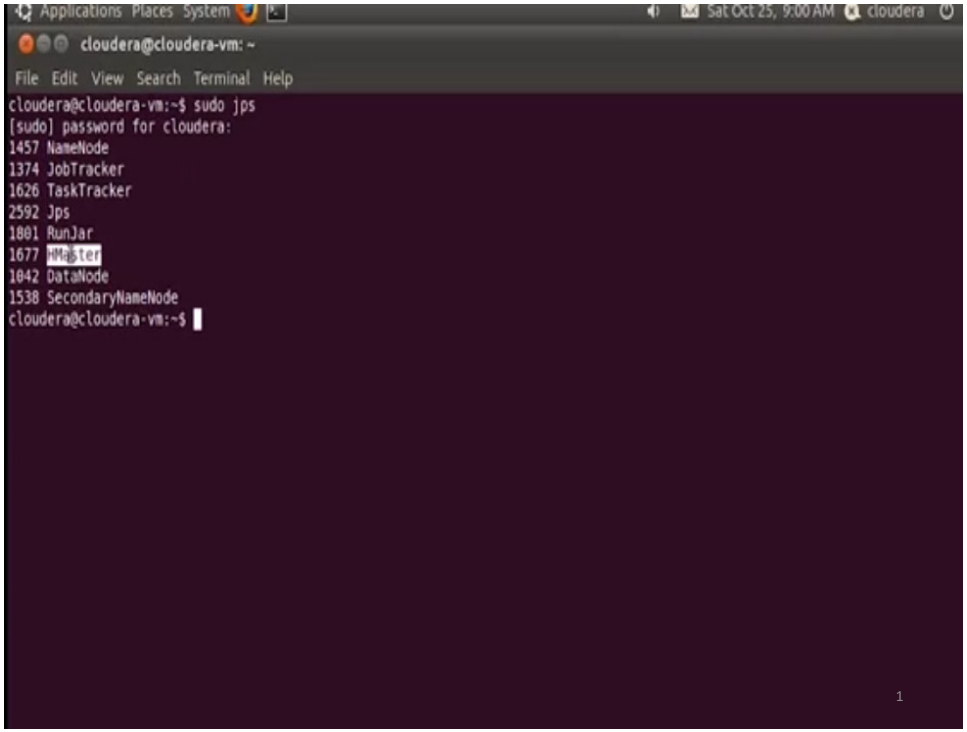
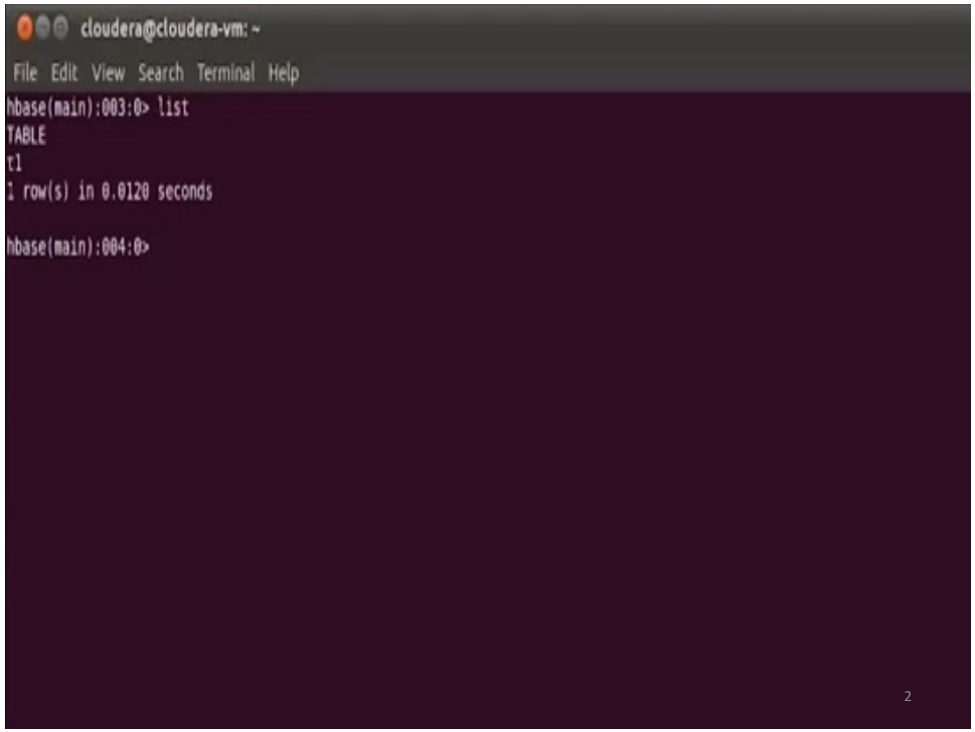
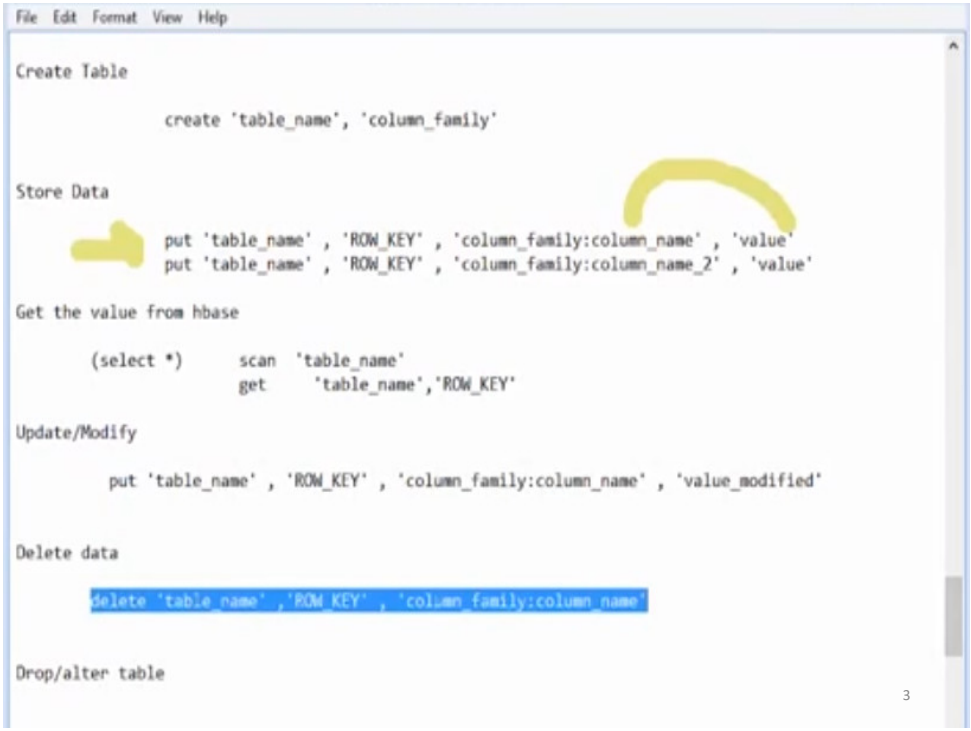

Applications Places System
Sat Oct 25, 9:00 AM cloudera
cloudera@cloudera-vm: ~
File Edit View Search Terminal Help
cloudera@cloudera-vm:~$ sudo jps
[sudo] password for cloudera:
1457 NameNode
1374 JobTracker
1626 TaskTracker
2592 Jps
1801 RunJar
1677 HMaster
1042 DataNode
1538 SecondaryNameNode
cloudera@cloudera-vm:~$

1


cloudera@cloudera-vm: ~
File Edit View Search Terminal Help
hbase(main):003:0> list
TABLE
t1
1 row(s) in 0.0120 seconds
hbase(main):004:0>

2


File Edit Format View Help
Create Table
create 'table_name', 'column_family'
Store Data
put 'table_name' , 'ROW_KEY' , 'column_family:column_name' , 'value'
put 'table_name' , 'ROW_KEY' , 'column_family:column_name_2' , 'value'
Get the value from hbase
(select *) scan 'table_name'
get 'table_name','ROW_KEY'
Update/Modify
put 'table_name' , 'ROW_KEY' , 'column_family:column_name' , 'value_modified'
Delete data
delete 'table_name' ,'ROW_KEY' , 'column_family:column_name'
Drop/alter table

3


hbase(main):004:0> create 'htest' , 'cf'
0 row(s) in 1.1650 seconds
hbase(main):005:0> list
TABLE
htest
t1
2 row(s) in 0.0090 seconds
hbase(main):006:0>

4

```
File  Edit  View  Search  Terminal  Help
hbase(main):004:0> create 'htest' , 'cf'
0 row(s) in 1.1650 seconds

hbase(main):005:0> list
TABLE
htest
t1
2 row(s) in 0.0090 seconds

hbase(main):006:0> put 'htest' , 'r1' , 'cf:c1' ,'v1'
0 row(s) in 0.1020 seconds

hbase(main):007:0> put 'htest' , 'r1' , 'cf:c2' ,'v2'
0 row(s) in 0.0510 seconds

hbase(main):008:0> put 'htest' , 'r1' , 'cf:c3' ,'v3'
0 row(s) in 0.0080 seconds

hbase(main):009:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
 r1                           column=cf:c3, timestamp=1414254290246, value=v3
1 row(s) in 0.0590 seconds

hbase(main):010:0>
```



```
hbase(main):009:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
 r1                           column=cf:c3, timestamp=1414254290246, value=v3
1 row(s) in 0.0590 seconds

hbase(main):010:0> get 'htest' , 'r1'
COLUMN                        CELL
 cf:c1                        timestamp=1414254248004, value=v1
 cf:c2                        timestamp=1414254280456, value=v2
 cf:c3                        timestamp=1414254290246, value=v3
3 row(s) in 0.0360 seconds

hbase(main):011:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated'
0 row(s) in 0.0220 seconds

hbase(main):012:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated_3'
0 row(s) in 0.0110 seconds
```



```
hbase(main):009:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
 r1                           column=cf:c3, timestamp=1414254290246, value=v3
1 row(s) in 0.0590 seconds

hbase(main):010:0> get 'htest' , 'r1'
COLUMN                        CELL
 cf:c1                        timestamp=1414254248004, value=v1
 cf:c2                        timestamp=1414254280456, value=v2
 cf:c3                        timestamp=1414254290246, value=v3
3 row(s) in 0.0360 seconds

hbase(main):011:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated'
0 row(s) in 0.0220 seconds

hbase(main):012:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated_3'
0 row(s) in 0.0110 seconds
```



```
hbase(main):011:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated'
0 row(s) in 0.0220 seconds

hbase(main):012:0> put 'htest' , 'r1' , 'cf:c3' ,'v3_updated_3'
0 row(s) in 0.0110 seconds

hbase(main):013:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
 r1                           column=cf:c3, timestamp=1414254592531, value=v3_updated_3
1 row(s) in 0.0760 seconds

hbase(main):014:0> get 'htest' , 'r1' , {COLUMN=>'cf:c3',VERSIONS=>3}
COLUMN                        CELL
 cf:c3                        timestamp=1414254592531, value=v3_updated_3
 cf:c3                        timestamp=1414254579862, value=v3_updated
 cf:c3                        timestamp=1414254290246, value=v3
3 row(s) in 0.0460 seconds

hbase(main):015:0> get 'htest' , 'r1' , {COLUMN=>'cf:c3',VERSIONS=>2}
COLUMN                        CELL
 cf:c3                        timestamp=1414254592531, value=v3_updated_3
 cf:c3                        timestamp=1414254579862, value=v3_updated
2 row(s) in 0.0490 seconds
```

```
hbase(main):026:0> put 'htest' , 'r1' , 'cf_altered:c3' ,'v3_updated_6'
0 row(s) in 0.0180 seconds

hbase(main):027:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf_altered:c3, timestamp=1414255410250, value=v3_updated_6
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
 r1                           column=cf:c3, timestamp=1414255008377, value=v3_updated_6
1 row(s) in 0.0200 seconds

hbase(main):028:0> delete 'htest' , 'r1' , 'cf:c3'
0 row(s) in 0.0140 seconds

hbase(main):029:0> scan 'htest'
ROW                           COLUMN+CELL
 r1                           column=cf_altered:c3, timestamp=1414255410250, value=v3_updated_6
 r1                           column=cf:c1, timestamp=1414254248004, value=v1
 r1                           column=cf:c2, timestamp=1414254280456, value=v2
1 row(s) in 0.0330 seconds
```


Row Vs. Column Oriented DBS

SQL Schema

URLS					
url-id INTEGER PK	**url** VARCHAR(4096)	**ref_short_id** CHAR(8)	**title** VARCHAR(200)	**description** VARCHAR(400)	**content** TEXT
1	http://hbase.apache.org	3fG4J	HBase Home	Great tool!	<html><head><title>Hbase Home</ti...
2	http://larsgeorge.com	1337	Lineland	<NULL>	<html><body>Newest Posts...
3	http://foobar.com/index.html	Hf34h	<NULL>	Read about it...	404 Page not found.
4	http://cnn.com/page123.html	0o001	Sport News	Soccer News	<html><body>Results, Reviews,...

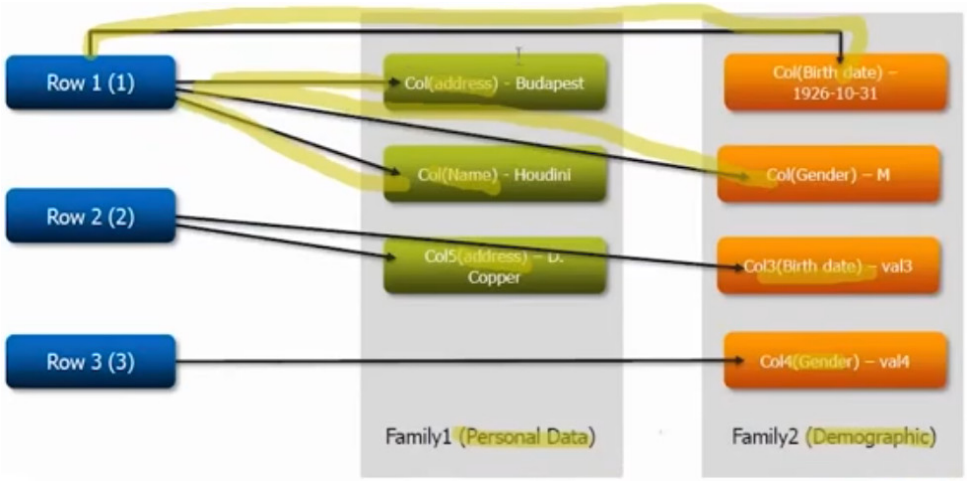
Column Oriented Storage

Col 1:**url**	http://hbase.apache.org	http://larsgeorge.com	http://foobar.com/index.html	http://cnn.com/page12...	...
Col 2: **ref_short_id**	3fG4J	1337	Hf34h	0o001	...
Col 3: **title**	HBase Home	Lineland	<NULL>	Sport News	...
Col 4: **description**	Great tool!	<NULL>	Read about it...	Soccer News	...
Col 5: **content**	<html><head><title>HBa...	<<html><body>Newest Po...	404 Page not found.	<html><body>Results,...	

Row Vs. Column Oriented DBS

Row key	Personal_data		demographic	
Persons ID	Name	Address	Birth Date	Gender
1	H. Houdini	Budapest	1926-10-31	M
2	D. Copper		1956-09-16	M
3	Merlin		1136-12-03	F
4	.....	.....	.....	M
500,000,000	F. Cadillac	Nevada	1964-01-07	M

Row Vs. Column Oriented DBS



How Does it Look Like ?

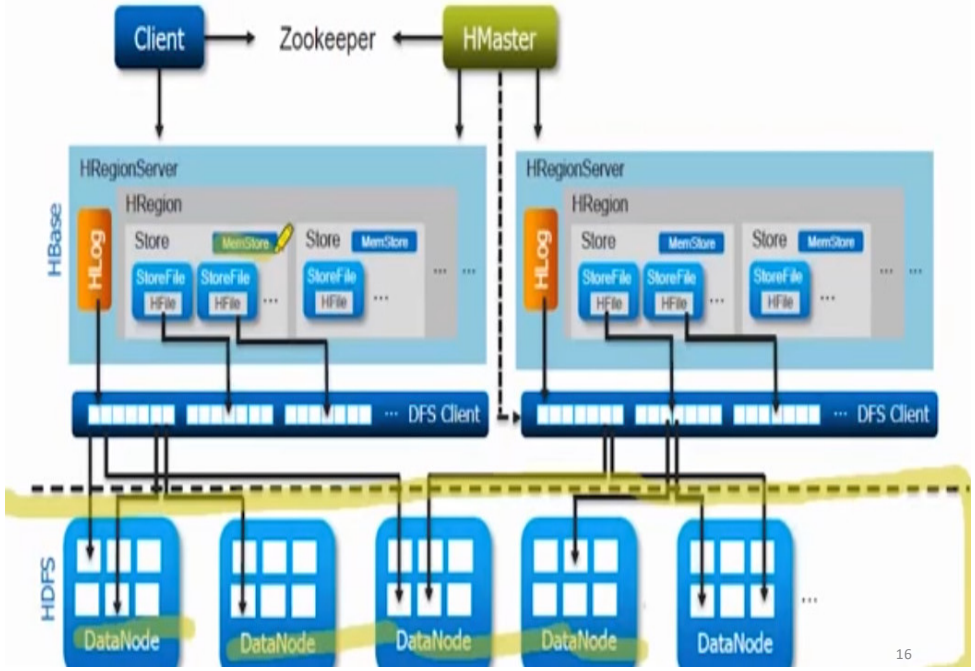
```
ROW                          COLUMN+CELL
 row1                        column=data:1, timestamp=1262873693421, value=value
1
 row2                        column=data:2, timestamp=1262873716906, value=value
2
 row3                        column=data:3, timestamp=1262873738843, value=value
3
3 row(s) in 0.0150 seconds
```

What it means?

Row Key	Column Family: Column Qualifier	Values
• Unique for each row • Identifies each row	• Less number of families gives faster access • Families are fixed column qualifiers are not	• Various versions of values are maintained • Scan shows only recent version

HBase Components

- ✓ Table made of regions
- ✓ Region – a range of rows stored together
- ✓ Region servers- serves one or more regions
  - ✓ A region is served by only one region server
- ✓ Master server – daemon responsible for managing HBase cluster
- ✓ HBase stores its data into HDFS
  - ✓ Relies on HDFS's High Availability and fault tolerance



Cassandra

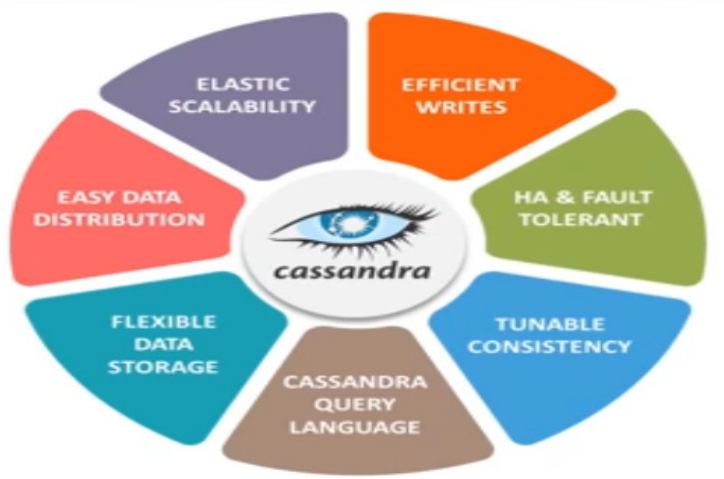
Apache's Cassandra was chosen by Instagram as the perfect solution to this problem, because of the following features it provides:

- Fully distributed with no single point of failure
- Free and open source with deep developer suppor
- Linearly Scalable
- Larger-than-memory Datasets
- Built-in-class performance
- Fully Durable
- Integrated Caching and Tuneable Consistency

Cassandra

Apache Cassandra is a **free and open-source distributed NoSQL database management system** designed to **handle large amounts of data** across many commodity servers, providing **high availability** with no single point of failure.

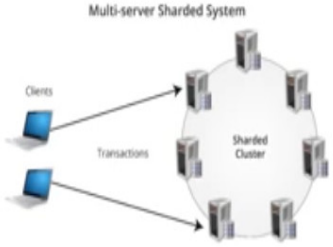
Cassandra



Cassandra

EASY DATA DISTRIBUTION

- Data distribution is very easy
- Provides the *flexibility to distribute data*
- *No master-slave issues* due to peer-to-peer architecture.



ELASTIC SCALABILITY

- Cassandra *scales horizontally*
- During scaling , *Read and Write throughput increase simultaneously*
- There is a *Zero downtime.*

Cassandra

HIGH AVAILABILITY & FAULT TOLERANT

- Data is *automatically replicated* to multiple nodes *for fault-tolerance*
- Failed nodes can be replaced with no downtime, due to replication

EFFICIENT WRITES

- Was designed to *run on cheap commodity hardware*
- It performs *blazingly fast writes*
- Can store hundreds of terabytes of data, without sacrificing the read efficiency

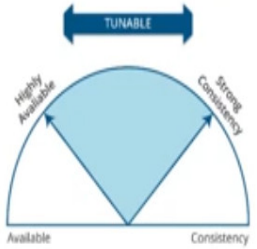
CQL

CQL

- Cassandra introduced the *Cassandra Query Language(CQL)*
- CQL 3 is the default and the primary interface into the Cassandra DBMS

Cassandra

Features of Cassandra

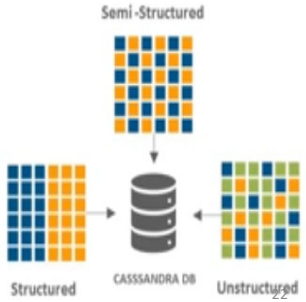


TUNABLE CONSISTENCY

- Provides a means for *tuning the level of consistency required*
- Consistency makes sure that the *client is approved as soon as the cluster accepts the write*

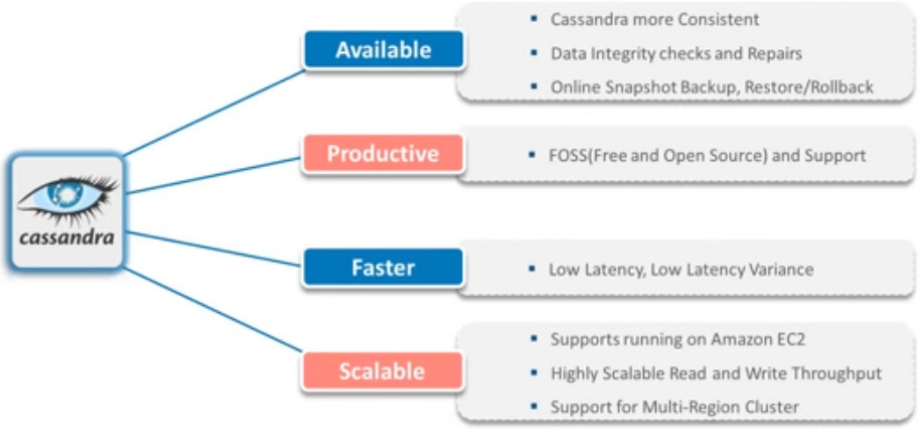
FLEXIBLE DATA STORAGE

- Cassandra accommodates *all possible data formats*
- *Structured, Semi-structured, and Unstructured*
- Can *dynamically accommodate changes* to data structures



Cassandra

Reasons to Chose Cassandra?



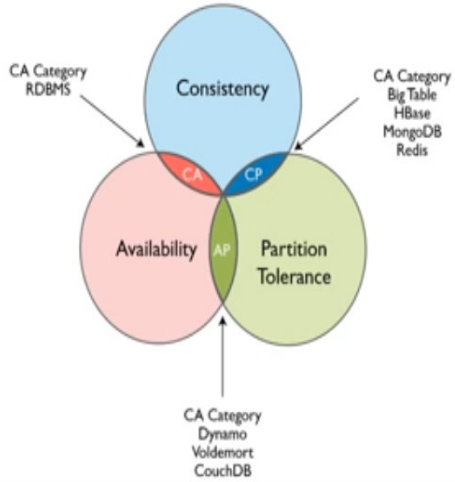
Cassandra

How Extensively Netflix Uses Cassandra

Cassandra in Netflix's Architecture

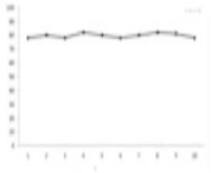
CAP Theorem

- Eric Brewer posited his CAP theorem in 2000, which states within a large-scale distributed data system, you can only have two out of the following three guarantees:
  - Consistency
  - Availability
  - Partition tolerance



CONSISTENCY

Read operation will return the value of the most recent write operation, causing all nodes to return the same data



AVAILABILITY

Every request gets a response on Success/Failure

System remains operational 100%. Every client gets a response, regardless of any individual node in the system

PARTITION TOLERANCE

System continues to work regardless of partial failure

Sustain any amount of network failure , which does not result in failure of the entire network