Big Data: motivation and overview

Current Trends in Artificial Intelligence

Guillaume Levasseur

IRIDIA,
Université libre de Bruxelles

Friday, 2nd March 2018

(Short) Formal definition of machine learning

Learning requires (more) data!

More data? Big Data!

Application

(Short) Formal definition of machine learning

$$f(\mathbb{X}, \vec{\omega}) = \vec{y} \tag{1}$$

- f Model
- $\mathbb{X}$ Matrix of the features (input data)
- $\vec{y}$ Output vector
  - $\vec{y}$ is continuous = regression
  - $\vec{y}$ is discrete = classification (supervised) or clustering (unsupervised)
  - $\vec{y}$ is symbolic = decision problem
- $\vec{\omega}$ Weights of the model f
  - Determining $\vec{\omega}$ = train the model, learn

Learning requires (more) data!

Learning Determining the weights of the model using a training dataset.

Training dataset Set of features (and output values) representative of the problem, as **big** as possible.

Points	**Attributes**		
0	$x_1[0]$	$x_2[0]$	$x_3[0]$
1	$x_1[1]$	$x_2[1]$	$x_3[1]$
2	$x_1[2]$	$x_2[2]$	$x_3[2]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- **Problem** How to store this data?

Spreadsheet storage

	A	B	C	D	E	F
1	week	day	time	value		
2	7	6	0.03999999910593	272		
3	7	6	0.090000003576279	260		
4	7	6	0.140000000596046	259		
5	7	6	0.189999997615814	261		
6	7	6	0.239999994635582	270		
7	7	6	0.28999999165535	270		
8	7	6	0.340000003576279	270		
9	7	6	0.389999985694885	260		
10	7	6	0.439999997615814	261		
11	7	6	0.490000009536743	266		
12	7	6	0.540000021457672	510		
13	7	6	0.589999973773956	259		
14	7	6	1.03999996185303	257		
15	7	6	1.0900000333786	265		
16	7	6	1.13999998569489	255		
17	7	6	1.19000005722046	268		
18	7	6	1.24000000953674	266		
19	7	6	1.28999996185303	258		
20	7	6	1.3400000333786	270		
21	7	6	1.38999998569489	261		
22	7	6	1.44000005722046	255		
23	7	6	1.49000000953674	270		
24	7	6	1.53999996185303	265		
25	7	6	1.5900000333786	263		
26	7	6	2.03999996185303	266		

Figure: Time series in a spreadsheet

- Limited to 1 million rows per file
- Cheesy
- Limited access to 1 user at a time, poor availability
- Sequential reading

$\Rightarrow$ Not suitable!

Relational database management system

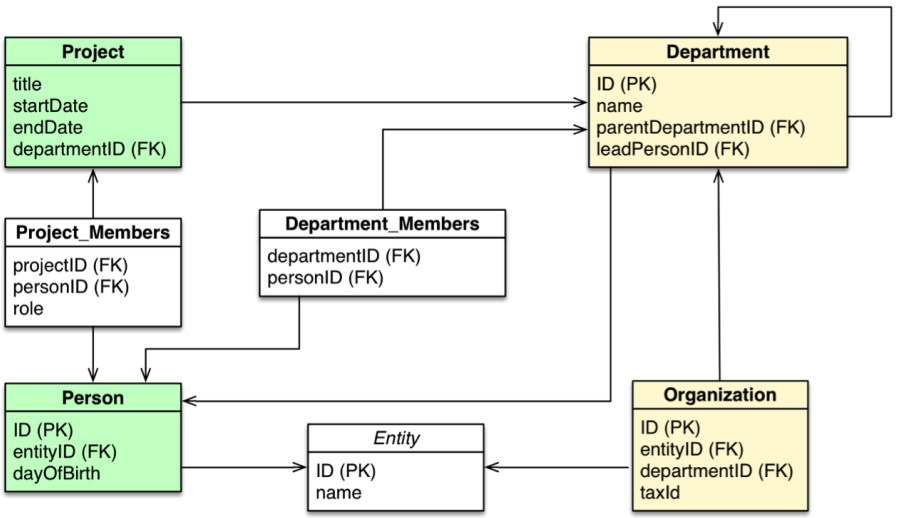


Figure: Relational model (`neo4j.com`)

- Greater capacity ($> 1\,\mathrm{M}$ rows)
- Guaranteed consistency through ACID properties (atomicity, consistency, isolation, durability)
- Relations ("shortcuts") between tables
- Structured query language (SQL) to retrieve data
- Better availability through serving systems (MySQL, SQLServer, PostgreSQL)

Relational database management system

- RDBMS limitations:
  - Fixed data schema: defined *a priori*, no missing data allowed
  - Server bound
- Problems:
  - What if there is missing data?
  - What if the data varies in size, format, structure? How to combine heterogeneous data sources?
  - What if the workload become too heavy for the server (too many requests for the CPU, to much data for the HDD)?

Relational database management system

- RDBMS limitations:
  - Fixed data schema: defined *a priori*, no missing data allowed
  - Server bound
- Problems:
  - What if there is missing data?
    - Use custom symbols like "NaN".
  - What if the data varies in size, format, structure? How to combine heterogeneous data sources?
    - Create separated tables and try to link them.
  - What if the workload become too heavy for the server (too many requests for the CPU, to much data for the HDD)?
    - KIWI (kill it whit iron) = hardware upgrade.

Relational database management system

- RDBMS limitations:
  - Fixed data schema: defined *a priori*, no missing data allowed
  - Server bound
- Problems:
  - What if there is missing data?
    - Use custom symbols like "NaN".
    - **Use flexible schema.**
  - What if the data varies in size, format, structure? How to combine heterogeneous data sources?
    - Create separated tables and try to link them.
    - **Use flexible schema.**
  - What if the workload become too heavy for the server (too many requests for the CPU, to much data for the HDD)?
    - KIWI (kill it whit iron) = hardware upgrade.
    - **Distribute the load on multiple machines.**

More data? Big Data!

Remember, back in 2006...

More data? Big Data!

Remember, back in 2006...

- More data, more variety, more requests.
- Google develops Bigtable, which later inspired

More data? Big Data!

The 3 Vs of Big Data:

- **Variety** Data of many different types are to be stored...
- **Velocity** it's coming fast...
- **Volume** and there is a **lot** of it!

Scalability

Ability to adapt when the order of magnitude changes.

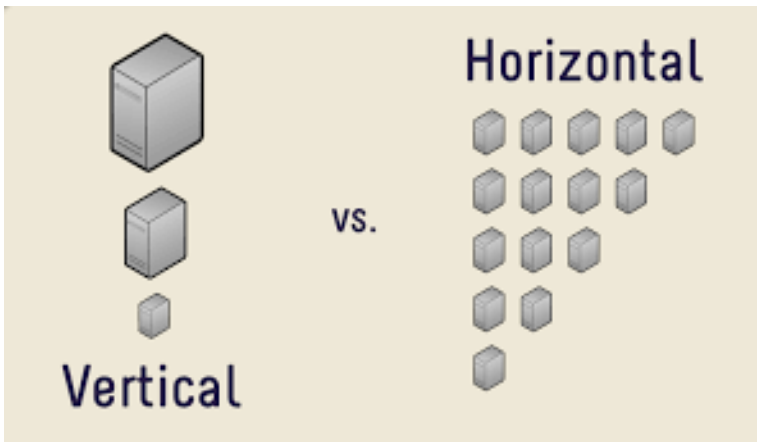


Figure: Vertical and horizontal scalability (`premaseem.files.wordpress.com`).

Consistency: from ACID to BASE

- Problem ACID does not support partitioning across multiple nodes.
- BASE: basically available, soft state, eventually consistent

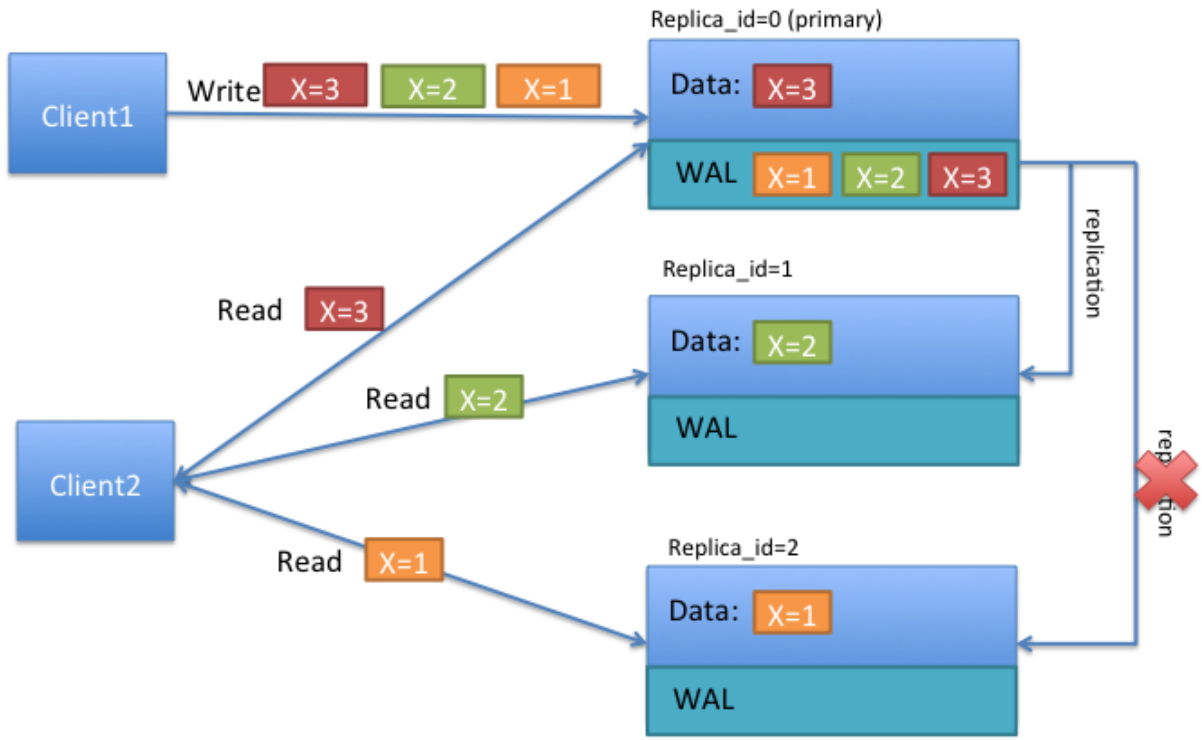


Figure: Consistency issue for distributed database systems (`hbase.apache.org`).

CAP theorem

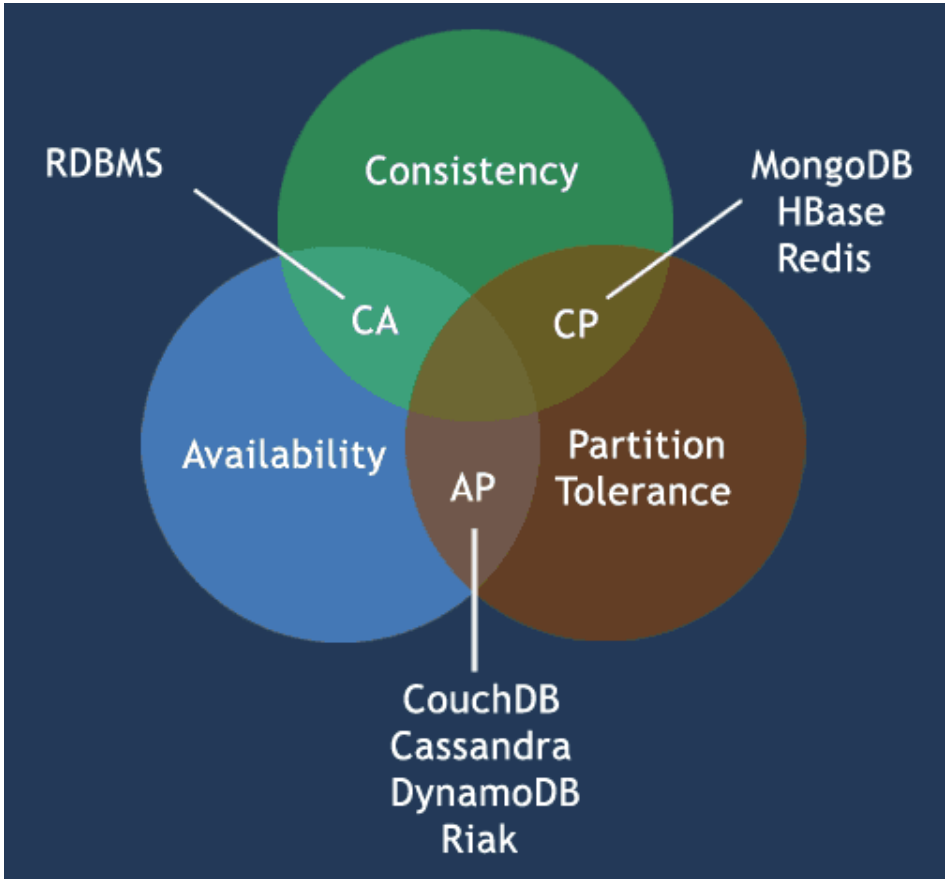


Figure: The CAP theorem: any database system has to drop either the consistency, the availability or the partition tolerance (`w3resource.com`).

NoSQL technologies

- NoSQL = not only SQL, NewSQL = distributed relational database
- Shared-nothing (distributed) architecture ⇒ CP, AP
- Supported operations: CRUD (create, read, update, delete) and scan.
- Classification:

key-value	unique ID	value	

columnar		Family 0		Family 1
	unique ID	column 0	column 1	column 2

document	unique ID	{attr0: value0, attr1: value1}	
graph	unique ID	arcs	vertices
search engines	inverted ID	plain text or document	

NoSQL technologies

More than 200 different systems!

Figure: Logos of some NoSQL database systems.

Hadoop

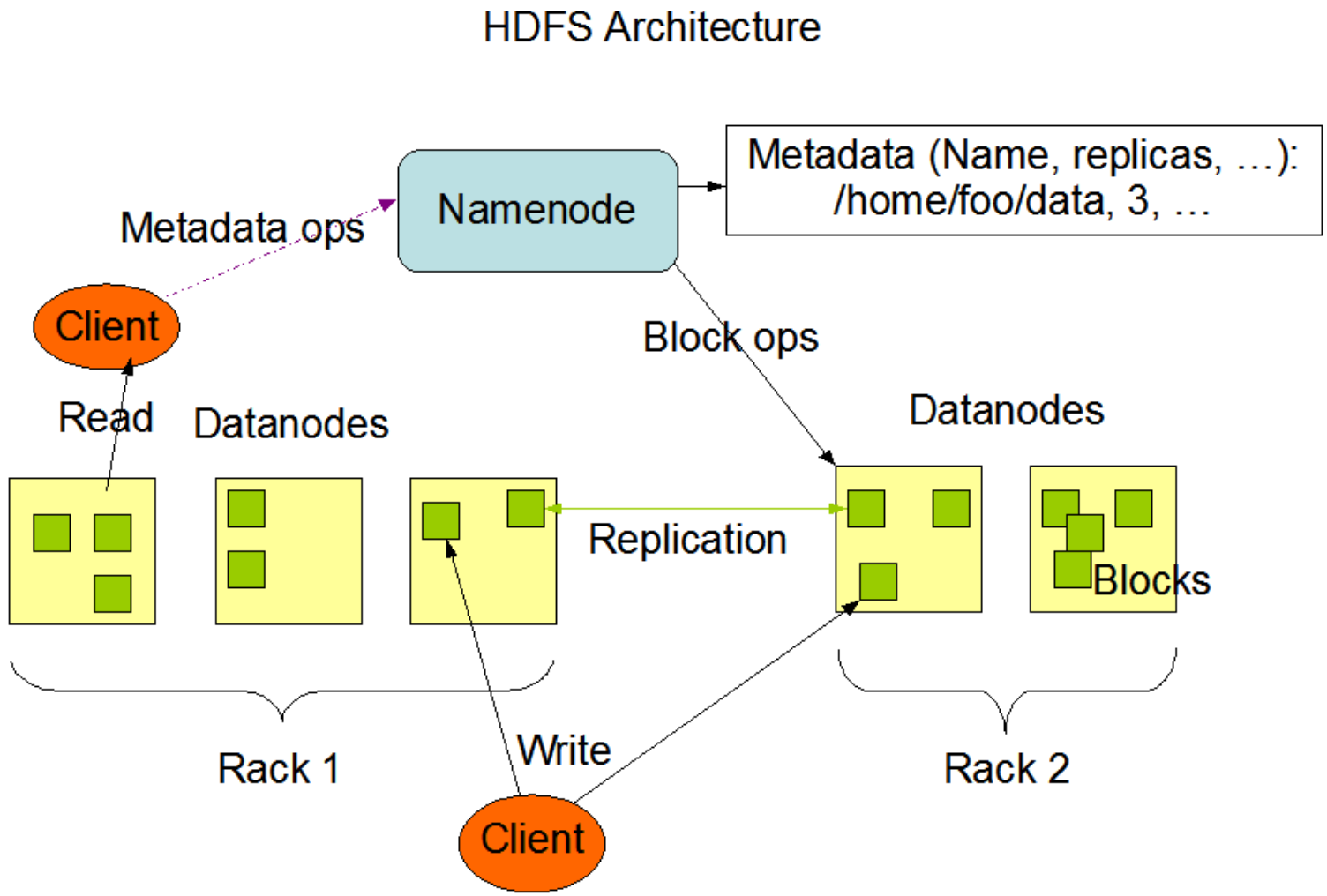


Figure: Architecture of the Hadoop distributed file system, the *Namenode* acts as master of the infrastructure (`hadoop.apache.org`).

On top of Hadoop: HBase

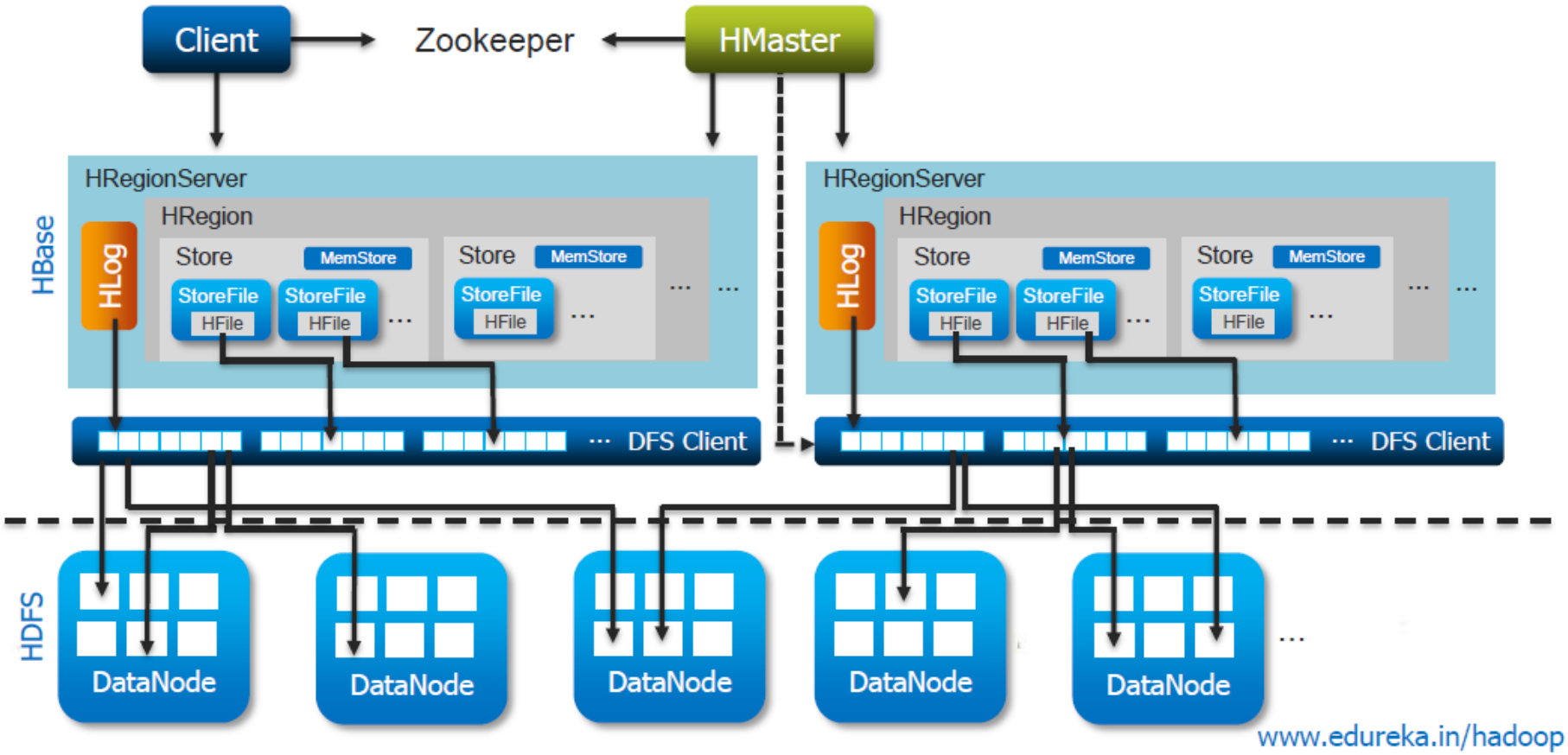


Figure: HBase architecture (`eureka.co`).

Cassandra

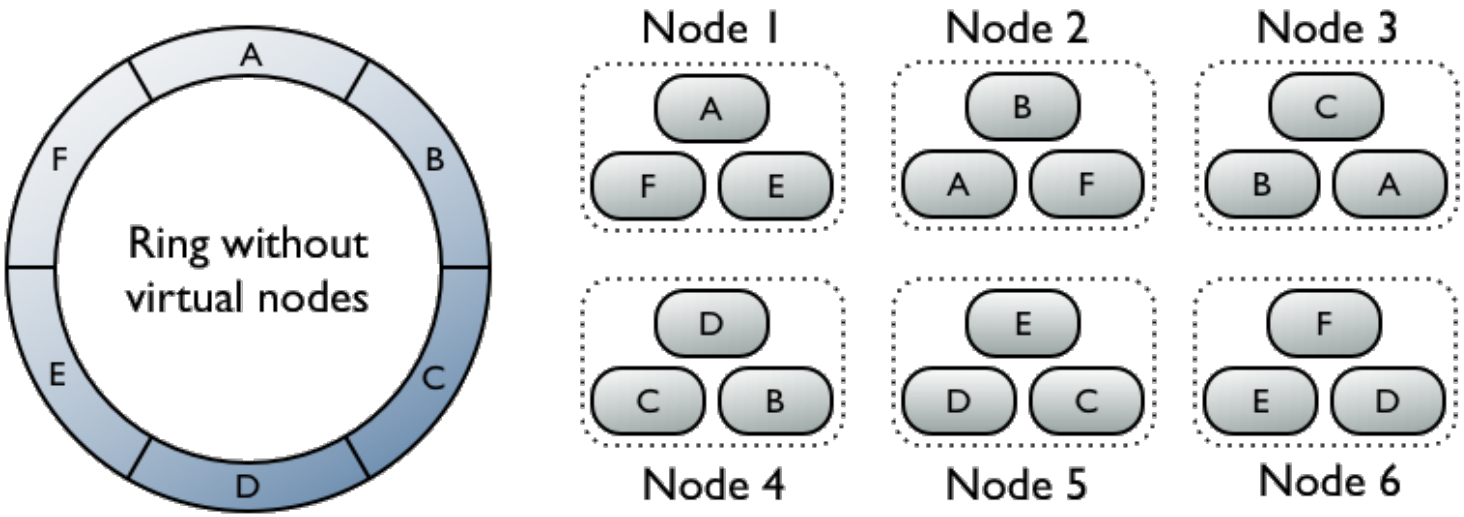


Figure: Cassandra's ring strategy for splitting (`docs.datastax.com`).

Cassandra

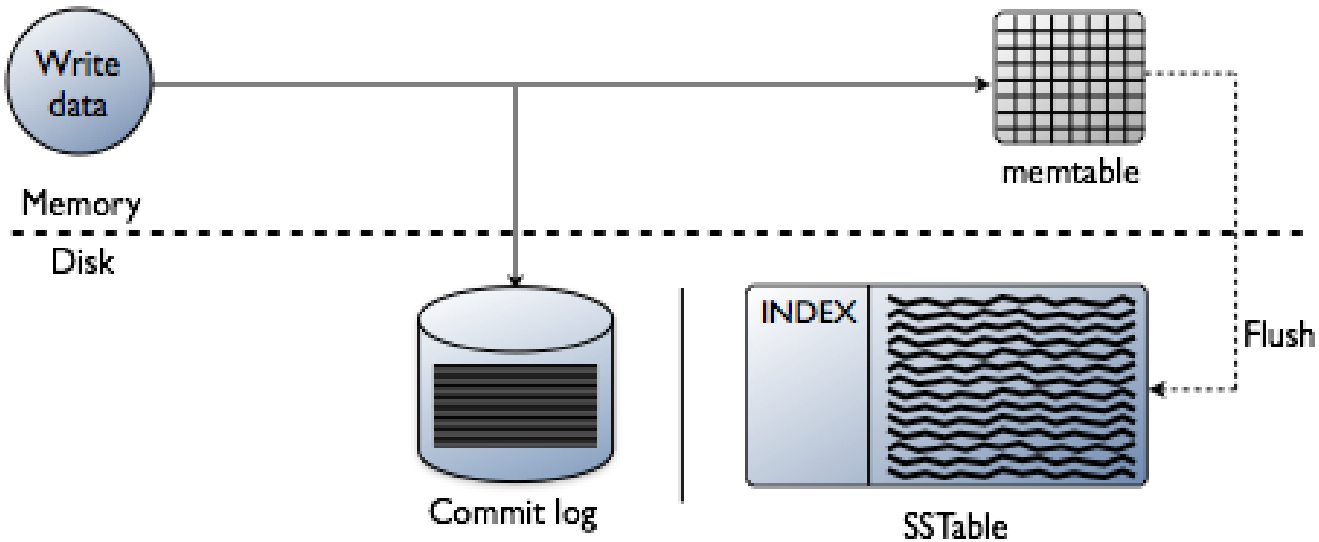


Figure: Cassandra's writing process (`docs.datastax.com`).

- No master: nodes use gossiping to exchange metadata about the current state
- Compaction: too many *SSTables* $\Rightarrow$ merge in one bigger *SSTable*.

Cassandra

```
Connected to IRIDIA at node001:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh> USE test;
cassandra@cqlsh:test> SELECT * FROM temp WHERE year=2018 AND week=8 AND day>3 ALLOW FILTERING;

 id                    | year | week | day | time | value
-----------------------+------+------+-----+------+-------
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.04 | 21.78
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.09 | 21.86
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.14 | 21.92
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.19 | 21.94
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.24 | 21.93
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.29 | 21.88
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.34 |  21.8
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.39 | 21.72
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.44 | 21.66
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.49 | 21.61
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.54 | 21.61
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 0.59 |  21.6
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.04 | 21.62
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.09 | 21.71
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.14 | 21.84
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.19 | 21.98
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.24 | 22.09
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.29 | 22.12
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.34 | 22.06
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.39 | 21.95
 B827EBA6DBA1::bedroom | 2018 |    8 |   4 | 1.44 | 21.82
```

Figure: Result of a SELECT query in CQL.

Elasticsearch

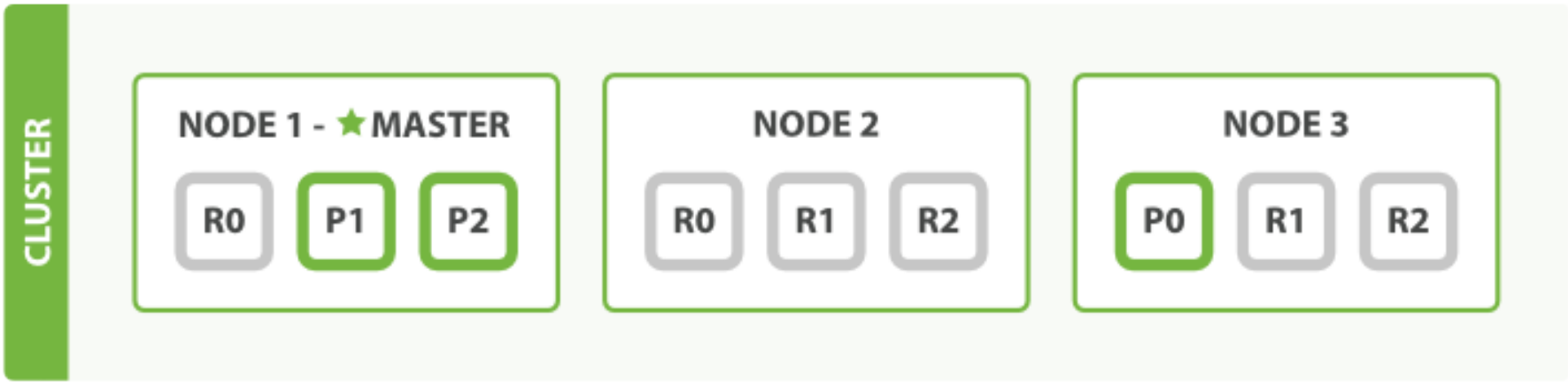


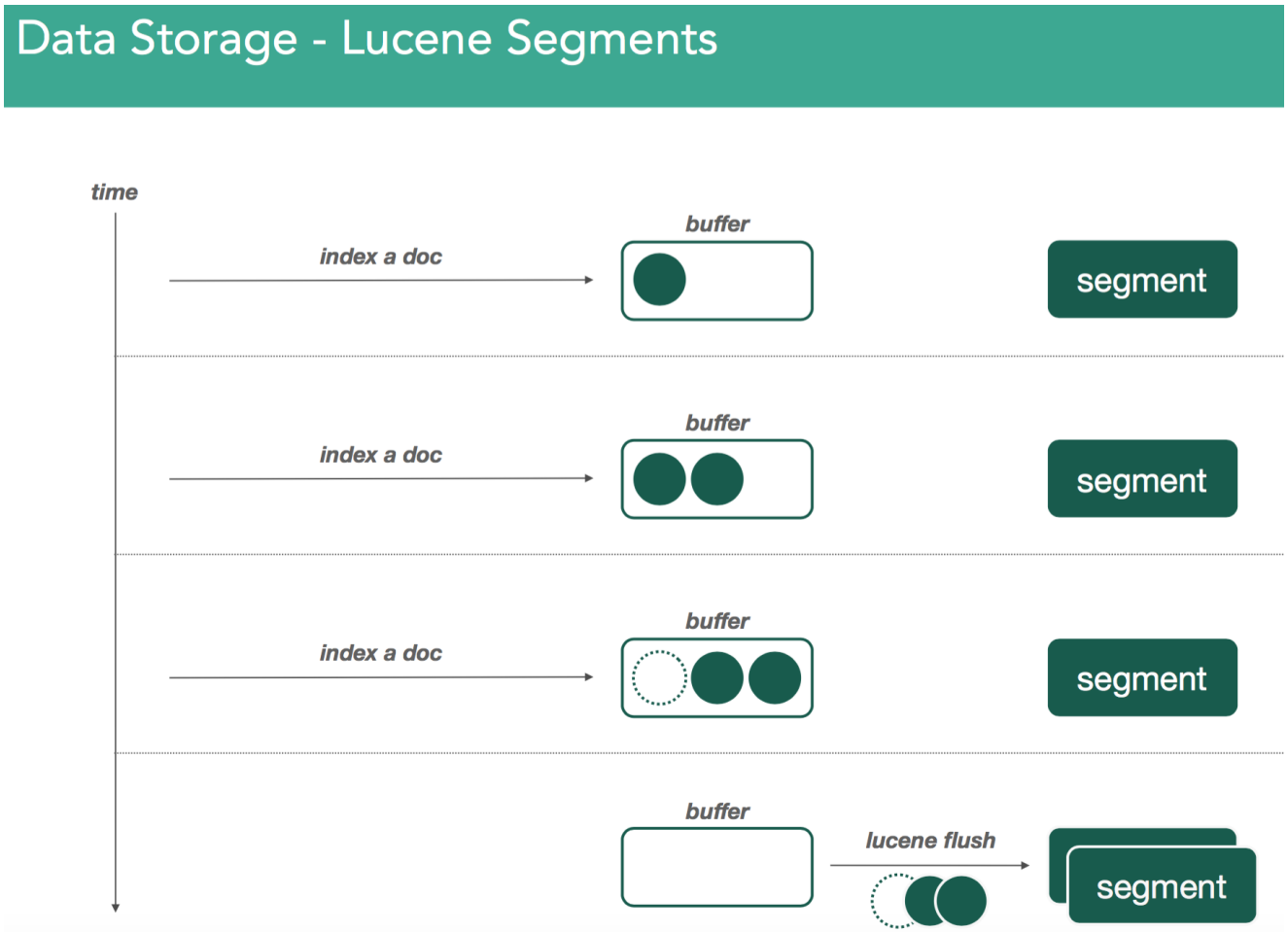
Figure: Sharding and replication with elasticsearch (`elastic.co`).

2 types of nodes:

Master Allocates the shards and the replicas to the other nodes.

Datanode Indexes, stores and reads the documents.

Elasticsearch



elastic.co

Elasticsearch - Lucene backend

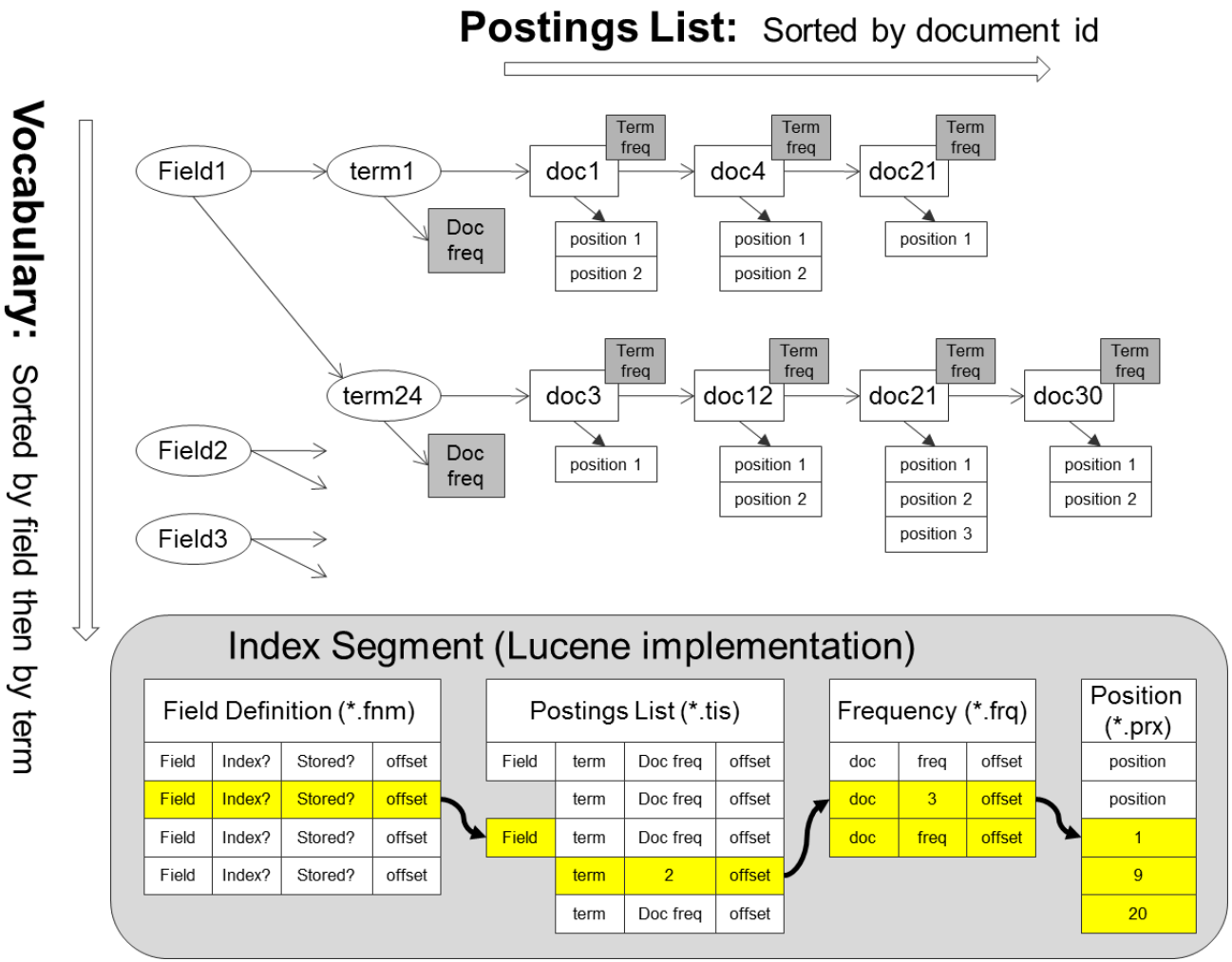


Figure: Lucene indexing process (`4.bp.blogspot.com`).

Distributed data processing

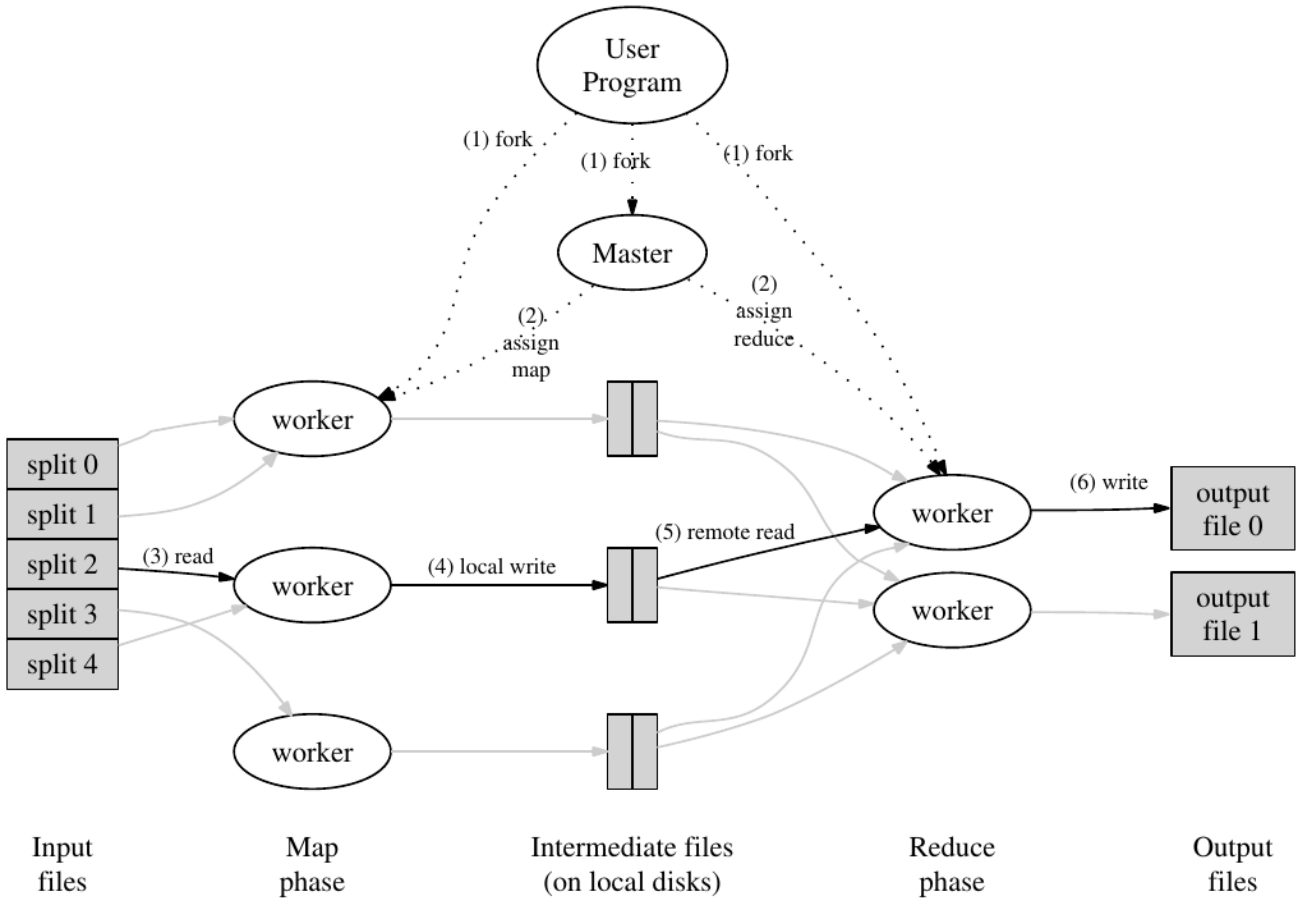


Figure: MapReduce: Distributed processing method for cluster computing (Dean *et al.*, 2004).

Batch processing

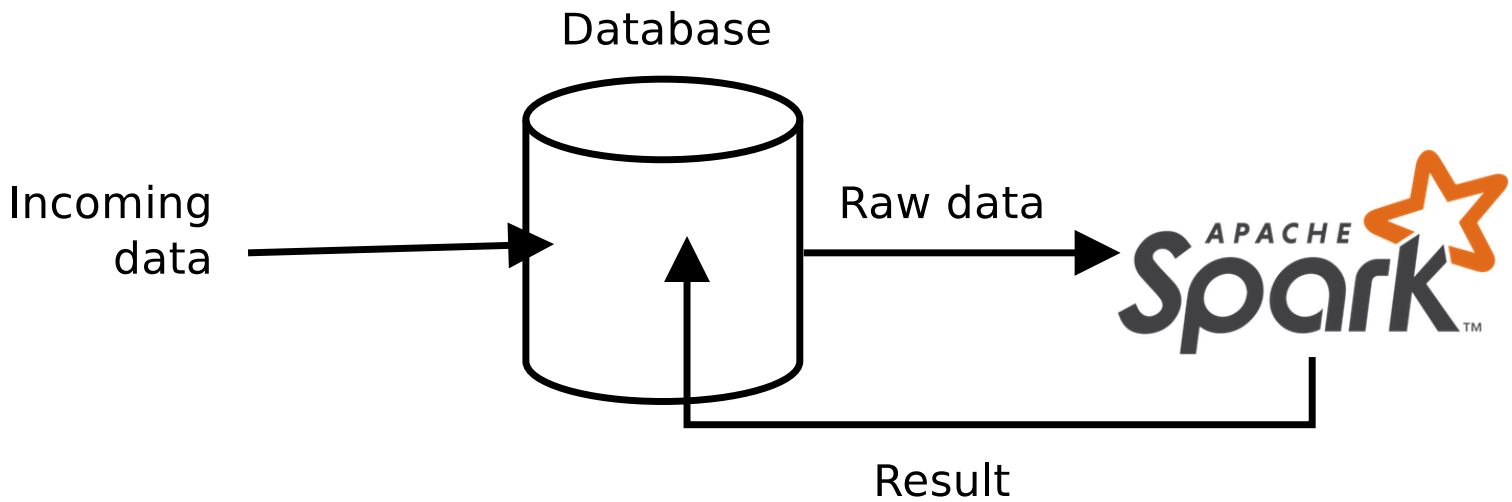


Figure: Distributed batch processing, *e.g.* with Apache Spark.

Streaming analytics

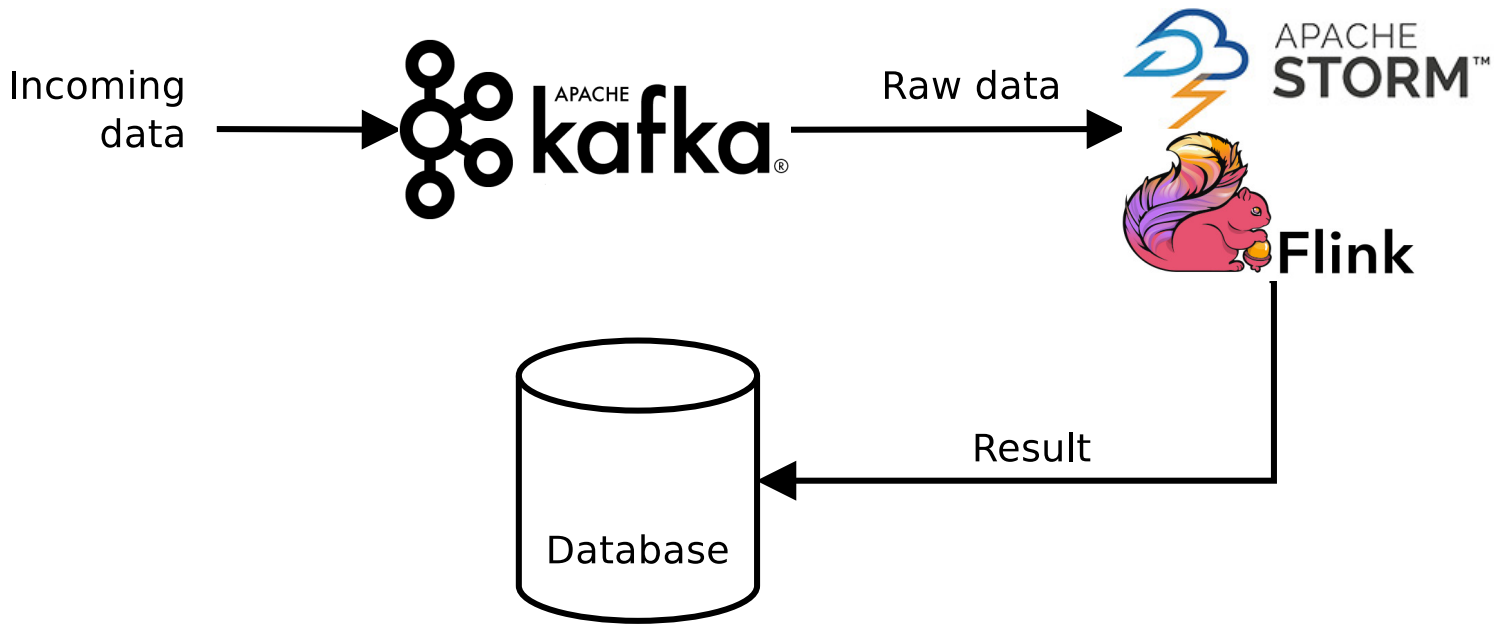


Figure: Distributed streaming analytics using *e.g.* Apache Storm or Flink.

Benchmarking with YCSB

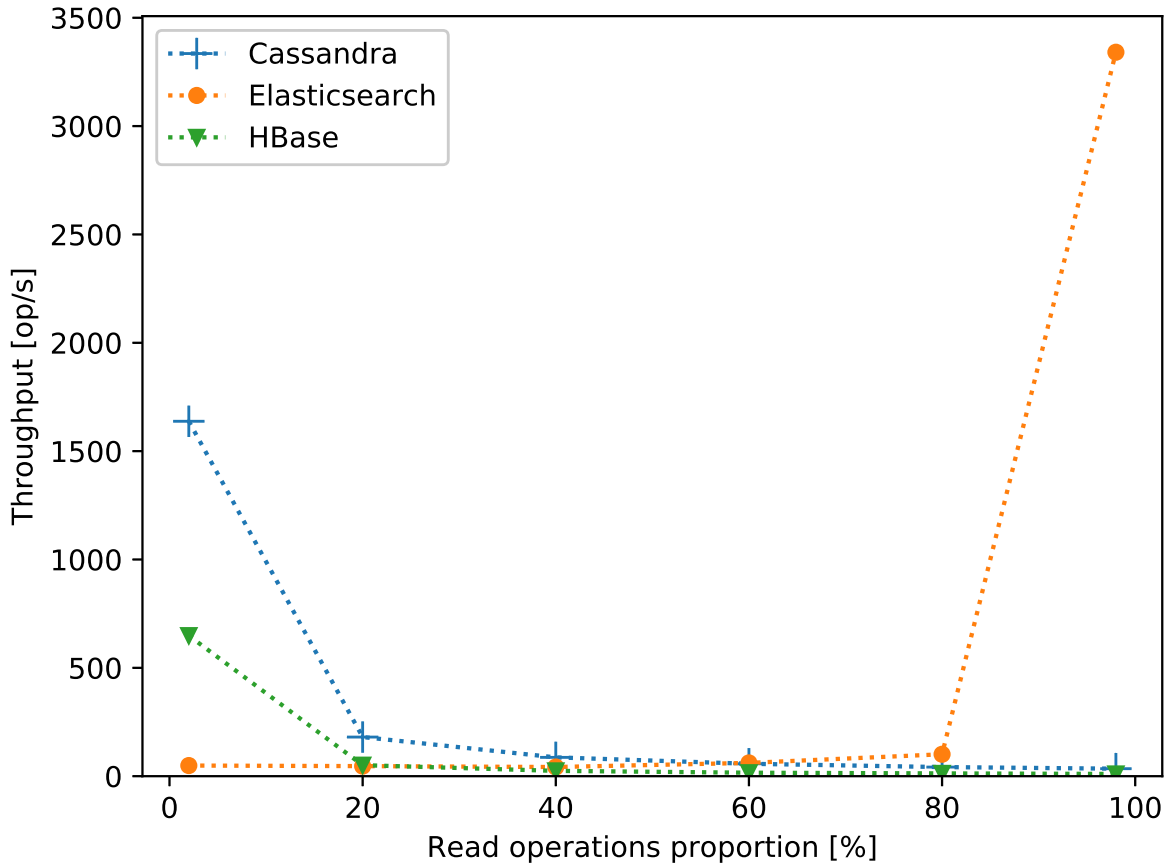
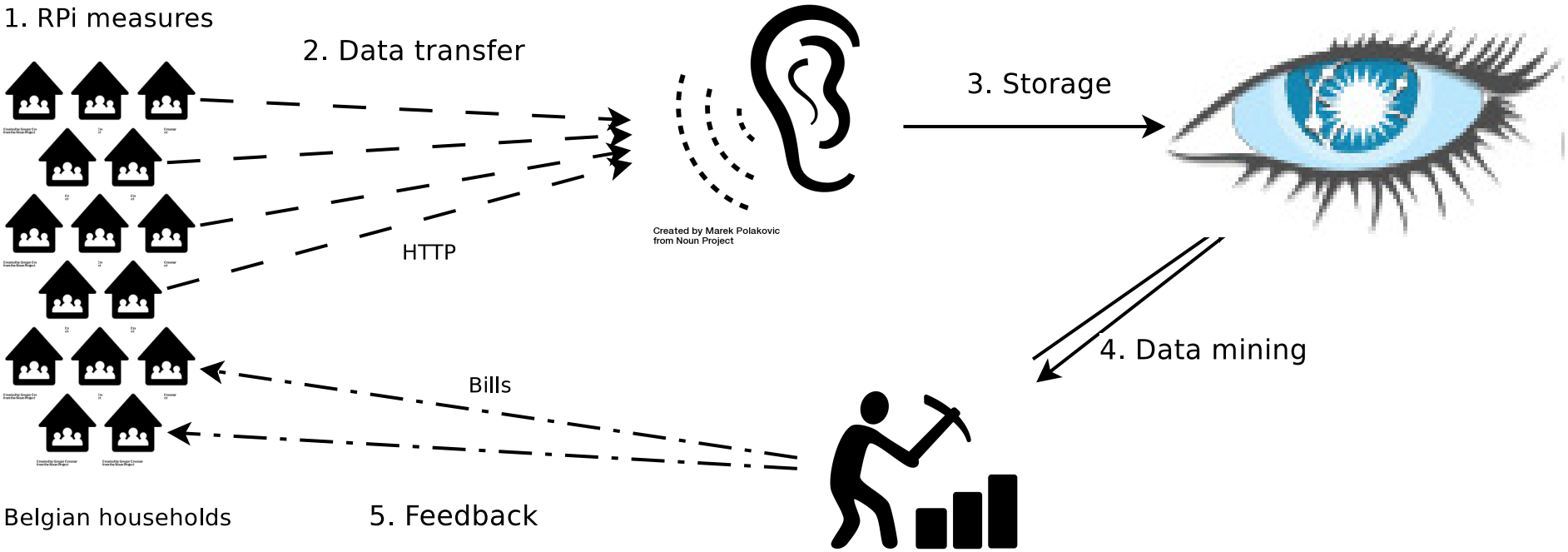


Figure: Throughput for 48 YCSB clients against a 11-nodes cluster for a varying read operations proportion with 1 M rows of 32×8 B.

Application



"Household" by Gregor Cresnar/thenounproject.com/CC BY 3.0
"Listen" by Marek Polakovic/thenounproject.com/CC BY 3.0

Figure: Dataflow for IoT application in household energy monitoring.

Application

Figure: Prototypes of home sensors.

Application

Wikimedia Foundation:
`https://grafana.wikimedia.org/`

References

[1][2][3]


Alejandro Corbellini et al. “Persisting big-data: The NoSQL landscape”. In: *Information Systems* 63 (2017), pp. 1–23.


Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: *OSDI 04*. 2004.


Fay Chang et al. “Bigtable: A Distributed Storage System for Structured Data”. In: *OSDI 06*. 2006.