Spark & Cassandra

Jose María Muñoz Rey

DevFest Córdoba
2016

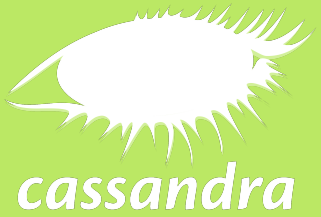
Who am I?

- Software Engineering Student.
- Data analytics and processing @ s|ngular.
- Community enthusiast.

s|ngular

GDG Cáceres

Smart Open Lab



About talk?

- Motivation
- Cassandra
- Spark
- Spark + Cassandra
- Demo
- Conclusions


The need to lead in data and analytics

What do you need?

- Decent analytical query performance.
- Big throughput of reads/writes.
- New ways of do data analytics.
- Strong consistency is not necessary.
- But Big availability!
- Combine more than one data source.


APACHE
Spark
TM

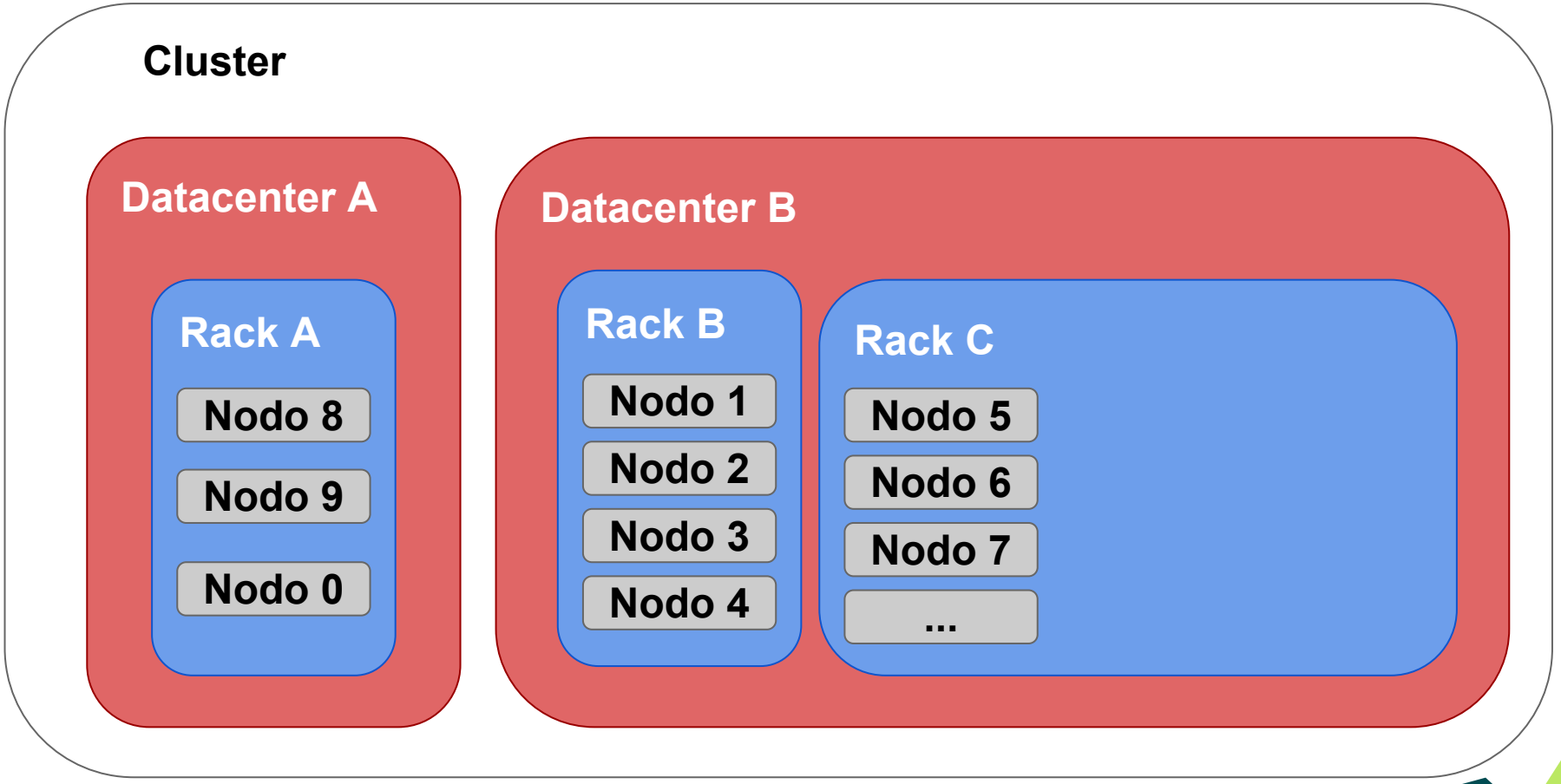
+


cassandra

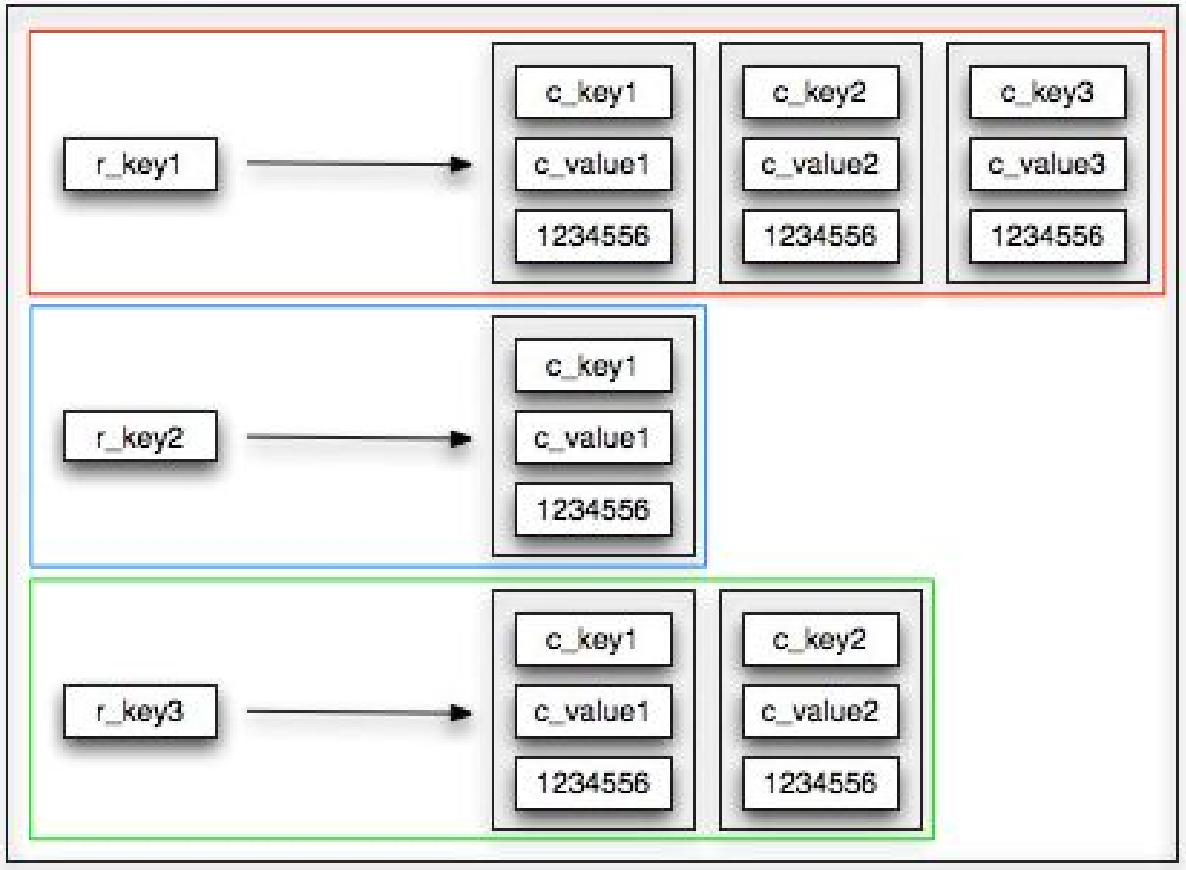
Apache Cassandra

- Google BigTable + Amazon DynamoDB.
- No single point of failure (Masterless).
- “Linear” scalability.
- Support multi-data center.
- CAP Theorem - AP.
- Consistency level on query level.
- Maintained by Datastax.

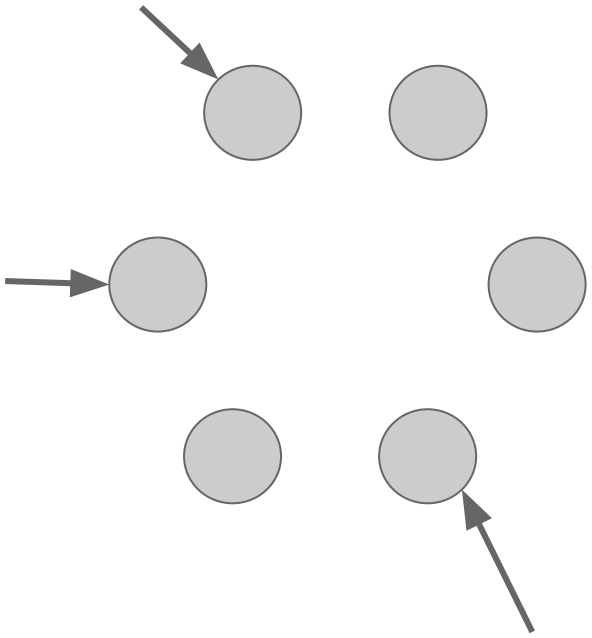
Composition



Data Modelling



About data



- PRIMARY KEY = PARTITION KEY + CLUSTERING KEY
  - PARTITION KEY (hashed)
    - Defines the data locality.
    - Can be compound.
  - CLUSTERING KEY
    - Define the order inside the partition
- Any node can coordinate a query.
- The driver usually have the partition function.

Be careful ***

CQL

Basically like standard SQL but significantly limited.

- Restrictions to avoid full-table queries. ***
- We must know how the data is stored.

But it has some nice functionalities.

- C**omplex data types:** JSON, Collections (set, map…) and user defined.
- **User defined functions*** in Java/JavaScript.
- **Secondary index** (columns, any value, even collections)***.
- **Materialized views**.

Querying data

((id, day), time), measure

```
SELECT *
FROM iot
WHERE id = "159323"
  AND day = "2016-12-03"
  AND “14:00:00.000” < time
  AND time < “20:00:00.000”;
```

Amount of data queried:

- Or full partition key (=, IN).
- Or Indexes (limited).

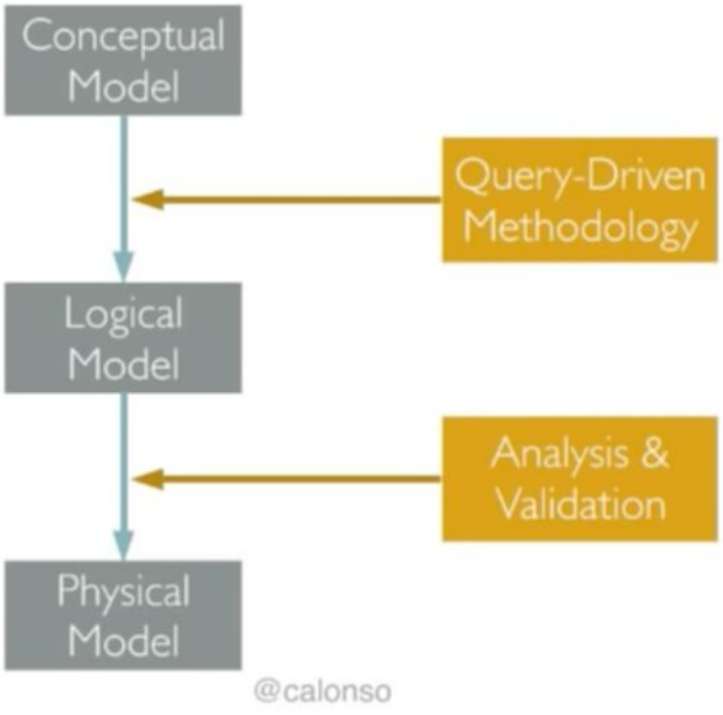
Sorting:

- Remember the clustering column? Use it! (<,>,>=,<=)

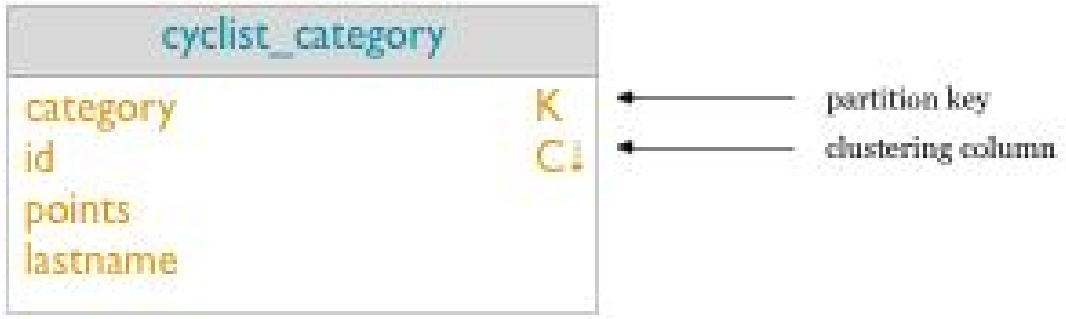
Other:

- JSON, CSV output/input support.
- No JOIN, GROUP BY.

Query-Driven Methodology



- Start from Entity-relationship diagram.
- Query-Driven Methodology.
- Logical model (Chebotko).



- Analysis & Validations.
- Physical implementation.
- **The Kashlev Data Modeler.**

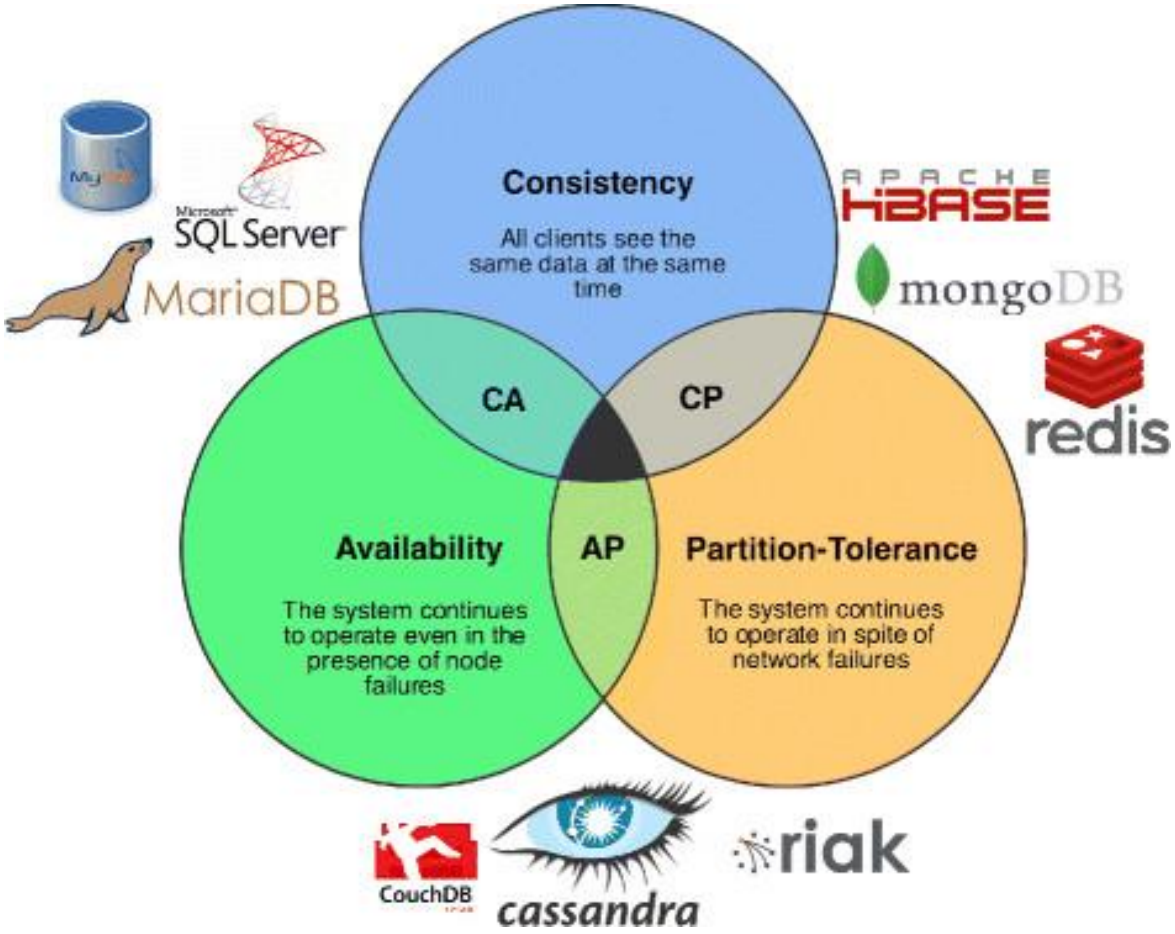
Replication

- **Replication Factor defined on keyspace:**

```
CREATE KEYSPACE my_database
WITH REPLICATION = {
    “Class” : ”networkTopologyStrategy”,
    “Dc-east” : 2, “dc-west” : 3
    };
```

- Multiple class styles to support multiple schemas.
- **Data close to the users**.
- **Functionality segregation** (DC for users, DC for analytics).

CAP Theorem



Consistency

Availability

Consistency VS Availability

- **Real State** vs **Velocity**

On **Keyspace** Creation

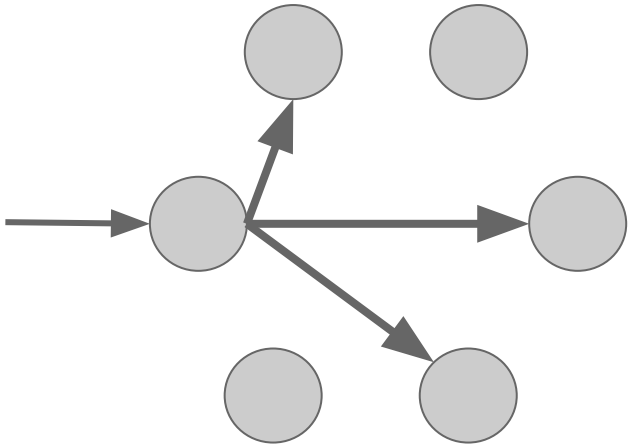
- Replication factor

On **Query**

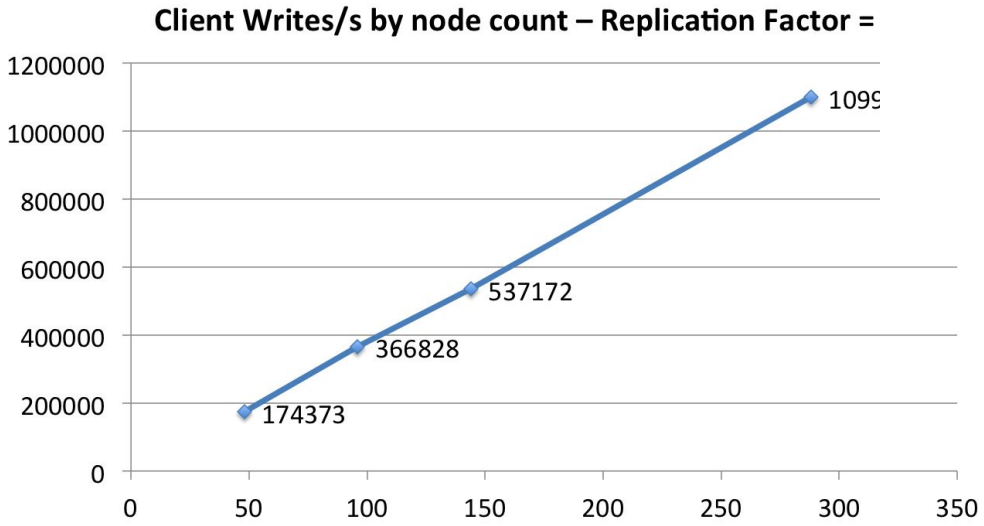
- ConsistencyLevel - ONE, QUORUM “(Total/2)+1”, ALL

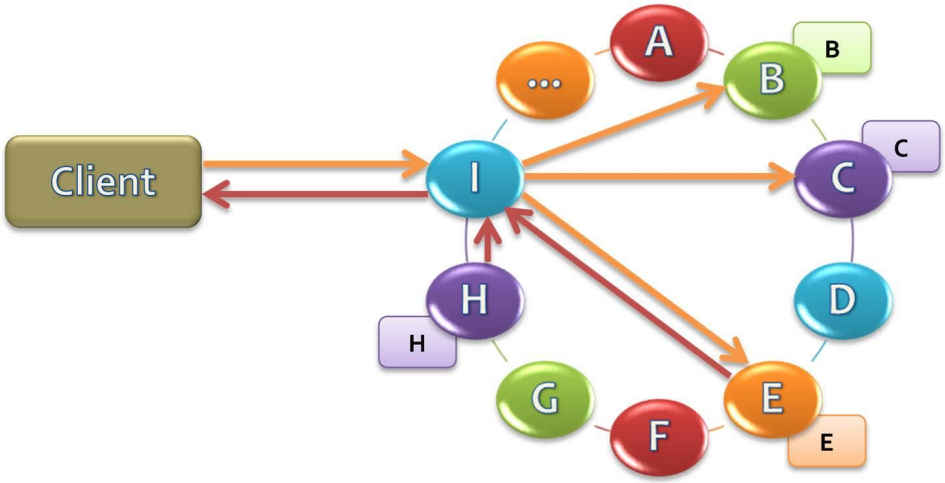
Lightweight Transactions

- Compare-and-set operations



Scale-Up Linearity





NETFLIX

Widely used

Company	Nodes	Data Volume	Operations
Apple	75.000	10 PB	
Netflix	2.500	420 TB	1 billion/day
Easou	270	300 TB	800 millions/day
eBay	100	250 TB	500 millions/day

NETFLIX

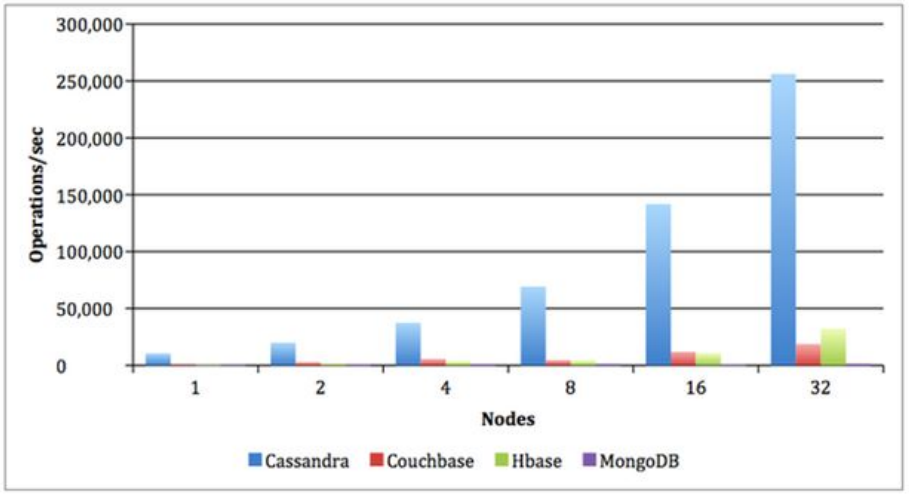
easou 宜搜

ebay

Spark

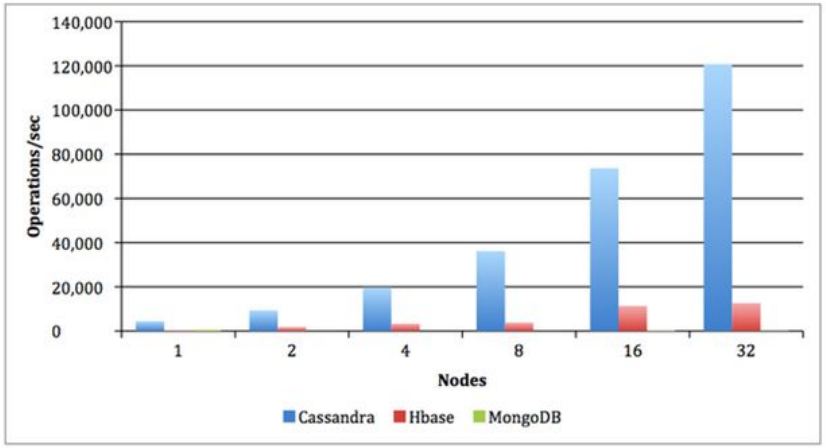
cassandra

Read-Modify-Write Workload

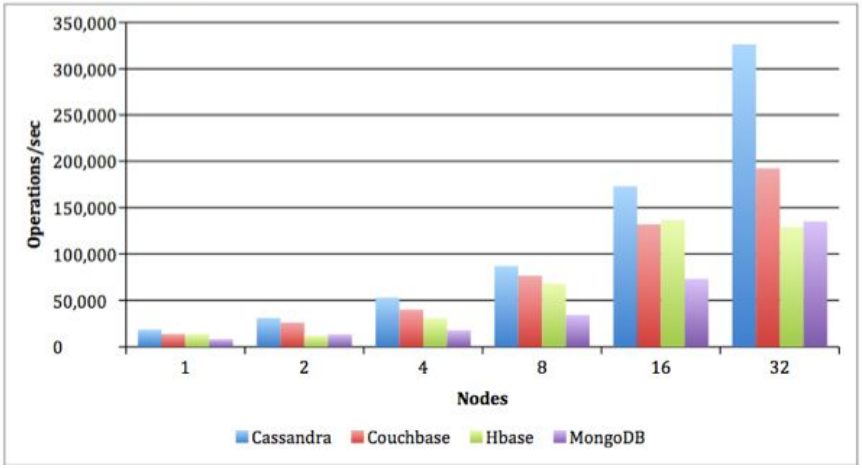


Mixed Operational and Analytical Workload

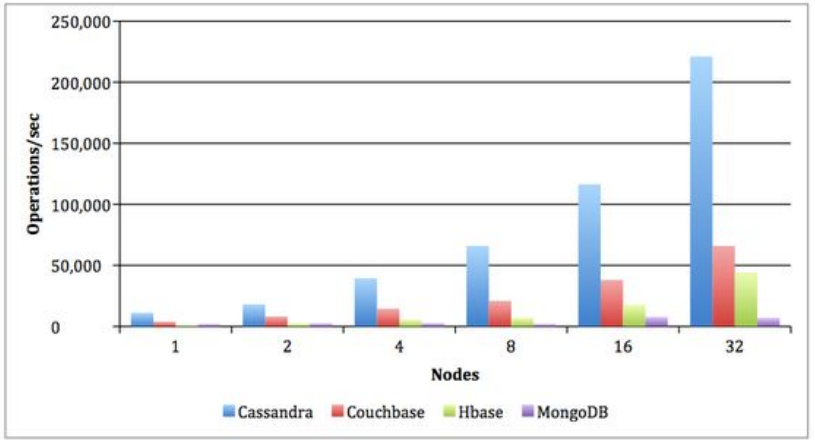
Note that Couchbase was eliminated from this test because it does not support scan operations (producing the error: "Range scan is not supported").



Load Process



Read-mostly Workload



Conclusions

The Bads

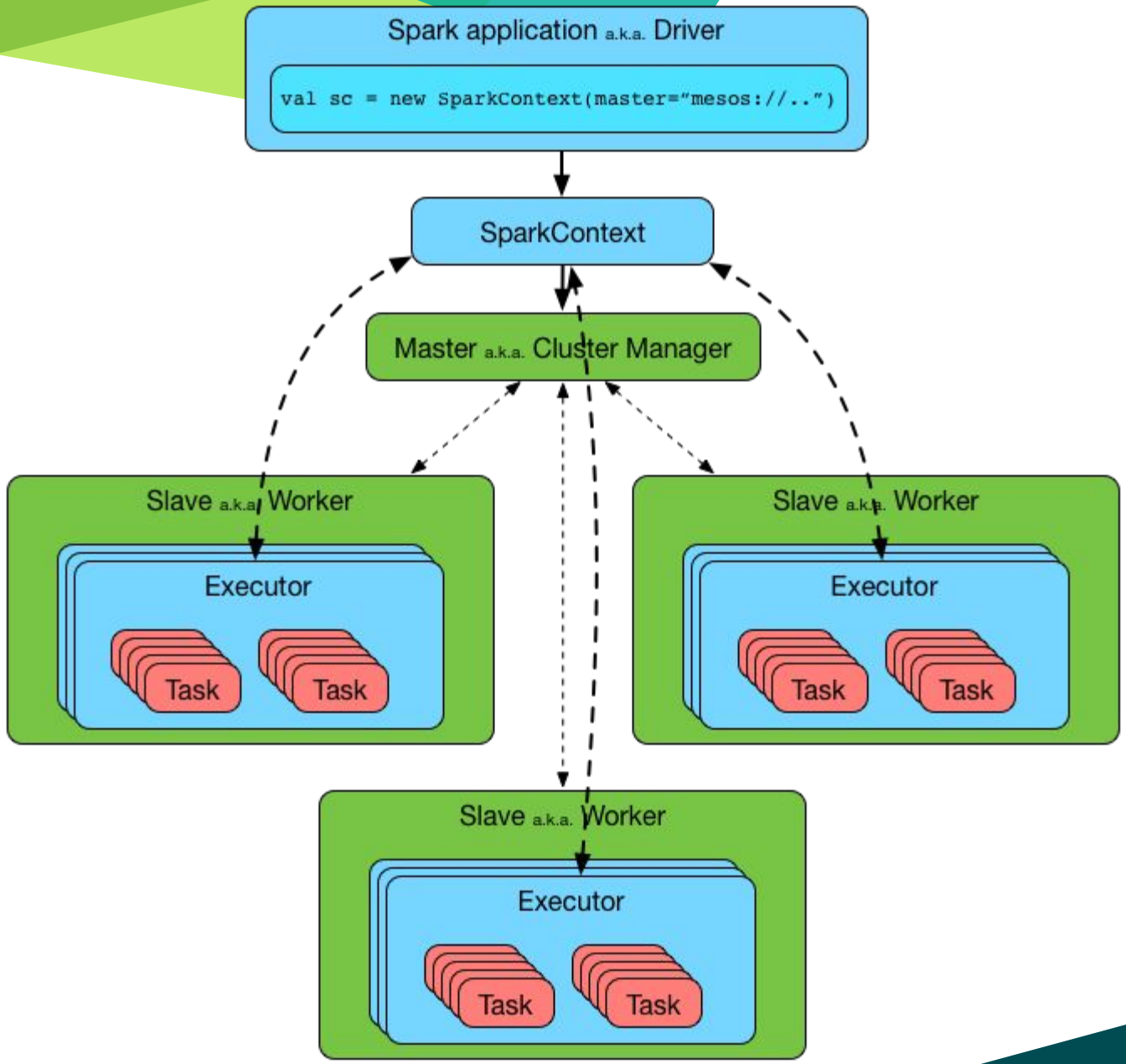
- Remember Query Driven Methodology, be careful.
- Queries have low flexibility.
- Use secondary Indexes usually are a bad practice.

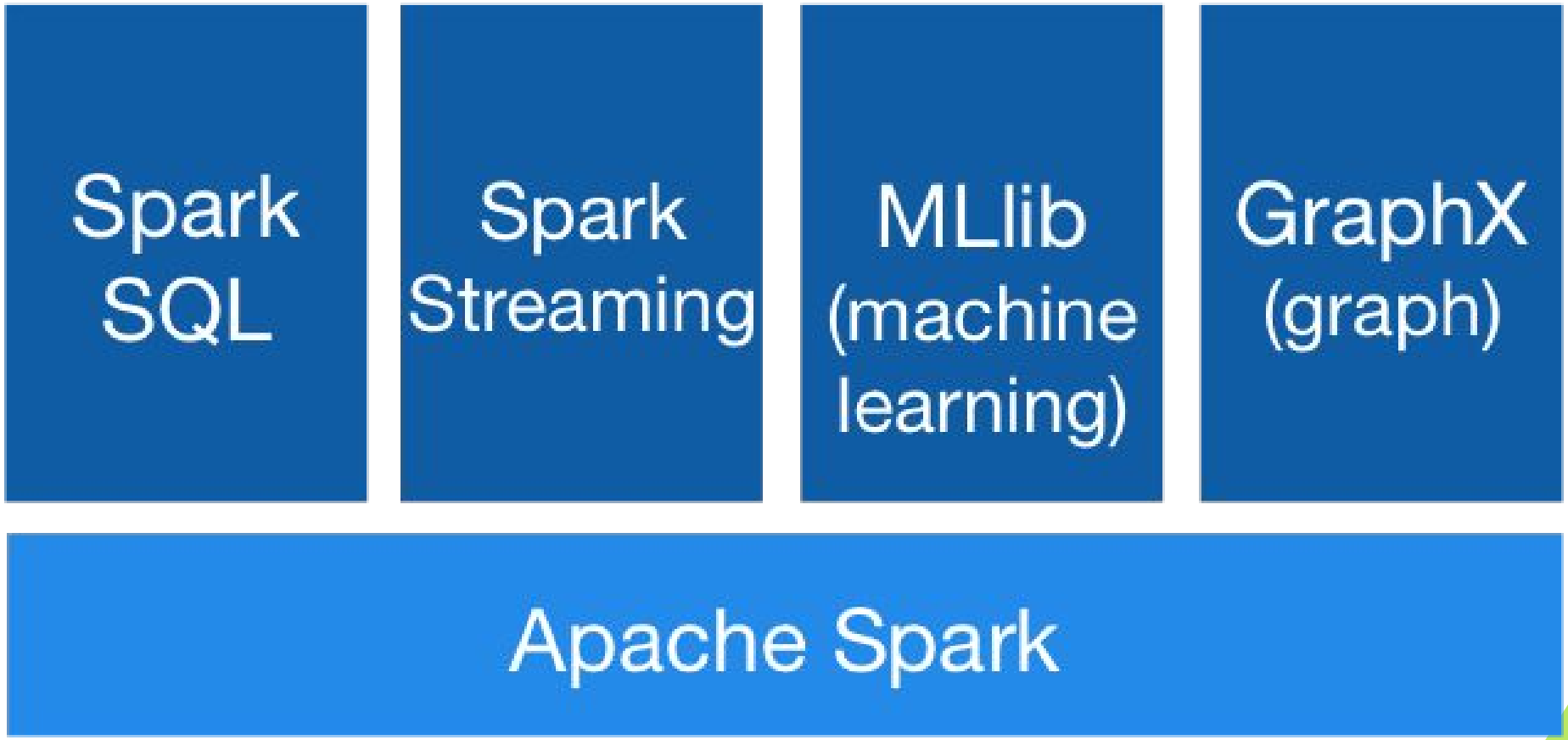
The Goods

- Linear scaling.
- Storage is cheaper than processing.

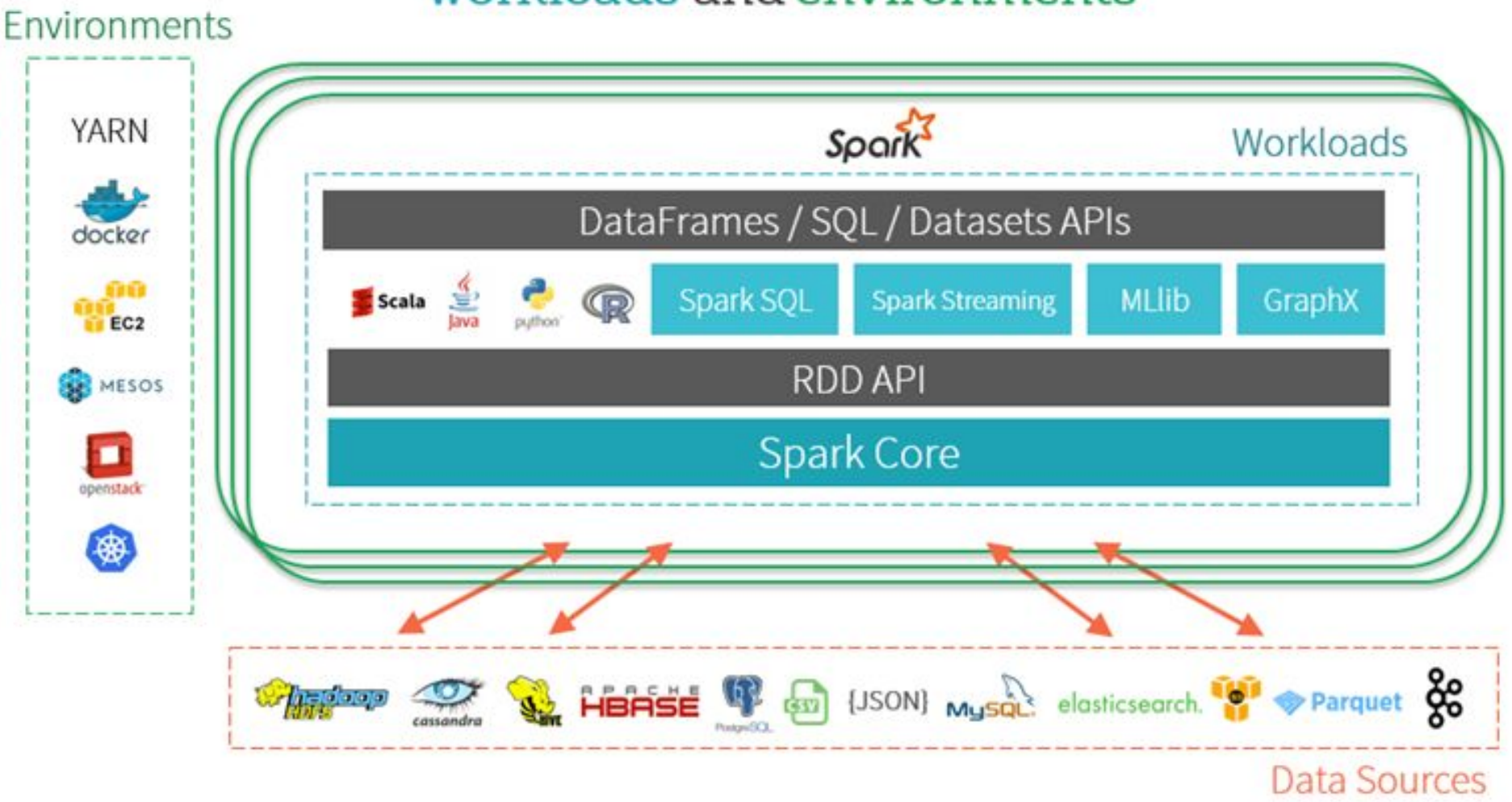
Apache Spark

- Framework for distributed data processing.
- Fault tolerant.
- Faster than Hadoop (10x-100x).
- Many integrations with frameworks, libraries, databases...
- Batches and streaming analytics.
- Machine Learning with distributed processing.
- Easy to use.

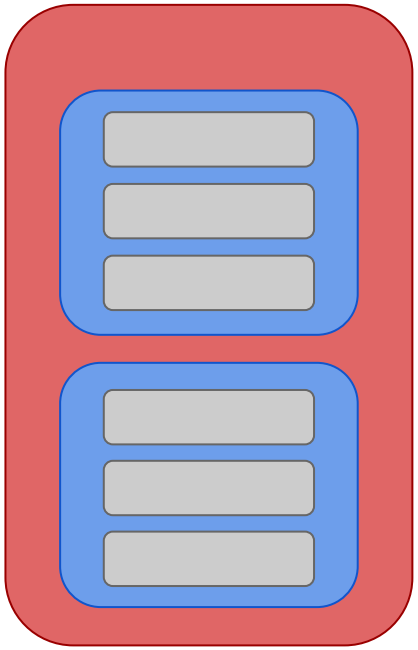

Spark application a.k.a. Driver
val sc = new SparkContext(master="mesos://..")
SparkContext
Master a.k.a. Cluster Manager
Slave a.k.a. Worker
Executor
Task
Task
Slave a.k.a. Worker
Executor
Task
Task
Slave a.k.a. Worker
Executor
Task
Task


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

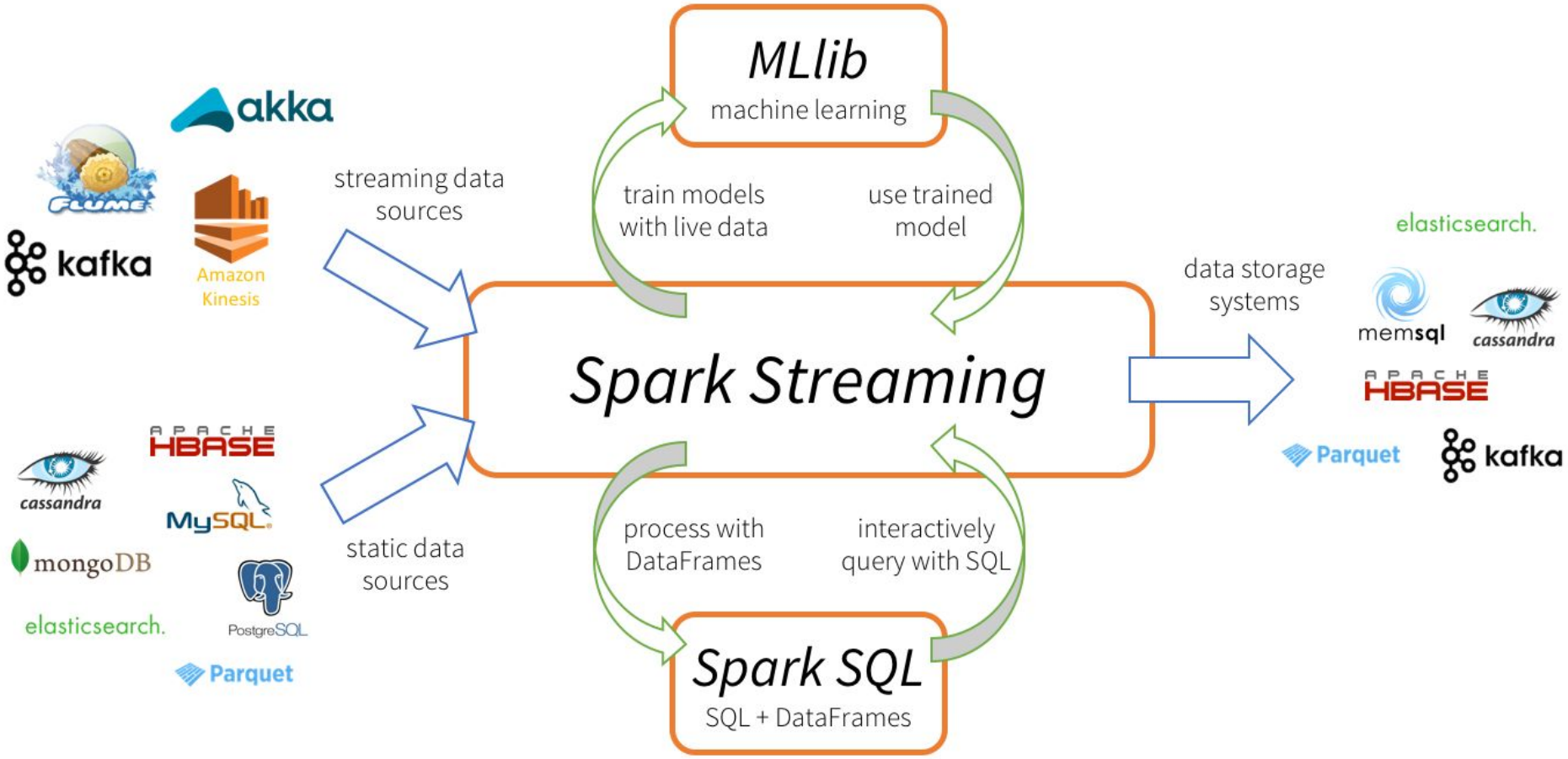
Goal: unified engine across data sources, workloads and environments



RDD, DataFrame & Dataset



- RDD: Resilient Distributed Dataset.
- DataFrame = RDD + Schema (SQL!).
- Dataset = DataFrame + Language.
  - Type-safe.
  - Object-oriented programming interface.


MLlib
machine learning
train models with live data
use trained model
akka
Flume
kafka
Amazon Kinesis
streaming data sources
Spark Streaming
data storage systems
elasticsearch.
memsql
cassandra
APACHE HBASE
Parquet
kafka
cassandra
APACHE HBASE
MySQL
mongoDB
elasticsearch.
PostgreSQL
Parquet
static data sources
process with DataFrames
interactively query with SQL
Spark SQL
SQL + DataFrames

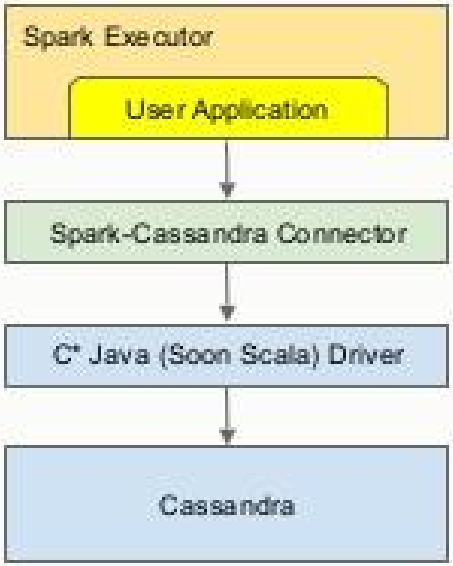
SQL or DSL

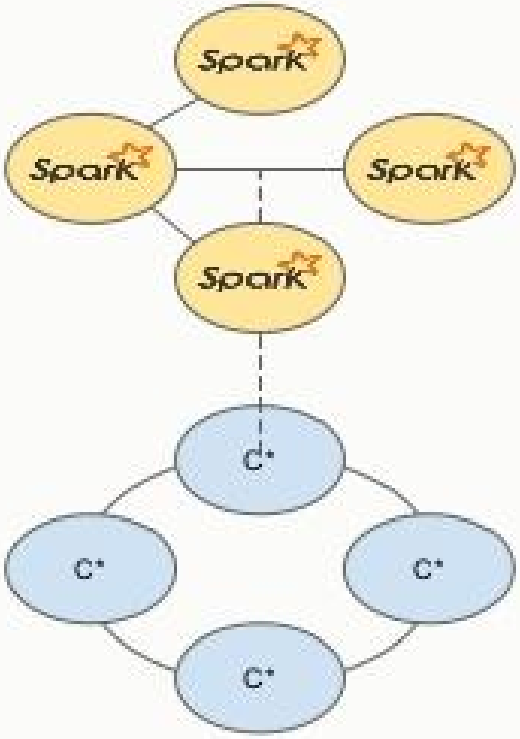
```
val total = sqlContext.sql(
    "SELECT count(*) FROM subvenciones"
);
total.show();

val group_by_spark = muestra
    .groupBy("organo_gestor")
    .sum("importe")
    .orderBy("sum(importe)")
group_by_spark.collect.forEach(println)
```

Spark & Cassandra

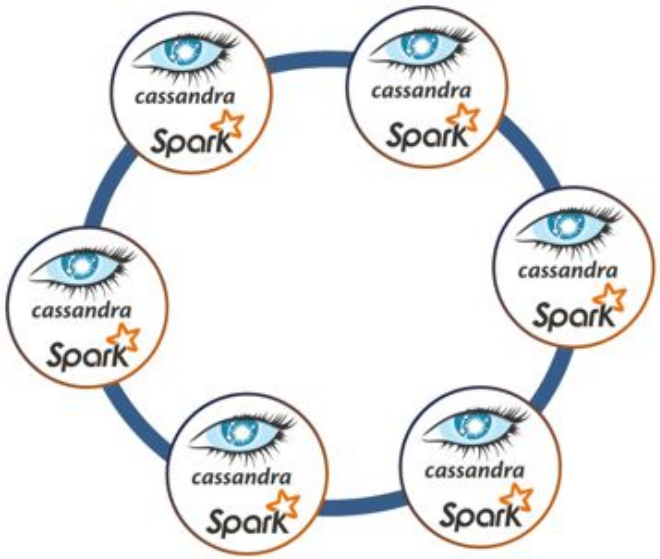
- Push filters to the server.
- Operations between tables (JOIN, UNION…).
- Know where the data lives (Data Locality).
- Special structures too, like “time-series”.

Spark Cassandra Connector

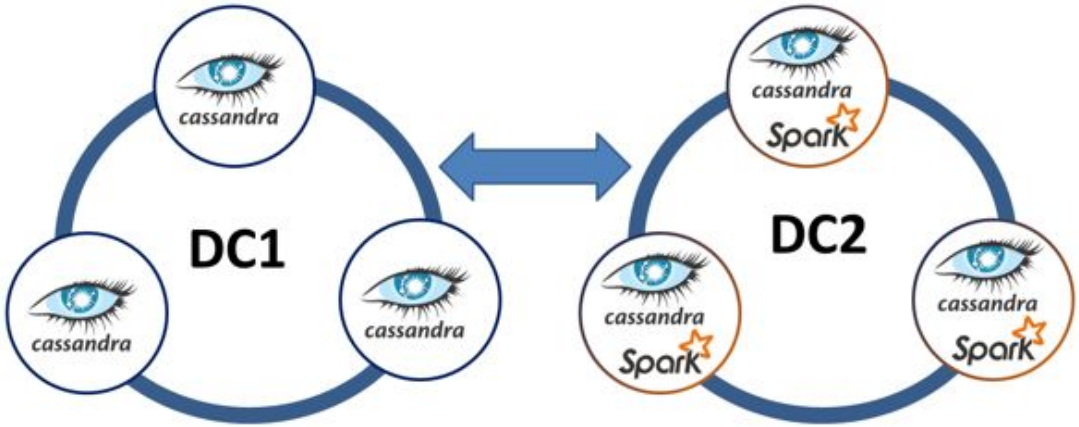




https://github.com/datastax/spark-cassandra-connector



Single DC V.S. Multi DC



One of the best things - Data Locality

- Any node can coordinate a read/write.
- The connector know where the data lives.
  - Any Spark node reads from his Cassandra node.
- Less use of **CPU** and **RAM** for each node.

I remember nothing!!!
Where can I learn about it?

- Cassandra:

https://academy.datastax.com/courses/

- Spark:

https://databricks.com/training

https://community.cloud.databricks.com/

- Cassandra + Spark:

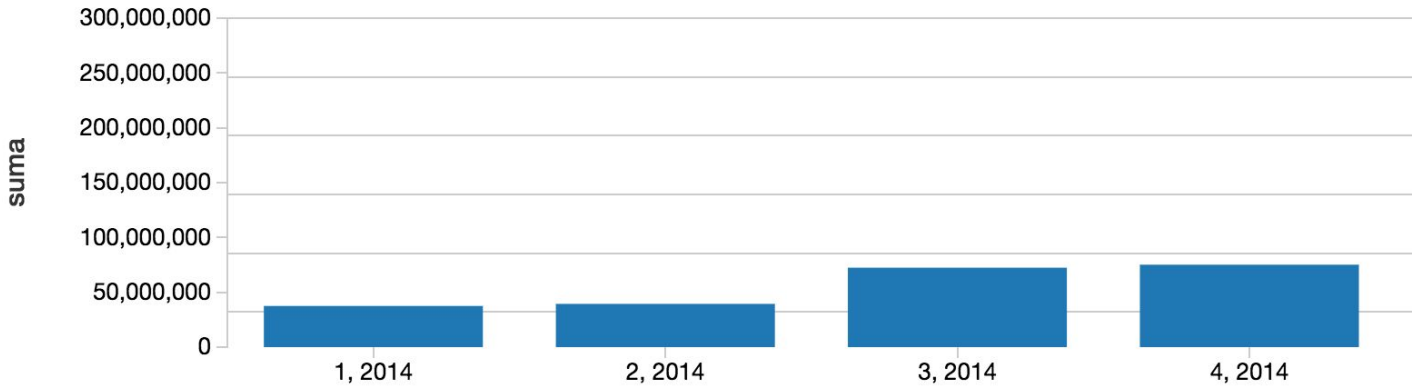
https://academy.datastax.com/courses/getting-started-apache-spark

Use the notebooks!

Importe por meses

```
%sql
Select sum(importe) AS suma, year(to_date(fecha_concesion)) as year,
month(to_date(fecha_concesion)) as month
from subvenciones
group by year(to_date(fecha_concesion)), month(to_date(fecha_concesion))
Order by year(to_date(fecha_concesion)), month(to_date(fecha_concesion));
```

(1) Spark Jobs



Spark + Cassandra

- Complementary systems.
- Very scalable and fast.
- Complete Analytical possibilities (Window functions, SQL, ML …).
- Do you have another database?
  - Do a JOIN with a Cassandra table!

Thanks!

Any question?

You can find me:

Koletzilla

Koletzilla

josemmunoz

Sources

- Slide theme: Escalus
  http://www.slidescarnival.com/es/plantilla-gratis-de-presentacion-escalus/967
- “The need to lead in data and analytics” :
  http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-need-to-lead-in-data-and-analytics
- Left image in pag 13: https://www.youtube.com/watch?v=jxwtmMTMrJA
- CAP image:
  https://www.researchgate.net/figure/282519669_fig1_Figure-1-CAP-theorem-with-databases-that-choose-CA-CP-and-AP
- Netflix Benchmark: http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html
- Pag 18, extracted from: http://cassandra.apache.org/
- Pag 19, benchmarks:
  https://academy.datastax.com/planet-cassandra/NoSQL-performance-benchmarks

Sources

- Pag 22: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-architecture.html
- Pag 24: https://docs.databricks.com/spark/latest/gentle-introduction/for-data-engineers.html
- Pag 26: https://www.datanami.com/2015/11/30/spark-streaming-what-is-it-and-whos-using-it/
- Pag 30: https://www.instaclustr.com/blog/2016/01/07/multi-data-center-apache-spark-and-apache-cassandra-benchmark/