NoSQL Database

Presented by: Harshit Shah
Samir Sharan
Jeevan Reddy

NoSQL

- Interpreted as **Not only SQL** - provides more capabilities beyond the classical approach.
- Motivations for the approach: simple design, horizontal scaling, finer control over availability.
- Increasingly being used in big data and real-time web applications.
- RDBMS follows the ACID property
- NoSQL databases are “**BASE**” Systems (**B**asically **A**vailable, **S**oft state, **E**ventually consistent)

Types of NoSQL

- **Key Value:** Using a hash table where there is a unique key and a pointer to a particular item. Eg: Redis, Riak, Memcached
- **Column Oriented:** These were created to store and process very large amounts of data distributed over many machines. There are still keys but they point to multiple columns. Eg: Cassandra, HBase, Big Table, Apache Parquet.
- **Document Stored:** The semi-structured documents are stored in formats like JSON. Document databases are essentially the next level of key-value, allowing nested values associated with each key. Eg: MongoDB, CouchDB.
- **Graph Based:** Instead of tables of rows and columns and the rigid structure of SQL, a flexible graph model is used which, again, can scale across multiple machines. Eg: Neo4J

MongoDB

MongoDB is a database management system designed for web applications and internet infrastructure.

Features:

- MongoDB’s data model is document-oriented
- Secondary indexes - can create upto 64 indexes per collection
- Replication - replica set
- Speed and durability - journaling(append-only log)
- Scaling - Horizontal scaling via auto-sharding

Document Oriented

```
{ _id: ObjectID('4bd9e8e17cefd644108961bb'),
  title: 'Adventures in Databases',
  url: 'http://example.com/databases.txt',

  author: 'msmith',
  vote_count: 20,
  tags: ['databases', 'mongodb', 'indexing'],
  image: {
    url: 'http://example.com/db.jpg',
    caption: '',
    type: 'jpg',
    size: 75381,
    data: "Binary"
  },
  comments: [
    { user: 'bjones',
      text: 'Interesting article!'
    },
    { user: 'blogger',
      text: 'Another related article is at http://example.com/db/db.txt'
    }
  ]
```

_id field is primary key

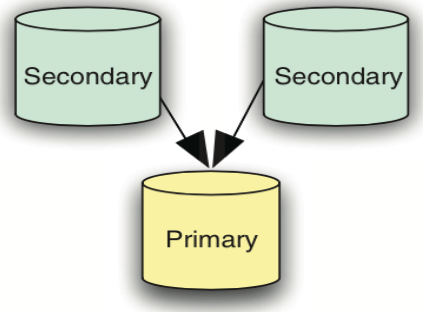
1 Tags stored as array of strings

2 Attribute points to another document

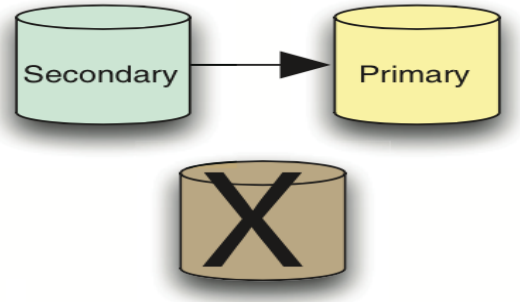
3 Comments stored as array of comment objects

Replication

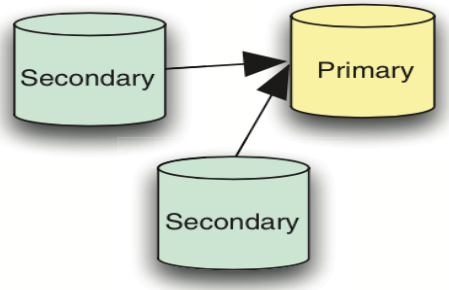
1. A working replica set



2. Original primary node fails and a secondary is promoted to primary



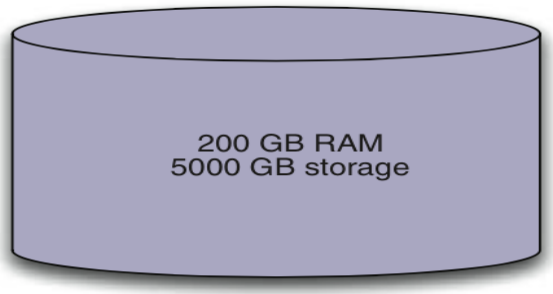
3. Original primary comes back online as a secondary



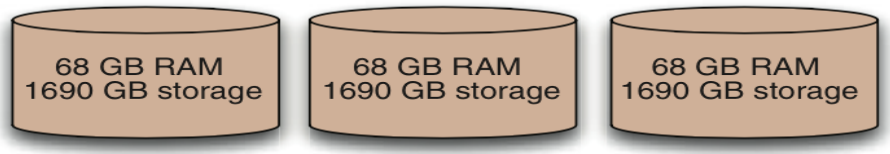
Horizontal vs Vertical Scaling

Original database

Scaling up increases the capacity of a single machine.



Scaling out adds more machines of the similar size.



Core Server and Shell

- The core database server runs via an executable called mongod (mongod.exe on Windows).
- The mongod server process receives commands over a network socket using a custom binary protocol.
- The MongoDB command shell is a JavaScript-based tool for administering the database and manipulating data
- The mongo executable loads the shell and connects to a specified mongod process

MongoDB through JavaScript Shell

Starting Shell:

- ./mongod: starts the mongoDB instance
- ./mongo: starts the mongoDB shell

```
Samirs-MacBook-Air:bin insignia$ ./mongo
MongoDB shell version: 3.0.4
connecting to: test
Server has startup warnings:
2015-06-24T19:28:26.746-0400 I CONTROL  [initandlisten]
2015-06-24T19:28:26.746-0400 I CONTROL  [initandlisten] ** WARNING: soft rlimits
 too low. Number of files is 256, should be at least 1000
>
```

Queries

```
Samirs-MacBook-Air:bin insignia$ ./mongo
MongoDB shell version: 3.0.4
connecting to: test
Server has startup warnings:
2015-06-24T19:28:26.746-0400 I CONTROL  [initandlisten]
2015-06-24T19:28:26.746-0400 I CONTROL  [initandlisten] ** WARNING: soft rlimits
 too low. Number of files is 256, should be at least 1000
> use demo
switched to db demo
> db.users.insert({username: "samir"})
WriteResult({ "nInserted" : 1 })
> db.users.find()
{ "_id" : ObjectId("558b6bcaf2cc197babb43e21"), "username" : "samir" }
> db.users.save({username: "harshit"})
WriteResult({ "nInserted" : 1 })
> db.users.count()
2
> db.users.find({username: "samir"})
{ "_id" : ObjectId("558b6bcaf2cc197babb43e21"), "username" : "samir" }
> db.users.update({username: "samir"}, {$set: {country: "USA"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.update({username: 'samir'}, {$unset: {country: 'USA'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.users.update({username: 'samir'}, {$set: {favorites: {cities: ['Chicago', '
Cheyenne'], movies: ['Dark Knight', 'Drop']}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.users.remove({'favorites.cities': 'Cheyenne'})
WriteResult({ "nRemoved" : 1 })
> db.users.drop()
true
>
```

```
> db.users.createIndex({userid: 1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

Why MongoDB ?

- MongoDB has the features of both key-value stores as well as the Relational Databases: MongoDB represents a mean between these two designs.
- MongoDB is well suited as a primary data store for web applications, for analytics and logging applications, and for any application requiring a medium-grade cache
- MongoDB is also good for capturing data whose structure can’t be known in advance.

Limitations

- MongoDB should usually be run on 64-bit machines
- MongoDB is best run on a dedicated server
- It’s important to run MongoDB with replication, especially if you’re not running with journaling enabled. Because MongoDB uses memory-mapped files, any unclean shutdown of a mongod not running with journaling may result in corruption.

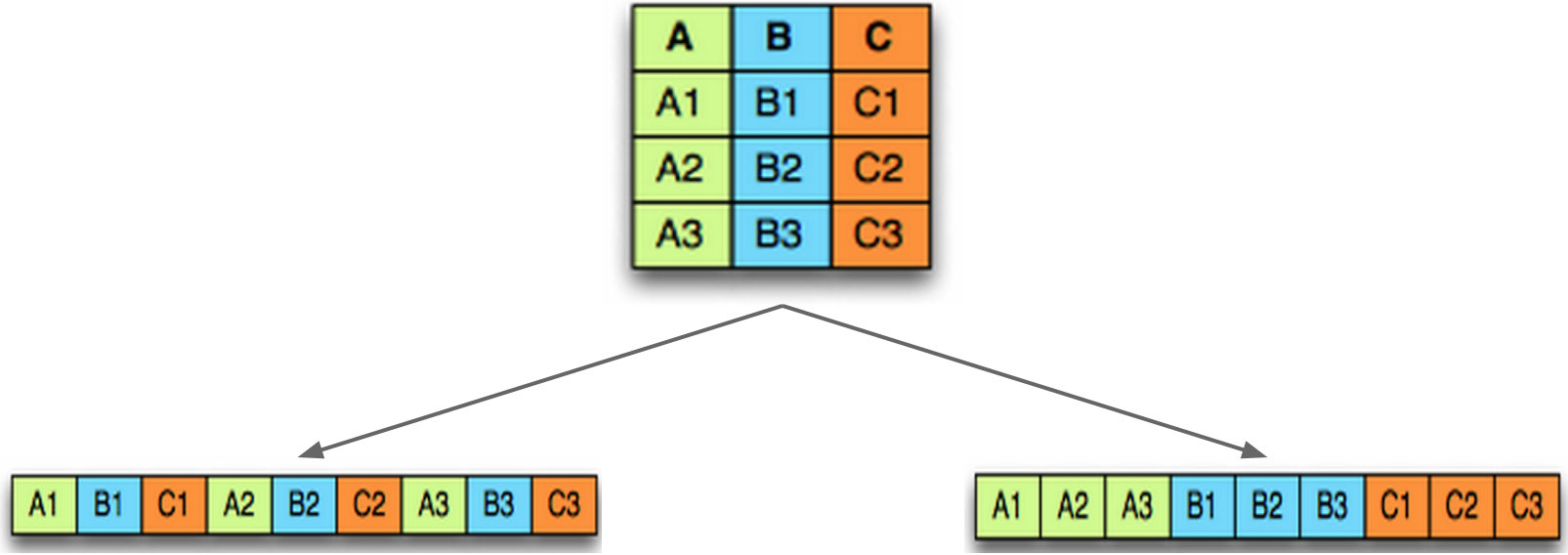
Relational Database Management Systems

- Typical RDBMS
  - Row oriented storage system
  - Mostly used for detail level data retrieval

- Issues?
  - High I/O when accessing large volumes of data
  - Space expensive
  - Poor data compression

- Solution?

Columnar Format Storage



Apache Parquet

Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language

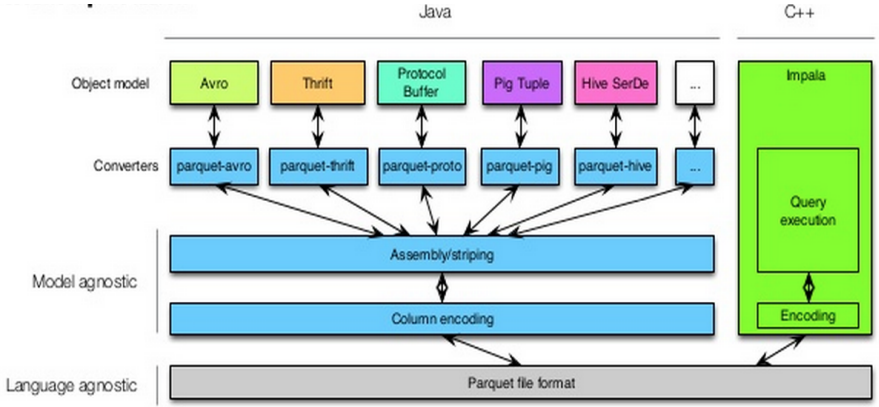
Evolution Timeline

- Fall 2012 - Twitter and Cloudera merge efforts to develop columnar formats
- March 2013 - Criteo signs on for Hive integration
- July 2013 - 1.0 release
- March 2014 - 26 incremental releases
- April 2015 - Parquet graduated as a top level project from ASF

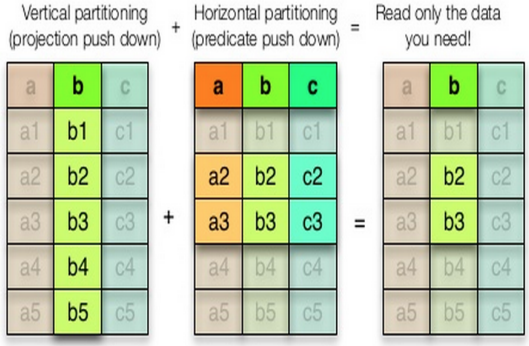
Parquet Design Advantages

- Interoperability
- Space Efficiency
- Query Efficiency

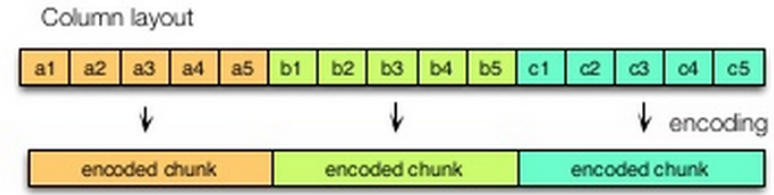
Interoperability



Query Efficiency



Space Efficiency



Parquet with PySpark

```
df = sqlContext.read.load("examples/src/main/resources/users.parquet")
df.select("name", "favorite_color").write.save("namesAndFavColors.parquet")
```

```
df = sqlContext.read.load("examples/src/main/resources/people.json", format="json")
df.select("name", "age").write.save("namesAndAges.parquet", format="parquet")
```

Adopters of Parquet

cloudera®

NETFLIX

criteo

salesforce

What is Cassandra?

Apache Cassandra is a

Distributed...

High performance...

Extremely scalable...

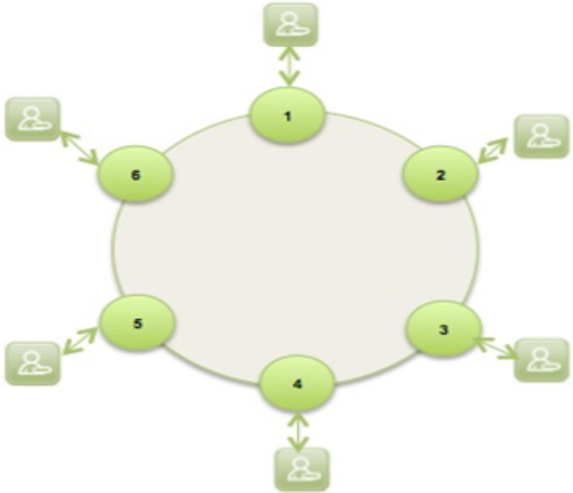
Fault Tolerant(i.e. no single point of failure)...

post-relational database solution which can serve as both real-time datastore online applications/transactions and as read-intensive database for BI systems

History of Cassandra

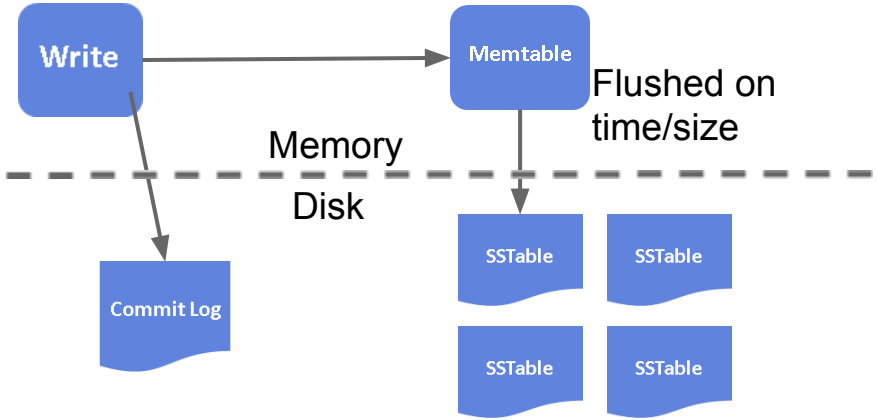
Architecture

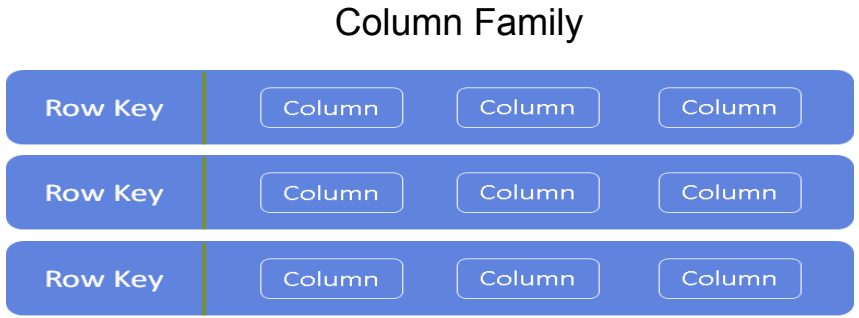
- Designed to withstand system failures
- Peer-to-peer architecture, distributed system
- No master and slave nodes, all nodes are the same
- Data partitioned, replicated among nodes to ensure fault tolerance
- Uses Gossip Protocol to exchange data, Can read/write-anywhere
- Data written to in-memory structure(memtable) and to disk once memory structure is full(an SStable)



Schema

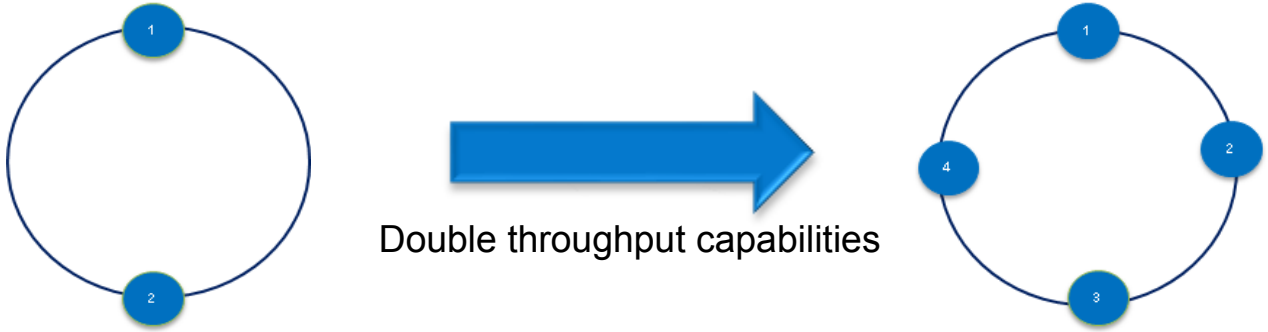
- Schema is mirrored after Google Bigtable
- Row-oriented, column structure
- A keyspace is akin to a database in the RDBMS world
- A column family is similar to RDBMS table, but is more flexible
- A row in a column family is indexed by its key, other columns may be indexed as well



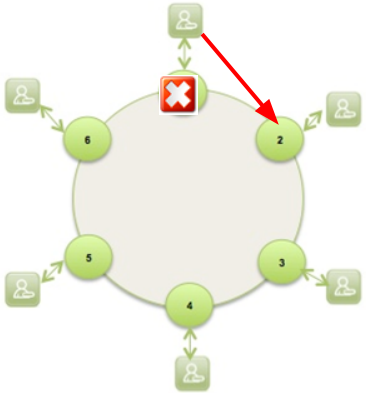


Why Cassandra

- Big Data Scalability:



- No Single Point of Failu



Why Cassandra(Cont..)

- Easy replication/data distribution
- No need for Caching software
- Tunable data consistency
- Flexible schema
- Data compression
- Makes it easier to use Hadoop integration
- CQL Language

```
create column family profiles
with key_validation_class = UTF8Type
and comparator = UTF8Type
and column_metadata = [
   {column_name: first_name, validation_class: UTF8Type},
   {column_name: last_name, validation_class: UTF8Type},
   {column_name: year, validation_class: IntegerType}
];
```

Limitations

- Do not provide ad-hoc query
- No join or subquery support
- Limited support for aggregation
- Ordering is done per partition
- All data for a single partition must fit (on disk) on a single machine in the cluster
- A single column value may not be larger than 2GB
- Maximum number of cells in a single partition is 2 billion

Questions?

SELECT answer(question) FROM audience