Elassandra :

Elasticsearch as a Cassandra Secondary Index

September the 8th, 2016 - LLD20

Elassandra : Elasticsearch as a Cassandra secondary index

&

Vincent Royer
Elassandra Author

Rémi Trouville
Strapdata Co-Founder

We have been working together for 8 years in the banking/insurance industry

Today's objectives :

- Sharing our vision and excitement about our project
- Receiving feedback from you all about elassandra
- Meeting NOSQL gurus to exchange ideas, solutions, questions, … and beers

1	Introduction
2	How Elassandra works ?
3	Cool Features
4	Elassandra’s ecosystem
5	Roadmap
6	Q&A

Elassandra's status (2016/09/08)

Current Usage

- Used for non-critical data including
  - Application logs
  - Server monitoring (CPU, memory…)
  - Consolidation and reporting from various SQL databases.

Current status of Elassandra

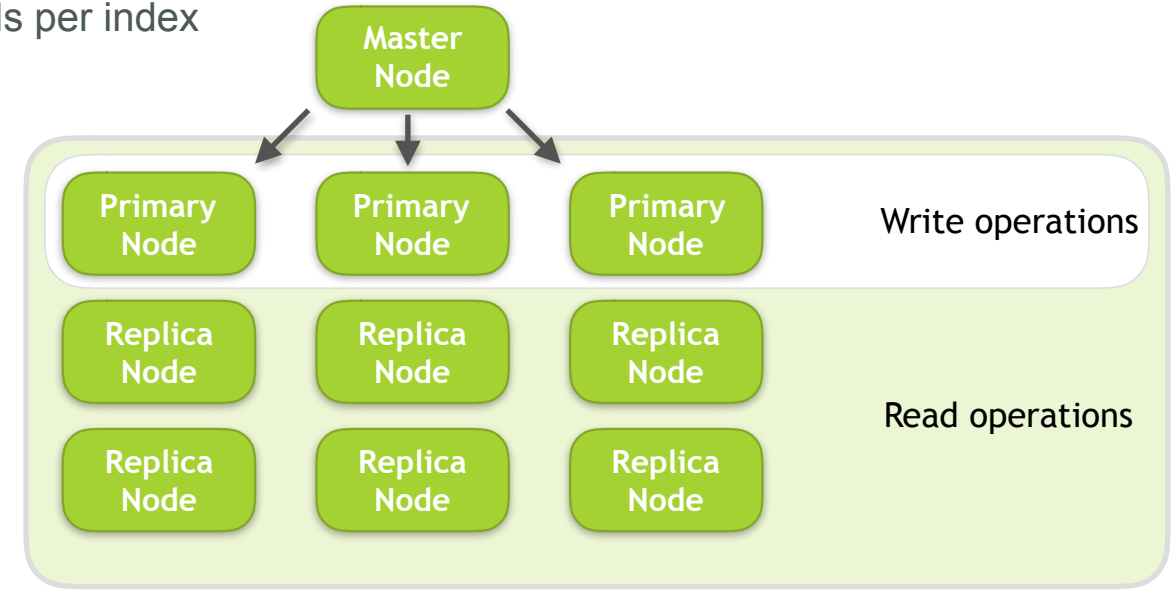
- Still in beta version
- Needs testing on larger deployments

Production-ready targeted End of 2016

1	Introduction
2	How Elassandra works ?
3	Cool Features
4	Elassandra’s ecosystem
5	Roadmap
6	Q&A

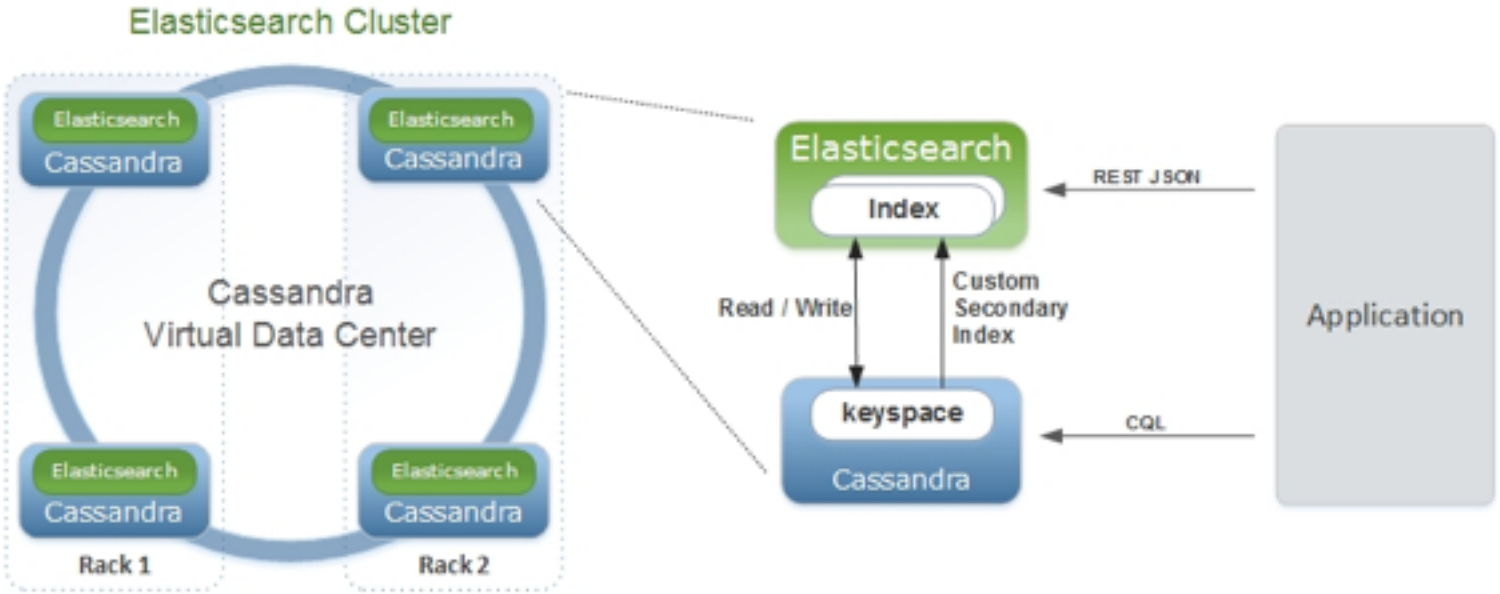
Elasticsearch design

- The master node manages and broadcasts the cluster state
- Only primary nodes supports write operations
- On failure, a new master or primary node is elected
- By default, 5 shards per index



Elassandra design

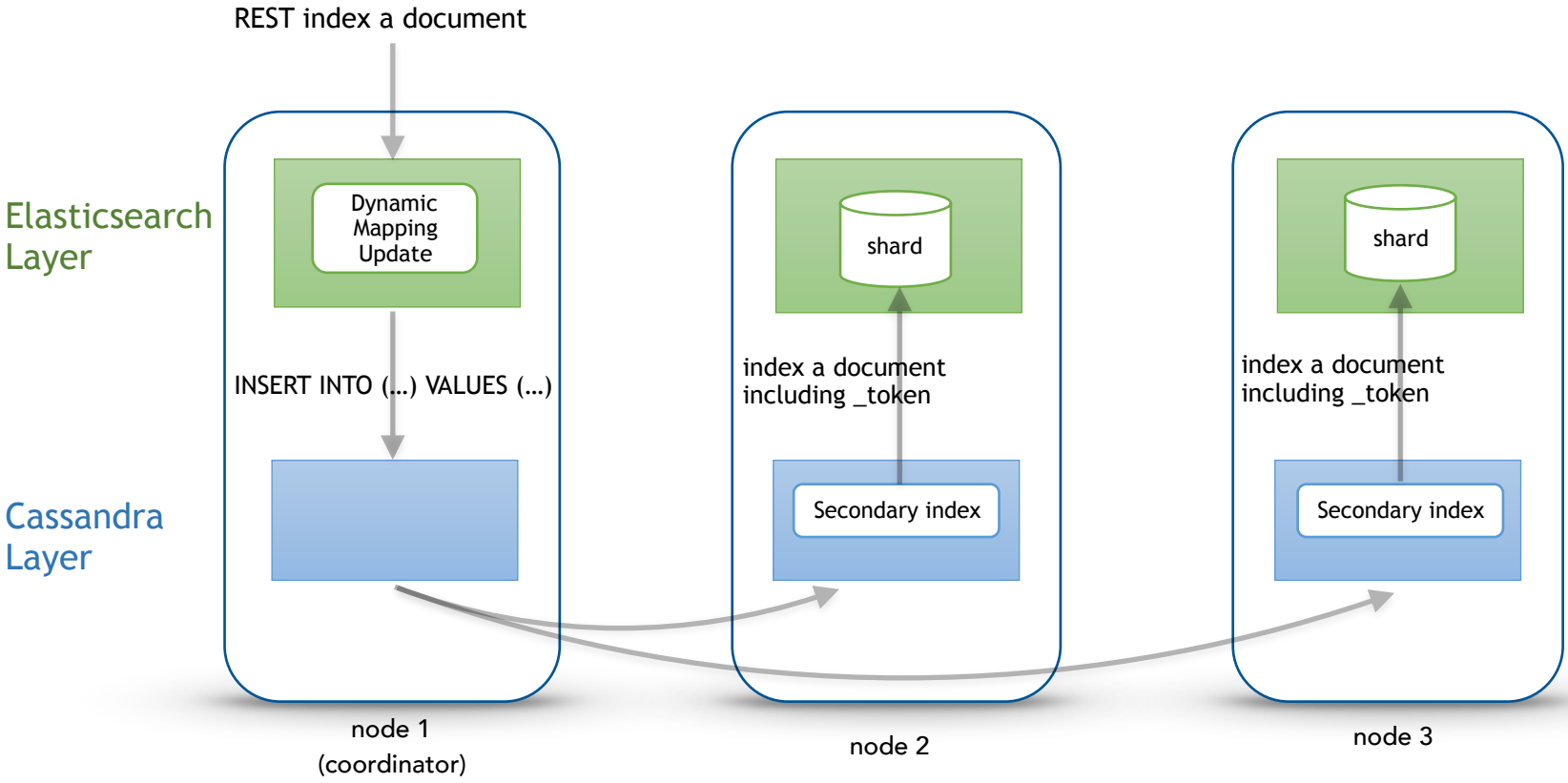
- Elasticsearch code is embedded in Cassandra nodes
- Documents are stored as row in a Cassandra tables (no more _source in Elasticsearch)
- A custom secondary index synchronously updates elasticsearch indices



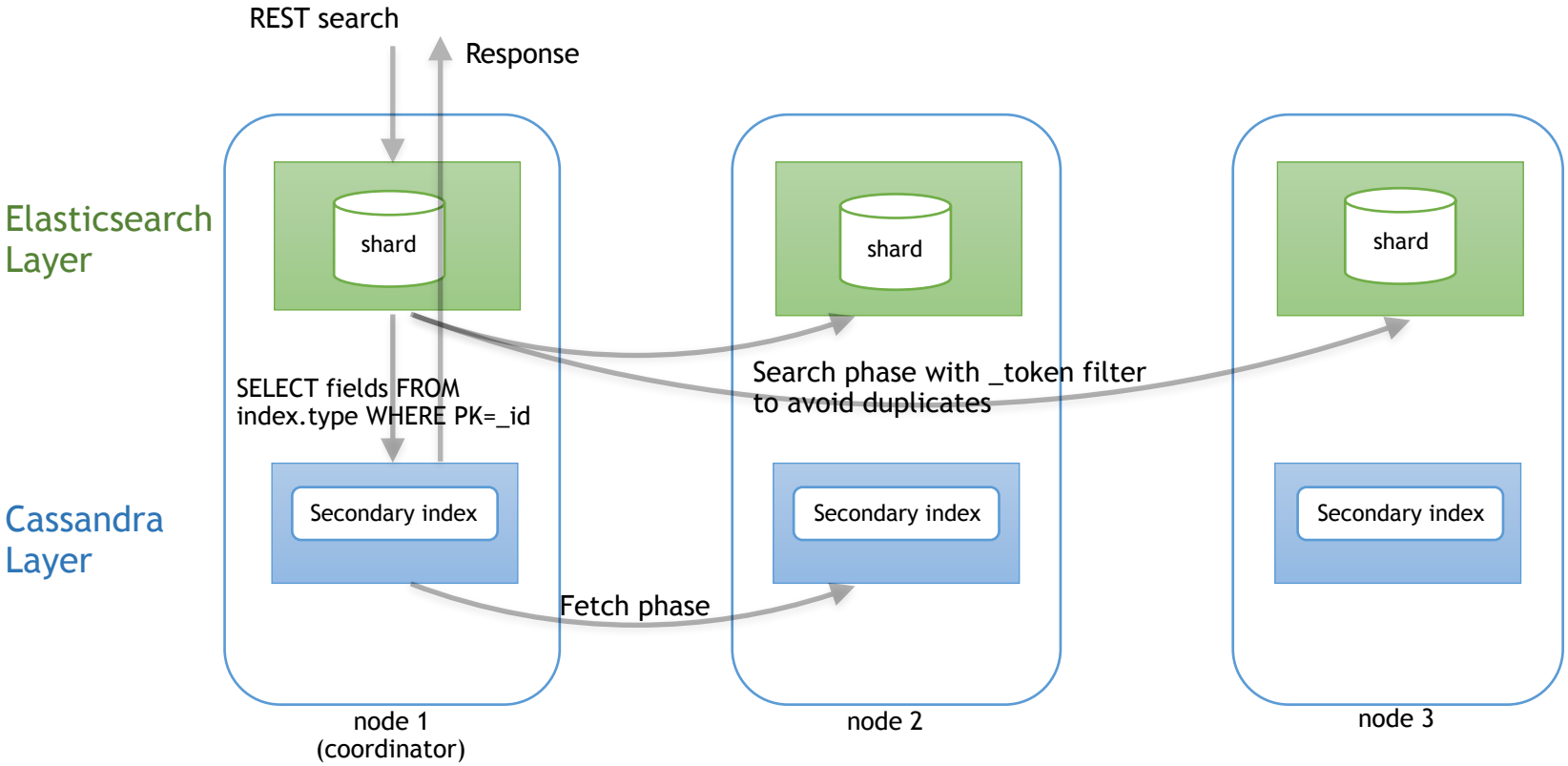
Terminology

Elasticsearch	Cassandra	Description
Mapping	Schema	Defines data structures
Cluster	Virtual Datacenter	An elassandra datacenter is an elastic search cluster
Index	Keyspace	An index relies on a keyspace.
Type	Table	Each document type is stored in a cassandra table
Document	Row	A document is stored as a cassandra row where _id is the primary key.
Field	Column	Each indexed field is backed by a cassandra column
Object or nested fields	User Defined Type	Automatically created User Defined Type to store elasticsearch objects.

Elassandra Write Path



Elassandra Search Path



Elasticsearch cluster state

Cluster state has 3 main sections :

1. Cluster information (cluster name, node ids)
2. Metadata (mapping definition, indices and data structures, stored in a Cassandra)
3. Routing table to route search operations (Built locally from the Cassandra topology)

```
{
  "cluster_name" : "Test Cluster",
  "version" : 7,
  "state_uuid" : "SkMDaaB-RA6n0DhmHZaTow",
  "master_node" : "e8b9c9f0-0c07-4845-9c02-211a4dbf7ea6",
  "blocks" : { },
  "nodes" : {
    "e8b9c9f0-0c07-4845-9c02-211a4dbf7ea6" : {
      "name" : "localhost",
      "status" : "ALIVE",
      "transport_address" : "127.0.0.1:9300",
      "attributes" : {
        "rack" : "rack1",
        "data" : "true",
        "data_center" : "dc1",
        "master" : "true"
      }
    }
  },
 }
```

```
"metadata" : {
    "version" : 2,
    "cluster_uuid" : "e8b9c9f0-0c07-4845-9c02-211a4dbf7ea6",
    "templates" : { },
    "indices" : {
      "twitter" : {
        "state" : "open",
        "settings" : {
          "index" : {
            "creation_date" : "1471681453347",
            "number_of_shards" : "1",
            "number_of_replicas" : "0",
            "uuid" : "j4zZS2eOTHaDcW3r1e_lDA",
            "version" : {
              "created" : "2010199"
            }
          }
        },
        "mappings" : {
          "tweet" : {
            "properties" : {
              "size" : { "type" : "long"},
              "post_date" : {
                "format" : "strict_date_optional_time||epoch_millis",
                "type" : "date"
              },
              "message" : {"type" : "string"},
              "user" : {"type" : « string" }
            }
          }
        }
```

```
"routing_table" : {
  "indices" : {
    "twitter" : {
      "shards" : {
        "0" : [ {
          "state" : "STARTED",
          "primary" : true,
          "node" : "e8b9c9f0-0c07-4845-9c02-211a4dbf7ea6",
          "relocating_node" : null,
          "shard" : 0,
          "index" : "twitter",
          "version" : 4,
          "token_ranges" : [ "(-9223372036854775808,9223372036854775807]" ],
          "allocation_id" : {
            "id" : "SdDlnqLXTuacrlHpaJkAwA"
          }
        } ]
      }
    }
  }
}
```

Elasticsearch mapping storage in Cassandra

Elasticsearch mapping is stored in :

- A Cassandra table **elastic_admin.metadata**
- In the internal cassandra system keyspace.

On node bootstrap (first start of a node)
Data are pulled from other nodes and are indexed in elasticsearch
=> Bootstrapping provides elasticsearch resharding.

On node startup :
Recovered data from commitlogs are indexed in elasticsearch.
=> This ensures consistency after a failure.

Masterless mapping management

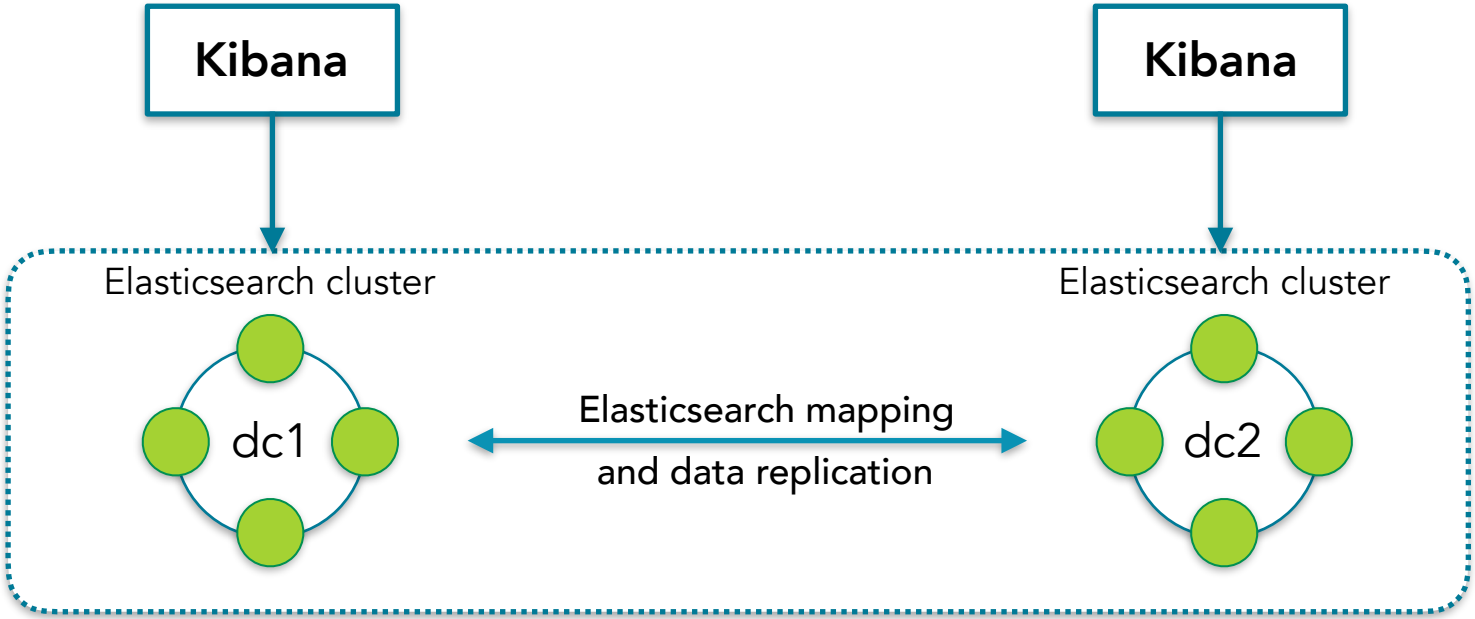
When a node update the elasticsearch mapping :

- A PAXOS transaction on **elastic_admin.metadata** table ensures no concurrent modification can be done.

- The GOSSIP protocol is used
  - to notify all the nodes to reload the new mapping
  - to check that all nodes have applied this new mapping
  - to broadcast shards status.

=> No more elasticsearch master node

Cross Datacenter Replication



Cassandra Hinted Handoff and Repair ensures data consistency

Elassandra : Backup & Restore

Backup Elasticsearch Lucene files like Cassandra SSTables

- Cassandra flush memtables and secondary indices when snapshotting
- Lucene file are immutable like cassandra SSTables
- Snapshot = hard link on immutable SSTables + lucene files.

Benefits :

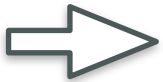
- Consistent backup of Cassandra and Elasticsearch indices
- Cassandra as a primary storage (No shared FS needed)

1	Introduction
2	How Elassandra works ?
3	Cool Features
4	Elassandra’s ecosystem
5	Roadmap
6	Q&A

Elassandra provides Bi-directionnal mapping

Inserting a document via elastic APIs creates/updates the underlying CQL schema

```
PUT /twitter/tweet/1 {
    "user" : "vince",
    "post_date" : "2009-11-15T14:12:12",
    "message" : "look at Elassandra",
    "size": 50
}
```



```
CREATE KEYSPACE twitter WITH …
CREATE TABLE twitter.tweet (
    "_id" text PRIMARY KEY,
    message   list<text>,
    post_date list<timestamp>,
    size      list<bigint>,
    user      list<text>
)
```

Discover the Elasticsearch mapping from an existing CQL schema

```
PUT /twitter/_mapping/tweet {
    "discover" : ".*"
}
```

```
PUT /twitter/_mapping/tweet {
    "twitter" : {
      "properties" : {
                "message" : {
                  "type" : "string"
                },
                "post_date" : {
                  "type" : "date",
                  "format" : "strict_date_optional_time||epoch_millis"
                },
                "size" : {
                  "type" : "long"
                },
                "user" : {
                  "type" : "string"
                }
            }
        }
}
```


CASSANDRA SUMMIT 2016

Elassandra supports nested documents with UDT

Nested documents are stored in a Cassandra **User Defined Type** dynamically generated from the Elasticsearch mapping.

```
curl -XPUT "http://$NODE:9200/directory/users/1" -d '{
  "group" : "fans",
  "name" : {
      "first" : "John",
      "last" :  "Smith"
    }
}'
```

```
CREATE KEYSPACE directory WITH replication = {'class': 'NetworkTopologyStrategy', 'dc1': '1'}  AND durable_writes = true;

CREATE TYPE directory.users_name (
    last frozen<list<text>>,
    first frozen<list<text>>
);

CREATE TABLE directory.users (
    "_id" text PRIMARY KEY,
    group list<text>,
    name list<frozen<users_name>>
);

CREATE CUSTOM INDEX elastic_users_name_idx ON directory.users (name) USING 'org.elasticsearch.cassandra.index.ExtendedElasticSecondaryIndex';
CREATE CUSTOM INDEX elastic_users_group_idx ON directory.users (group) USING 'org.elasticsearch.cassandra.index.ExtendedElasticSecondaryIndex';
```

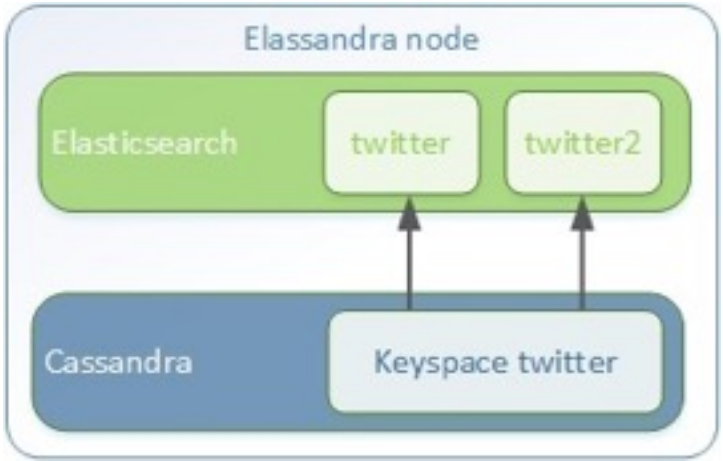
Many elasticsearch indices for a keyspace

A keyspace content can be indexed in many elasticsearch indices with various mappings.

Standard Cassandra index rebuild (use C* compaction manager threads) :

```
nodetool rebuild_index <keyspace> <tablename> elastic_<tablename>
```

Benefits : Change index mappings with zero downtime



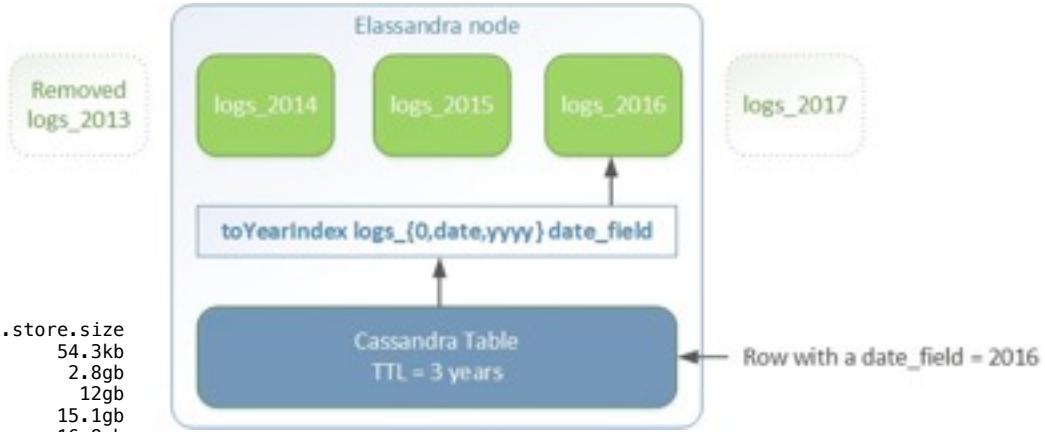
Partitioned indices for logs analysis with Kibana

At index time, a partition function builds the target elasticsearch index name.

- Time-frame indices are removed when too old.
- A default TTL on the underlying C* tables removes old logs.
- Comes with a cost : duplicate lucene term dictionaries.

```
curl -XPUT "http://localhost:9200/logs_${YEAR}" -d '{
   "settings":{
      "keyspace":"logs",
      "index.partition_function":"year logs_{0,date,yyyy} date_field" }
   }
}'
```

```
curl -s -XGET http://localhost:9200/_cat/indices?v
health status index       pri rep docs.count docs.deleted store.size pri.store.size
green  open   kibana        2   1         14            1     54.3kb         54.3kb
green  open   logs_2005     2   0   22770654       874988      2.8gb          2.8gb
green  open   logs_2006     2   0   93003294      5466480       12gb           12gb
green  open   logs_2007     2   0  118455836      4856867     15.1gb         15.1gb
green  open   logs_2008     2   0  131107405      5969785     16.8gb         16.8gb
green  open   logs_2009     2   0   58150615      1296827      7.4gb          7.4gb
green  open   logs_2010     2   0         23            2     86.8kb         86.8kb
green  open   logs_2011     2   0          0            0       142b           142b
```



CASSANDRA SUMMIT 2016

1 Introduction

2 How Elassandra works ?

3 Cool Features

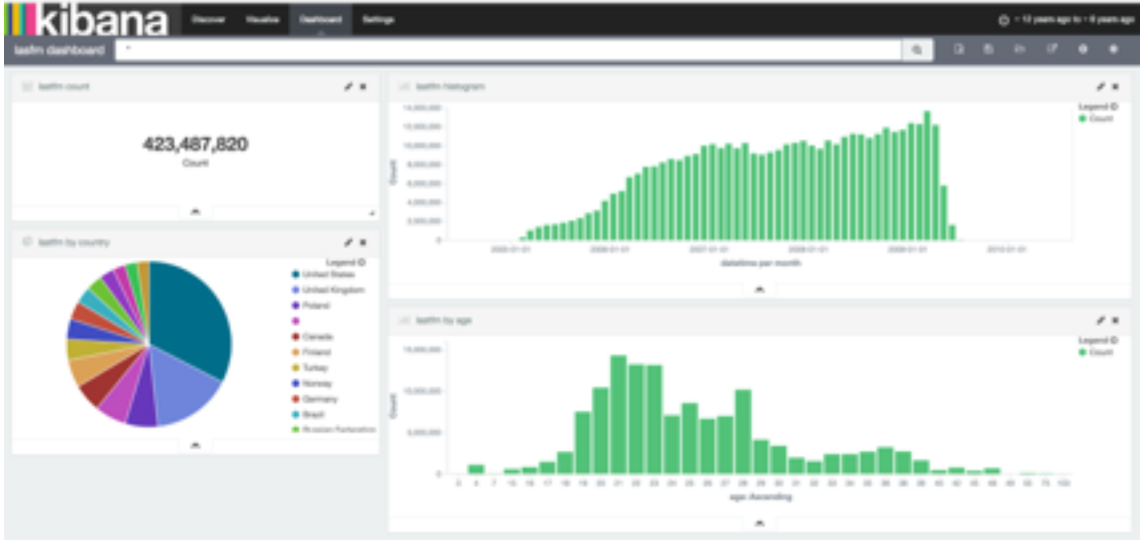
4 Elassandra’s ecosystem

5 Roadmap

6 Q&A

Elassandra + Kibana

Kibana for search and data visualization :



Even Kibana's configuration is in Cassandra

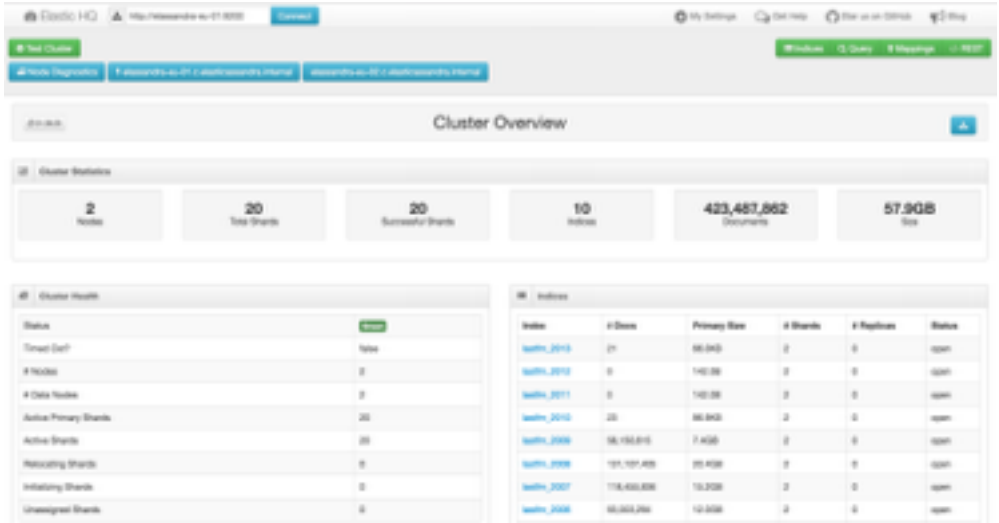
```
cqlsh> SELECT "_id",title,"kibanaSavedObjectMeta" FROM kibana.visualization;

 _id                  | title                    | kibanaSavedObjectMeta
----------------------+--------------------------+----------------------------------------------------------------------------------------------------------------------------
     lastfm-by-age |     ['lastfm by age'] | [{searchSourceJSON: ['{"index":"lastfm_*","query":{"query_string":{"query":"*","analyze_wildcard":true}},"filter":[]}']}]
  lastfm-histogram |  ['lastfm histogram'] | [{searchSourceJSON: ['{"index":"lastfm_*","query":{"query_string":{"query":"*","analyze_wildcard":true}},"filter":[]}']}]
 lastfm-by-country | ['lastfm by country'] | [{searchSourceJSON: ['{"index":"lastfm_*","query":{"query_string":{"analyze_wildcard":true,"query":"*"}},"filter":[]}']}]
      lastfm-count |      ['lastfm count'] | [{searchSourceJSON: ['{"index":"lastfm_*","query":{"query_string":{"query":"*","analyze_wildcard":true}},"filter":[]}']}]
```


CASSANDRA SUMMIT 2016

Elassandra tools & plugins

Elassandra supports Elasticsearch tools & plugins

- Logstash & Beats
- ElasticHQ (royrusso)
- Elasticsearch-sql (NLPchina)
- JDBC sql4es (Anchormen)



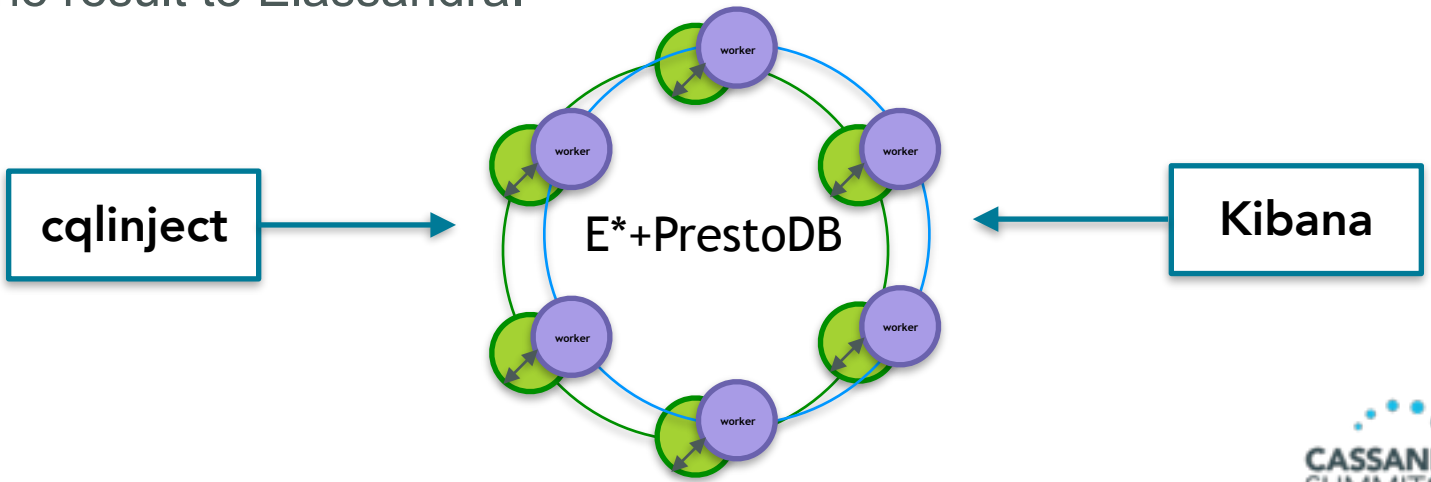
CASSANDRA SUMMIT 2016

Elassandra + Kibana + PrestoDB + CQLInject

Denormalizing our dataset for visualization with kibana

```
cqlinject jdbc "SELECT A.a, B.b FROM A INNER JOIN B on A.c=B.c"
```

1. Execute a JDBC request on prestoDB.
2. From the response metadata, add new columns to the target C* table.
3. Refresh the Elasticsearch mapping from C* table.
4. Write back the result to Elassandra.



Elassandra + Spark

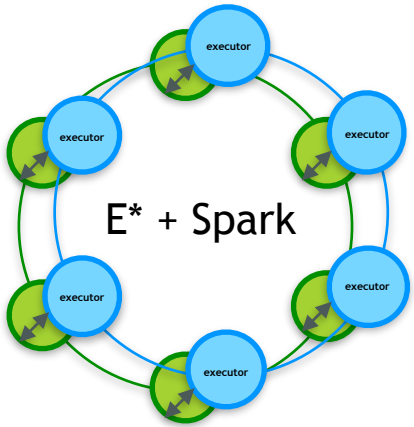
Principle : The Elasticsearch-Hadoop connector creates 1 partition per shard whereas Elassandra has only 1 shard on each node.

Benefits :

- Workers/executors read/write locally on Elassandra nodes.
- Elassandra resharding functionality allows to scale out cassandra +elasticsearch+spark
- The elasticsearch-Spark connector supports pushdown

How :

A slight modification in elasticsearch-hadoop connector to add token_ranges filter from the coordinator routing table to avoid duplicates if nodes have overlapping routing tables.



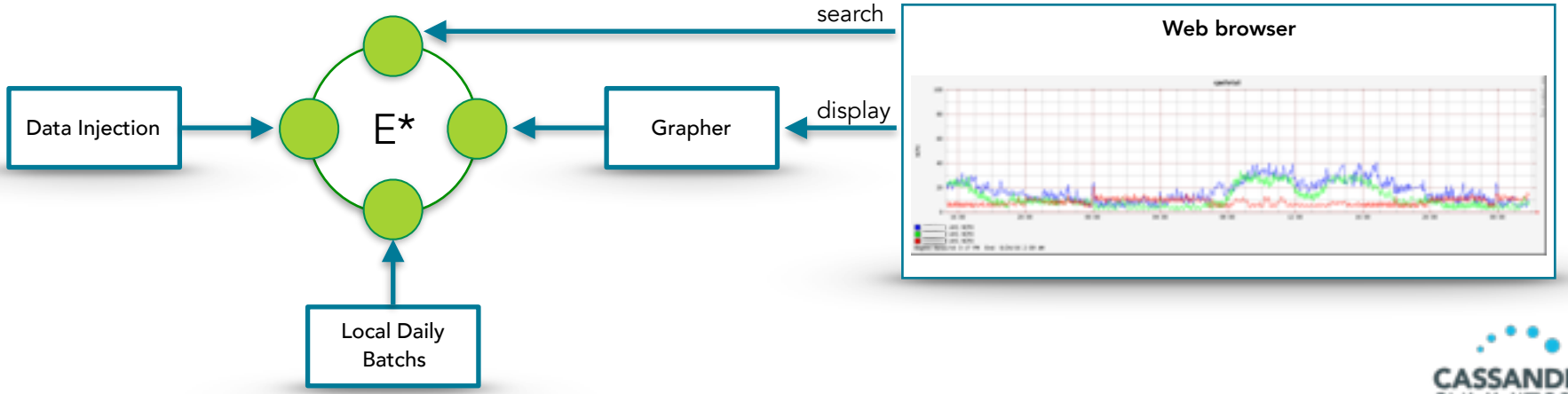
Time Series with Elassandra

Storing large time series in Cassandra

- N tables with different levels of precision and retention
- Daily rollup batches on each node to aggregate local data and compute metadata (min/max/avg/stdev….)
- Automatic purge with default TTL + DateTieredCompactionStrategy

Searching with only index metric names and metadata

- Metadata enrichment by joining other sources of data (ex: datacenters, applications, hardware info….)
- Search with regex on any metadata to display relevant time series



Write throughput

- Write Throughput is the same if your node is not overloaded
- CPU x2 for Elassandra
- (#threads + #classes) X2 for Elassandra

Cassandra = 30k write/s

Elassandra = 30k write/s

2 nodes cluster, **RF=1**, Google Cloud VM n1-highcpu-16 (16 vCPU - 14,4 Go mem)

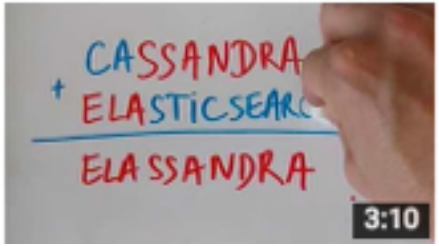
1	Introduction
2	How Elassandra works ?
3	Cool Features
4	Elassandra’s ecosystem
5	Roadmap
6	Q&A

Elassandra Roadmap

Make it a deployed enterprise grade solution:

- Improve the documentation and packaging
- Implement Elasticsearch missing features
- Upgrade to Cassandra 3.0.<lastest> and Elasticsearch 2.<lastest>
- Make it ready for Windows OS
- Provide security features (SSL, LDAP, document and field level security)
- Deliver professional services

More about us …



Elassandra

http://www.elassandra.io
https://github.com/vroyer/elassandra

Vincent Royer
Elassandra Author
vroyer@strapdata.com

Rémi Trouville
Strapdata Co-Founder
rtrouville@strapdata.com

CASSANDRA SUMMIT 2016

1	Introduction
2	How Elassandra works ?
3	Features
4	Use case examples
5	Roadmap
6	Q&A

Thank you