0 >/selinux/enforce`

- Cassandra 1.2 and CFs without “COMPACT STORAGE”

```
MariaDB [test]> create table t1 ( ... ) engine=cassandra  ... ;
ERROR 1429 (HY000): Unable to connect to foreign data source: Column family
cf1 not found in keyspace mariadbtest
```

  - Caused by a change in Cassandra 1.2
  - They broke Pig also
  - We intend to update CassandraSE for 1.2

Accessing Cassandra data

- Can get Cassandra's data

```
MariaDB [test]> select * from t2;
+------+-------------------+-------+
| pk   | data1             | data2 |
+------+-------------------+-------+
| row1 | data-in-cassandra |  1234 |
+------+-------------------+-------+
```

- Can insert data

```
MariaDB [test]> insert into t2 values ('row2','data-from-mariadb', 123);
Query OK, 1 row affected (0.00 sec)
```

- Cassandra sees inserted data

```
cqlsh:mariadbtest> select * from cf1;

 pk   | data1             | data2
------+-------------------+-------
 row1 | data-in-cassandra |  1234
 row2 | data-from-mariadb |   123
```

Data mapping between Cassandra and SQL

Data mapping between Cassandra and SQL

```
create table tbl (
    pk varchar(36) primary key,
    data1 varchar(60),
    data2 bigint
) engine=cassandra keyspace='ks1' column_family='cf1'
```

- MariaDB table represents Cassandra's Column Family
  - Can use any table name, column_family=... specifies CF.

Data mapping between Cassandra and SQL

```
create table tbl (
   pk varchar(36) primary key,
   data1 varchar(60),
   data2 bigint
) engine=cassandra keyspace='ks1' column_family='cf1'
```

- MariaDB table represents Cassandra's Column Family
  - Can use any table name, column_family=... specifies CF.
- Table must have a primary key
  - Name/type must match Cassandra's rowkey

Data mapping between Cassandra and SQL

```
create table tbl (
    pk varchar(36) primary key,
    data1 varchar(60),
    data2 bigint
) engine=cassandra keyspace='ks1' column_family='cf1'
```

- MariaDB table represents Cassandra's Column Family
  - Can use any table name, column_family=... specifies CF.
- Table must have a primary key
  - Name/type must match Cassandra's rowkey
- Columns map to Cassandra's static columns
  - Name must be the same as in Cassandra
  - Datatypes must match
  - Can any subset of CF's columns

Datatype mapping

- CF column datatype determines MariaDB datatype

Cassandra	MariaDB
blob	BLOB, VARBINARY(n)
ascii	BLOB, VARCHAR(n), use charset=latin1
text	BLOB, VARCHAR(n), use charset=utf8
varint	VARBINARY(n)
int	INT
bigint	BIGINT, TINY, SHORT
uuid	CHAR(36) (text in MariaDB)
timestamp	TIMESTAMP (second precision), TIMESTAMP(6) (microsecond precision), BIGINT
boolean	BOOL
float	FLOAT
double	DOUBLE
decimal	VARBINARY(n)
counter	BIGINT

Dynamic columns

- Cassandra supports “dynamic column families”
- Can access ad-hoc columns with MariaDB's dynamic columns feature

```
create table tbl
(
  rowkey  type PRIMARY KEY

  column1 type,
  ...
  dynamic_cols blob DYNAMIC_COLUMN_STORAGE=yes
) engine=cassandra keyspace=... column_family=...;

insert into tbl values
   (1, column_create('col1', 1, 'col2', 'value-2'));

select rowkey,
       column_get(dynamic_cols, 'uuidcol' as char)
from tbl;
```

Data mapping is safe

- Cassandra SE will refuse incorrect mappings

```
create table t3 (pk varchar(60) primary key, no_such_field int)
  engine=cassandra `keyspace`='mariadbtest' `column_family`='cf1';

ERROR 1928 (HY000): Internal error: 'Field `no_such_field` could not
be mapped to any field in Cassandra'
```

```
create table t3 (pk varchar(60) primary key, data1 double)
  engine=cassandra `keyspace`='mariadbtest' `column_family`='cf1';

ERROR 1928 (HY000): Internal error: 'Failed to map column data1
to datatype org.apache.cassandra.db.marshal.UTF8Type'
```

Command mapping

Command Mapping

- Cassandra commands
  - PUT (upsert)
  - GET
    - Scan
  - DELETE (if exists)
- SQL commands
  - SELECT → GET/Scan
  - INSERT → PUT (upsert)
  - UPDATE/DELETE → read+write.

SELECT command mapping

- MariaDB has an SQL interpreter
- Cassandra SE supports lookups and scans
- Can now do
  - Arbitrary WHERE clauses
  - JOINs between Cassandra tables and MariaDB tables
    - Batched Key Access is supported

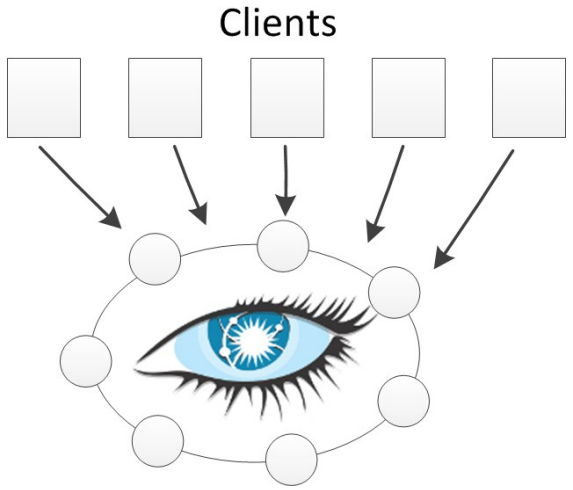
DML command mapping

- No SQL semantics
  - INSERT overwrites rows
  - UPDATE reads, then writes
    - Have you updated what you read
  - DELETE reads, then deletes
    - Can't be sure if/what you have deleted
- Not as bad as it sounds, it's Cassandra
  - Cassandra SE doesn't make it SQL.

Cassandra SE use cases

Cassandra use cases

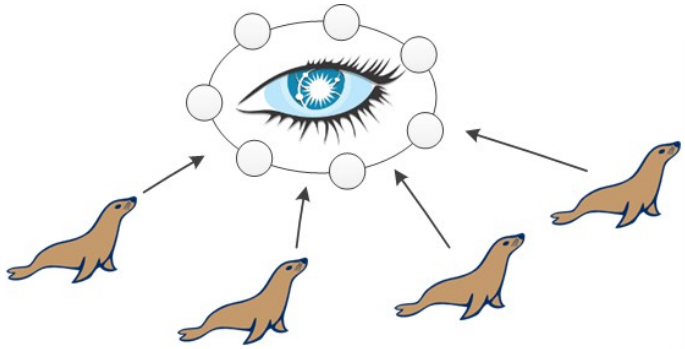
- Collect massive amounts of data
  - Web page hits
  - Sensor updates



- Updates are naturally non-conflicting
  - Keyed by UUIDs, timestamps
- Reads are served with one lookup
- Good for certain kinds of data
  - Moving from SQL entirely may be difficult

Cassandra SE use cases (1)

Access Cassandra data from SQL

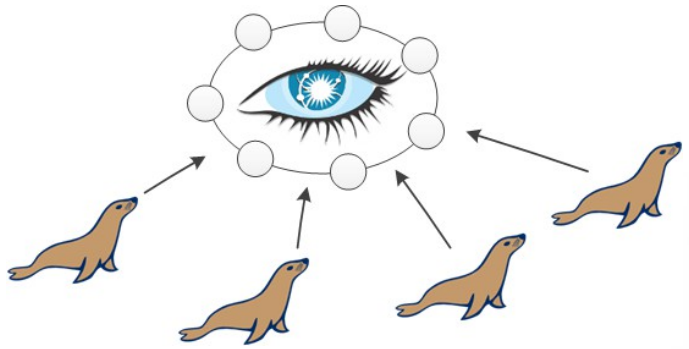


- Send an update to Cassandra
  - Be a sensor
- Grab a piece of data from Cassandra
  - “This web page was last viewed by …”
  - “Last known position of this user was …”.

Cassandra SE use cases (2)

Coming from MySQL/MariaDB side:

- Want a special table that is
  - auto-replicated
  - fault-tolerant
  - Very fast?



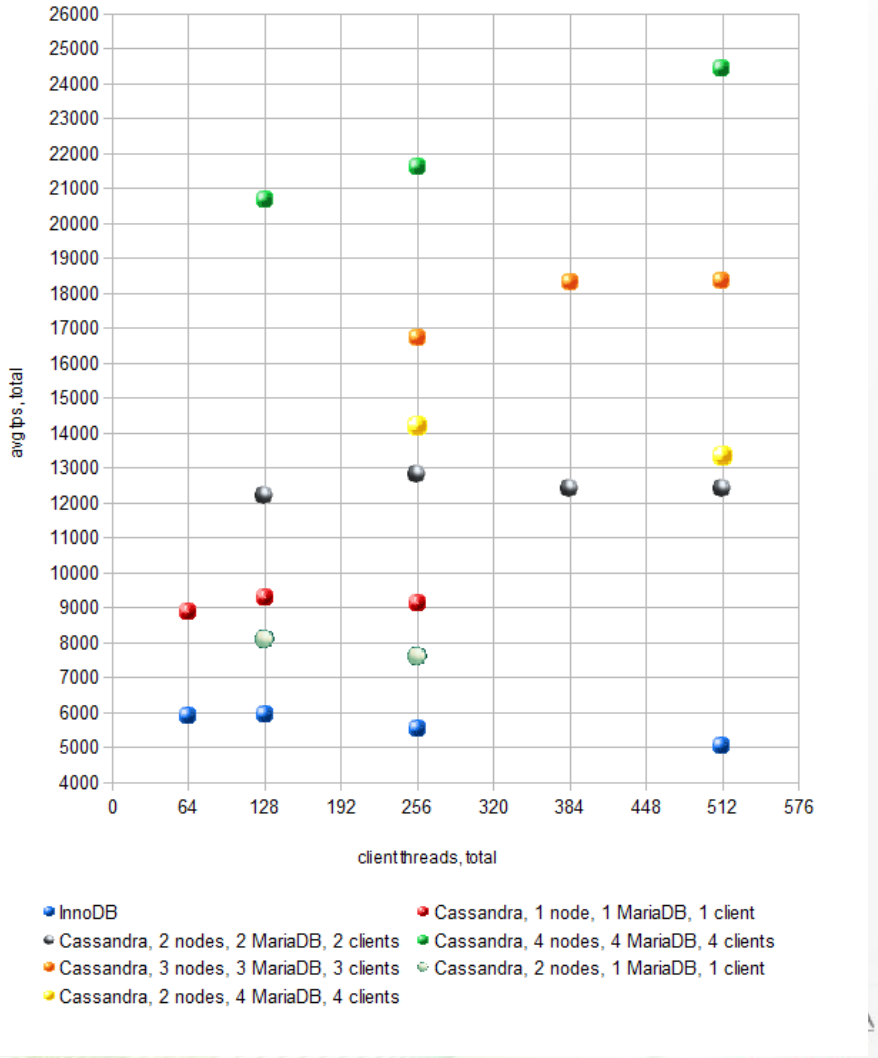
- Get Cassandra, and create a Cassandra SE table.

Cassandra Storage Engine non-use cases

- Huge, sift-through-all-data joins
  - Use Pig
- Bulk data transfer to/from Cassandra cluster
  - Use Sqoop
- A replacement for InnoDB
  - No full SQL semantics

A “benchmark”

- One table
- EC2 environment
  - m1.large nodes
  - Ephemeral disks
- Stream of single-line INSERTs
- Tried Innodb and Cassandra
- Hardly any tuning



Conclusions

- Cassandra SE can be used to peek at data in Cassandra from MariaDB.
- It is not a replacement for Pig/Hive
- It is really easy to setup and use

Going Forward

- Looking for input
- Do you want support for
  - Fast counter columns updates?
  - Awareness of Cassandra cluster topology?
  - Secondary indexes?
  - …?

Resources

- https://kb.askmonty.org/en/cassandrase/
- http://wiki.apache.org/cassandra/DataModel
- http://cassandra.apache.org/
- http://www.datastax.com/docs/1.1/ddl/column_family

Thanks!

Q & A

Extra: Cassandra SE internals

- Developed against Cassandra 1.1
- Uses Thrift API
  - cannot stream CQL resultset in 1.1
  - Cant use secondary indexes
- Only supports AllowAllAuthenticator
- In Cassandra 1.2
  - “CQL Binary Protocol” with streaming
  - CASSANDRA-5234: Thrift can only read CFs “WITH COMPACT STORAGE”