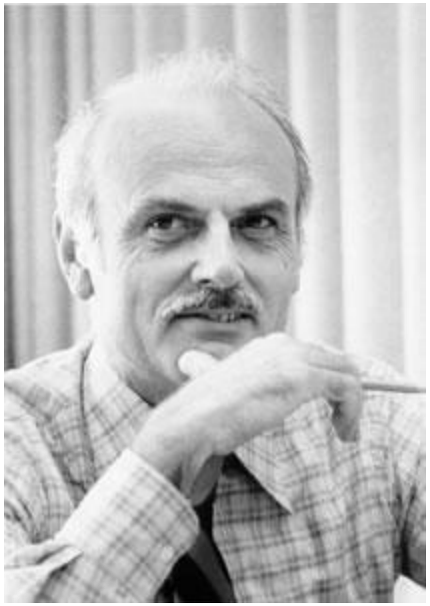
Not Only SQL databases

Tomas Barina

Database history 1/2

"A database system is a way of organizing information on a computer, implemented by a set of computer programs."

Dr. Codd introduced relational model, normalization and SEQEL language



Database history 2/2

- Larry Ellison based on Codd's paper started Relational Software Inc. (1977) today Oracle
- Many others have inspired from Codd => Informix, MySQL, PostgreSQL, etc..
- RDBMS are still the most widely used database systems

Transactions

- Set of operation treated as atomic
- ACID
  - **Atomicity** - All operations are executed at once or rolled back
  - **Consistency** - System can switch only between legal states
  - **Isolation** - Others do not see modified data until they are committed
  - **Durability** - Data is persisted even in case of hw/sw crash

End of "one size fits all" era

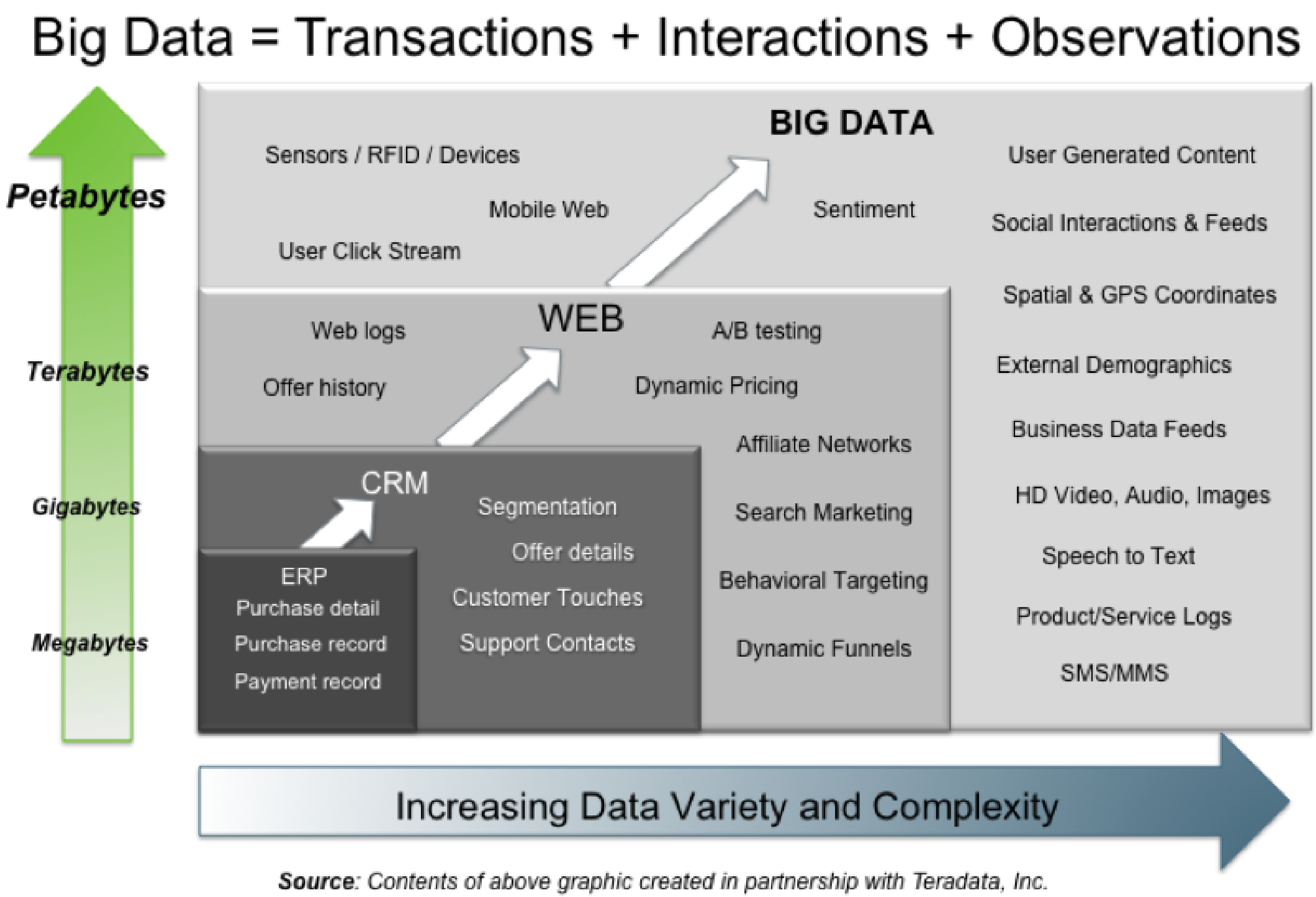
img source: http://morganlinton.com/wp-content/uploads/2010/08/one-size-fits-all.jpg

What are today's requirements?

- Large volume of data (Informational explosion)
  - In 2011 ~1 800 exabytes of data created
  - World information is doubling every 2 years
- Dynamic adoption of changes
  - ALTER on big table is issue
- Strong parallelism
- Minimal downtime (SLAs)
- Low latency
- Low price - e.g. commodity hardware
- Streaming data
- Analytics
- ....


12+ TBs
of tweet data
every day
? TBs of
data every day
500+ TBs
of
log data
every day
30 billion RFID
tags today
(1.3B in 2005)
4.6
billion
camera
phones
world
wide
100s of
millions
of GPS
enabled
devices
sold
annually
76 million smart
meters in 2009…
200M by 2014
2+
billion
people
on the
Web by
end 2011
http

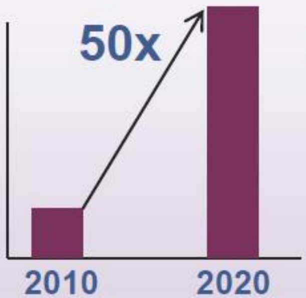
Big Data era



img src: http://hortonworks.com/wp-content/uploads/2012/05/bigdata_diagram.png

- **V⁴ =** Volume Velocity Variety Veracity

Cost efficiently processing the growing Volume



35 ZB

Responding to the increasing Velocity

30 Billion RFID sensors and counting

Collectively analyzing the broadening Variety

80% of the worlds data is unstructured

Establishing the Veracity of big data sources

1 in 3 business leaders don't trust the information they use to make decisions

How to satisfy these requirements?

- Many CPUs, memory
- But hard to fit to one node (even rack)
- Even SANs are expensive

Solution:

Split data and computation across servers

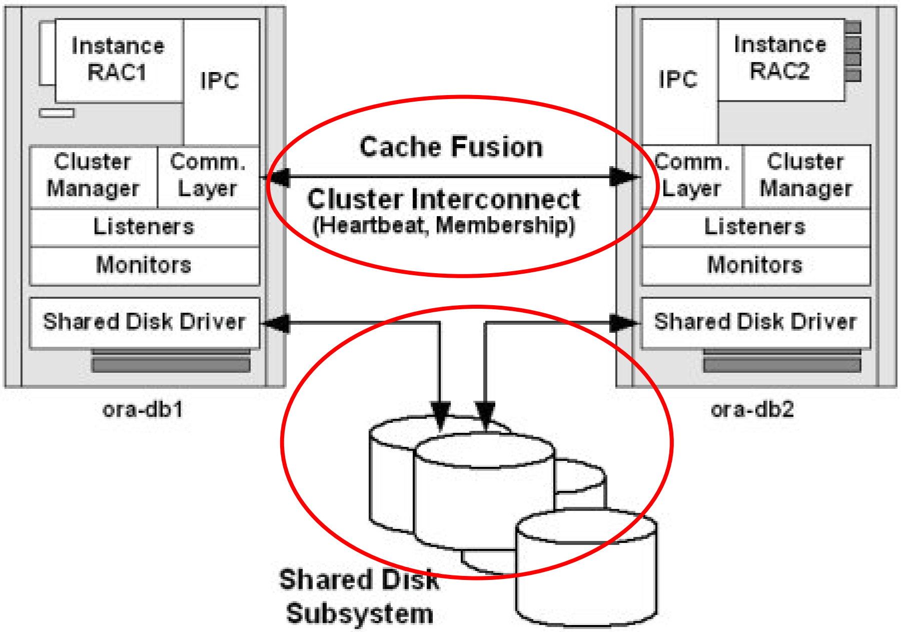
(horizontal scaling)

Try different DB architecture

Oracle RAC? - Superb but not for everything

- Add CPUs and memory to one server
- Add servers for failover or balancing (but limited number and non-linear scaling)
- Oracle RAC:

The following diagram, adapted from *Oracle Real Application Clusters* by Murali Vallath [*Elsevier Digital Press*, 2004] illustrates some of the basic components of an Oracle RAC cluster under Oracle 9*i*.



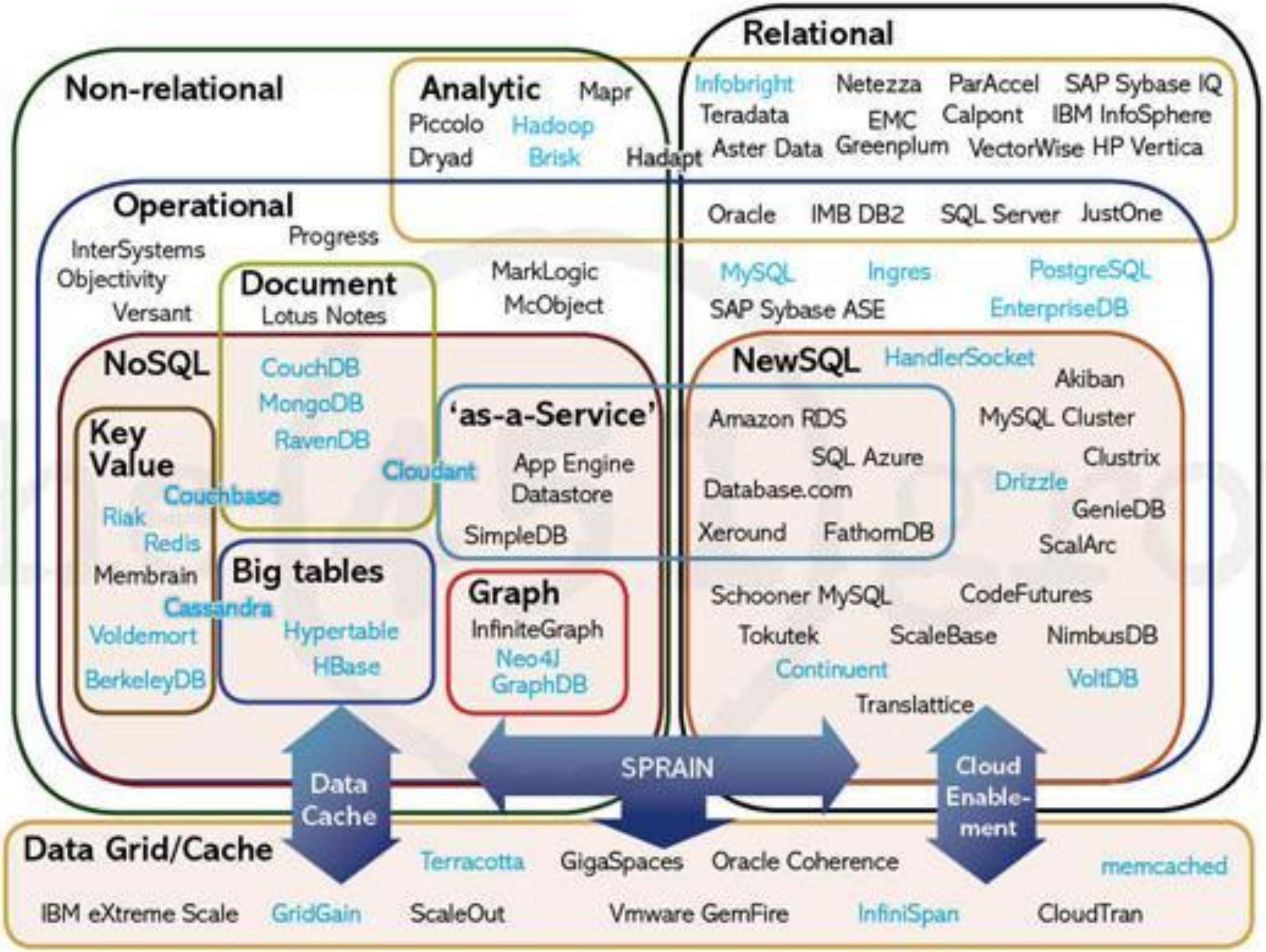
NoSQL

"Let's relax from some RDBMS features"

NoSQL = Not only SQL

- Different storage architecture
- Schemaless
- Relax JOINs
- Eventual consistency
- Elasticity
- Cheap hardware (Usually simplified)

Types of databases



(c) www.infoq.com

NoSQL taxonomy (1/2)

Key / Value

- Distributed Hash Table

Document database

- Semistructured, stores JSON/XML.

Graph database

- From graph theory. Stores vertices, edges, attributes.

NoSQL taxonomy (2/2)

Column store

- One key have multiple columns. Store similar column values nearby.

disk = Seagate ST2000DL003, 2TB

number of blocks N = 500

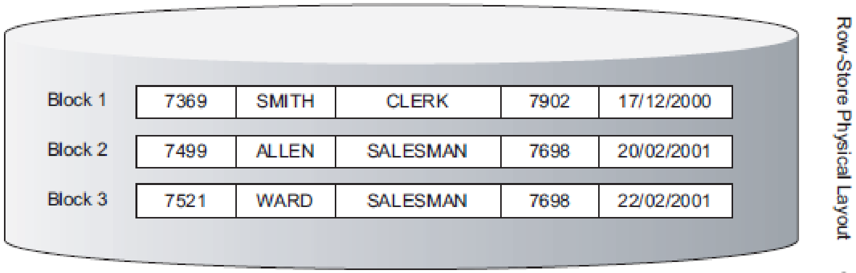
block size s = 8kB

Random Access:

$$t_{rand} = N \cdot (t_s + t_{rd} + t_r) = 500 \cdot (12 + 5.08 + 0.06) = \mathbf{8.5s}$$

Sequential read:

$$t_{seq} = t_s + t_{rd} + N \cdot t_r + T \cdot t_{ttt} = 12 + 5.08 + 30 + 16 = \mathbf{63ms}$$

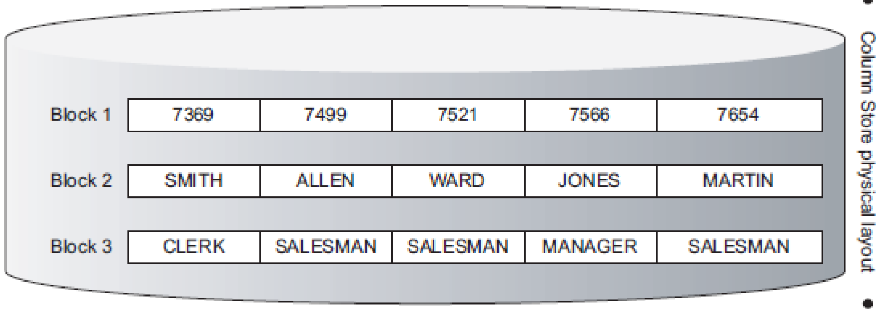


Row Database stores row values together

Logical Schema

EmpNo	EName	Job	Mgr	HireDate
7369	SMITH	CLERK	7902	17/12/1980
7499	ALLEN	SALESMAN	7698	20/02/1981
7521	WARD	SALESMAN	7698	22/02/1981
7566	JONES	MANAGER	7839	2/04/1981
7654	MARTIN	SALESMAN	7698	28/09/1981
7698	BLAKE	MANAGER	7839	1/05/1981
7782	CLARK	MANAGER	7839	9/06/1981

http://www.fredberinger.com/musings-on-nosql/



Column Database stores column values together

Architecture point of view

- Hybrid architecture might be suitable
  - "One size fits all"? ->Use right tool for right use case
- RDBMs for metadata and transactional processing
  - Even Twitter/Facebook still use MySQL for "small datasets".
    - Twitter for datasets < 1.5 TB
  - e.g. Constrained tree schema
    - Most DB schemas have tree structure
      - Store only data near root in RDBMS
- NoSQL for semistructured/unstructured/graph data
- Analytical for batch processing (patterns)

KeyValue - Memory cache

- Distributed non-persistance key/value with high performance (Distributed HashTable)
- Use cache to decrease load of DB (or any other expensive resource)
- Can help with consistency
- Can specify expiration or put/delete listeners

KeyValue - Redis

- REmote DIctionary Service
- Master->Slave (async)
  - Resends all modif commands to slaves
- It is often referred to as a **data structure server** since keys can contain strings, hashes, lists, sets and sorted sets.
- To use as a cache `maxmemory-policy allkeys-lru`
- Jedis Java API - so simple

```
Jedis j = new Jedis("localhost",6379);
j.set("name", "JohnDoe");
j.get("name");
```

Redis

- Data must fit to memory
- Write-write consistency guaranteed,write-read consistency eventual
- Always take care of our use case!
  - Show latest items in home page
  - Counters (number of access from IP)
  - Publish/Subsribe (keep map of requestors + SUBSCRIBE command)
  - Queues
  - Unique sets
  - Time-outing data
  - Cache, Transactions, Pipelining

Column family - BigTable

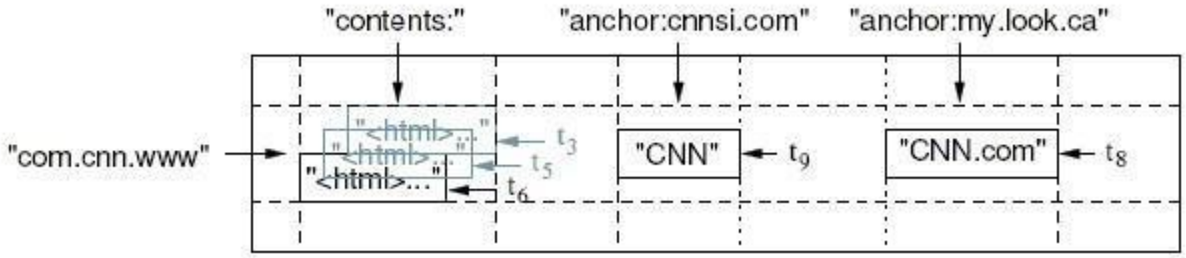
- Distributed multi-dimensional sorted map
- Fault tolerant
- Self managing
- Providing elasticity
- Use GFS for data storage

BigTable - Data model

(row: string, column: string, time: int64) -> string

- Lexicographical order by row key
- Nulls are skipped
- Easy to store 1:N (multivalue)
- Versioning of values with garbage collection
- Data stored in tablet (chunk of data+metadata)
- Column family

  - What columns should be stored nearby



GFS - Google File System

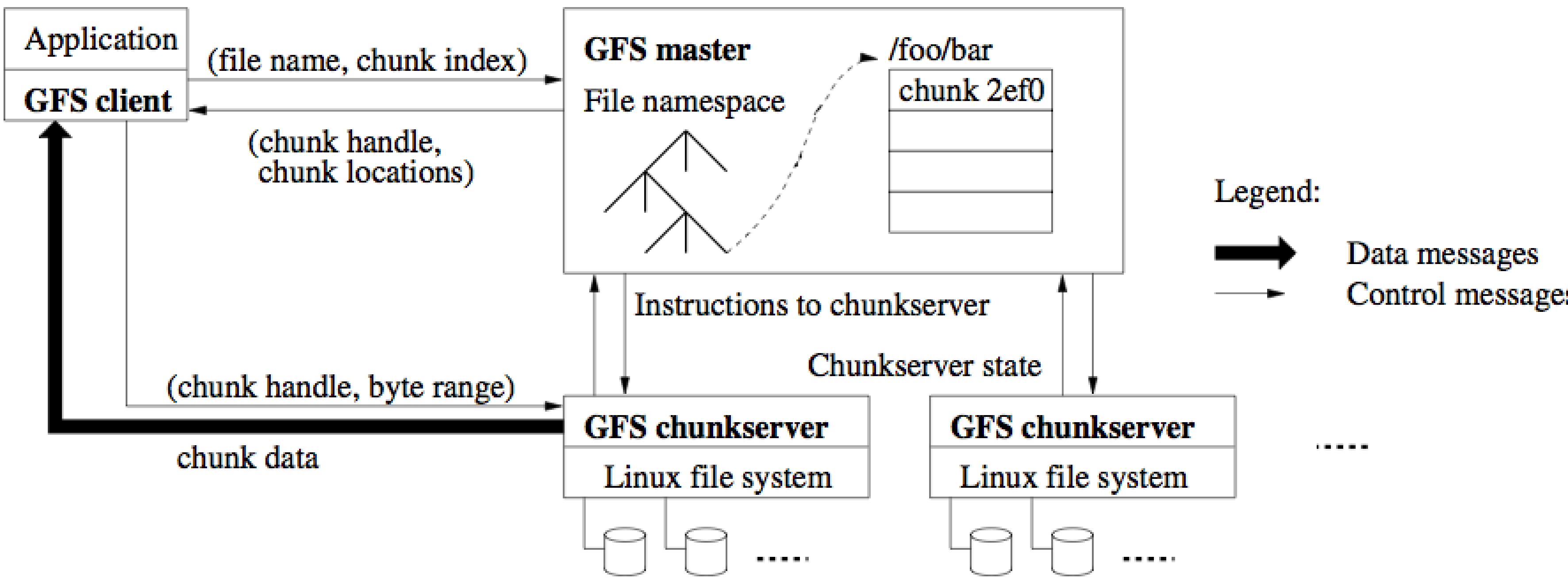


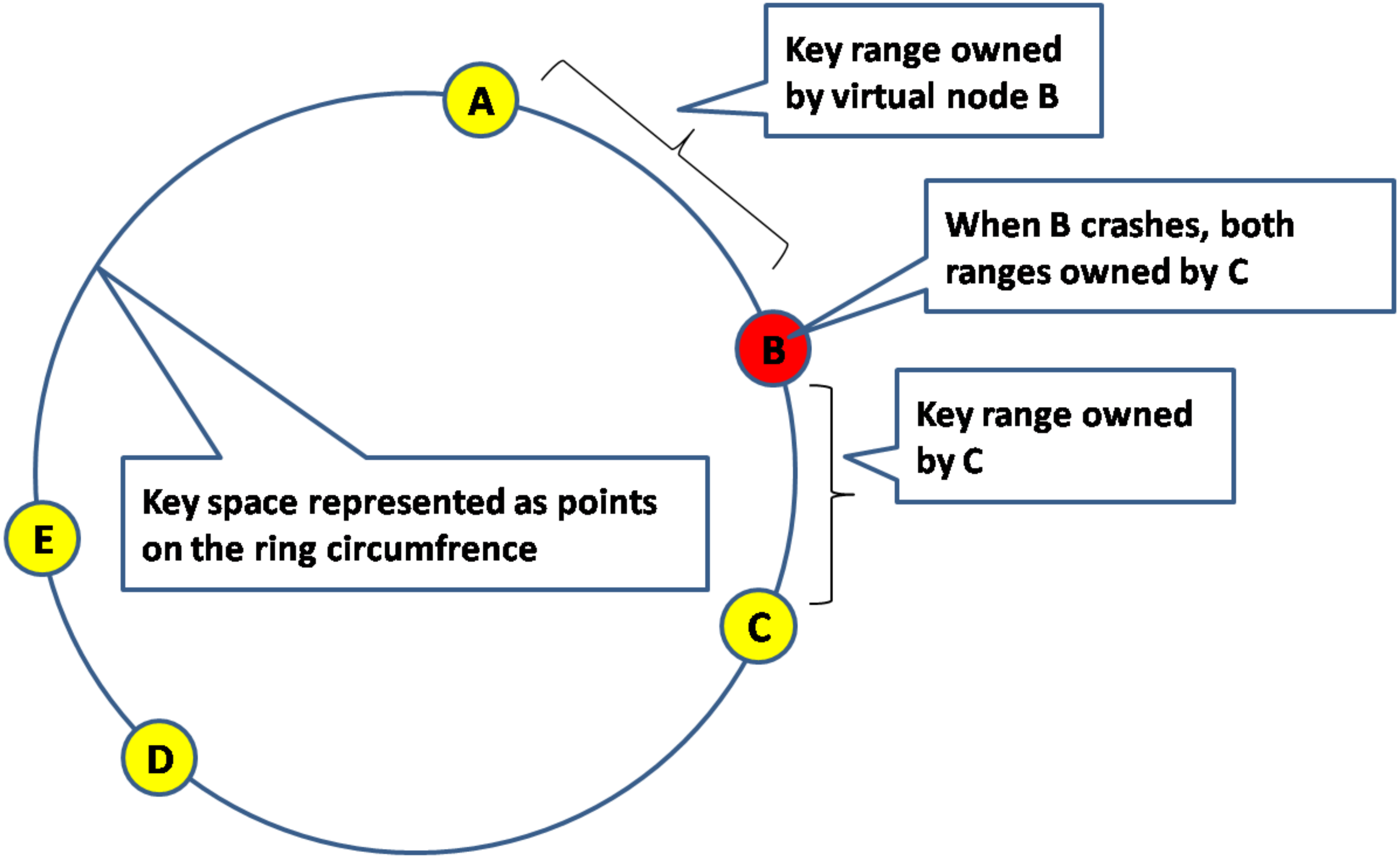
Figure 1: GFS Architecture

- PBs of data
- Master for medatada (can have hot stand by)
- Chunkserver for data (usually 64MB chunks)
- Write once read many times, heartbeat, replication

KeyValue/Document Riak

- Opensource written in Erlang
- No Master - All nodes equal
- Limited MapReduce
- Linear scalability
- Automatic recovery from node fail
- Fully distributed
  - Elasticity
- Fulltext
  - Solr, Lucene

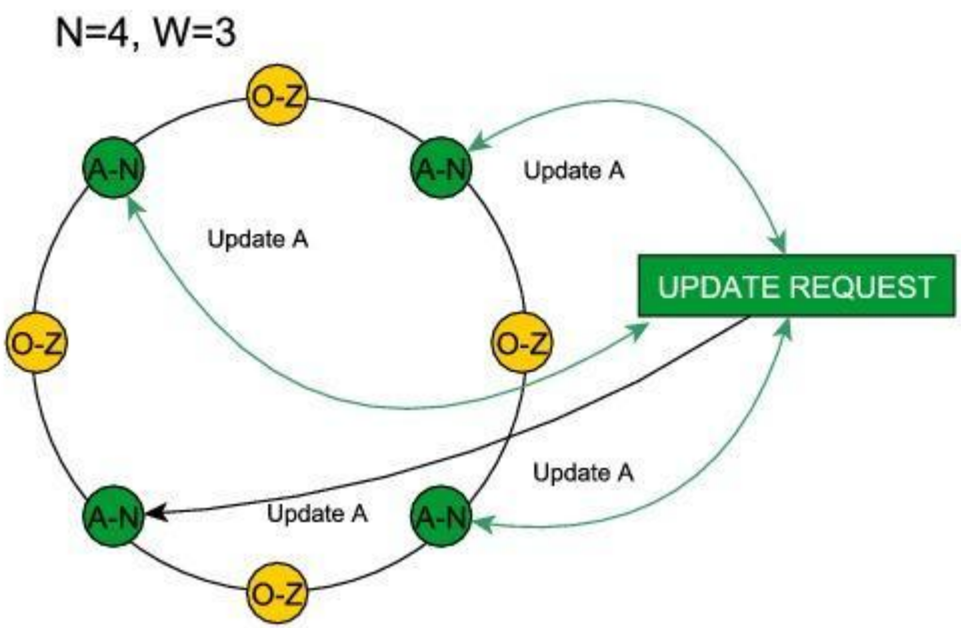
Consistent Hashing - SHA1



http://1.bp.blogspot.com/_j6mB7TMmJJY/SwohQZ9HTAI/AAAAAAAAAXM/X9CAGfpnL2o/s1600/p1.png

Quorum

- N = replication factor
- W nodes must respond before considered successful
- N/2 + 1 optimal



Riak - node failure

- Hinted handoff
  - neighboring node takes control over storage
  - After node recovery, data transferred to recovered node
- Read repair
  - When using quorum, if one node returns old data (using vector clock) of missing it will be repaired
    - This is done within clients query

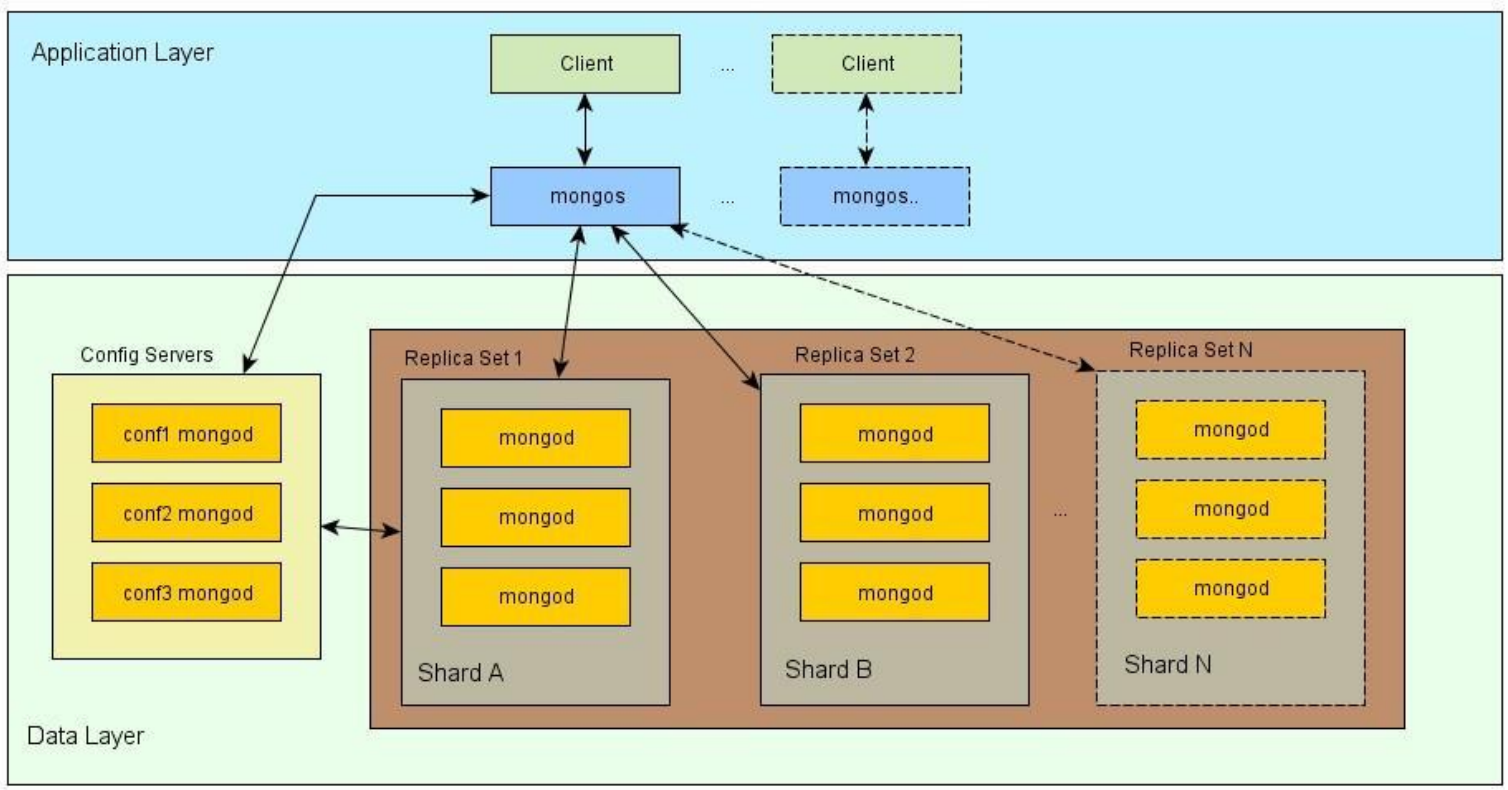
Document Database - MongoDB

- **Opensource in C++**
- **Document DB**
  - **JSON/BSON documents**
- **Master/Slave with dynamic voting**
- **Supports replication and sharding**
- **Support index**
  - Distributes it's across shard

Document Database - MongoDB

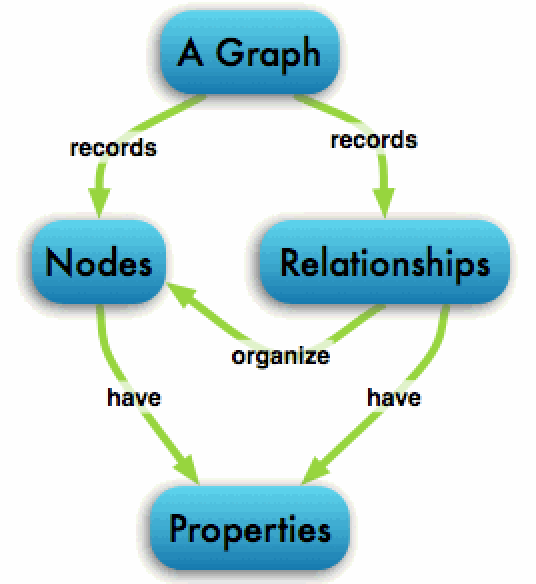
- Replica Set
  - Have one master serving all requests
  - In case of master failure new master is voted (the freshest)
- Sharding
  - Divide data and store them on different nodes (replica sets)
  - Data accessed together can be stored nearby
  - Can store data on right geographic location

MongoDB - Architecture

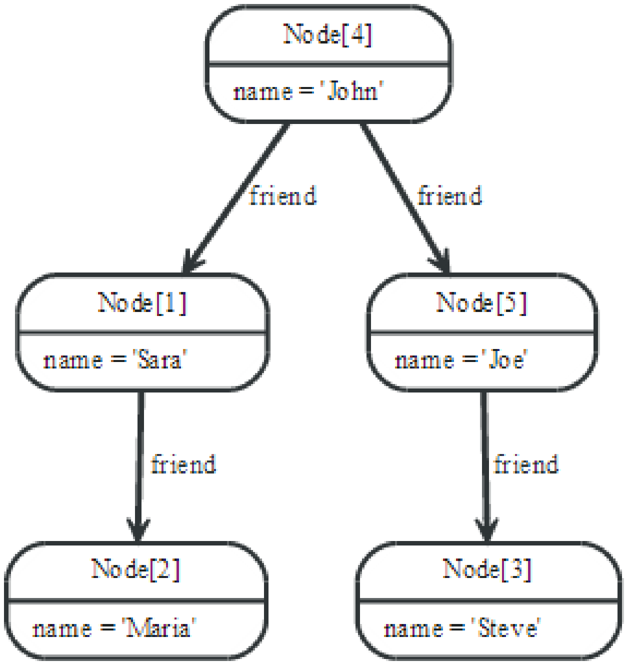


Graph DB - Neo4j

- Most generic structure
- Easy graph traversal
  - Good for queries: "Whom you might know'
  - Multiple traversals
  - much faster than JOIN
- Master/Slave
- ACID
- Bult-in algorithms
  - Dijkstra, A*, shortest paths,all paths, ...
- Cypher - declarative language



Cypher




```
START john=node:node_auto_index(name = 'John')
MATCH john-[:friend]->()-[:friend]->fof
RETURN john, fof
```

Resulting in:

john	fof
Node[4]{name:"John"}	Node[2]{name:"Maria"}
Node[4]{name:"John"}	Node[3]{name:"Steve"}
2 rows	
3 ms	

Hadoop - Analytical

- Open source Apache project
- Provides elasticity - scale from one to thousands nodes
- Based on Hadoop Distributed Filesystem
  - HDFS is open source implementation of GFS
- Map/Reduce framework
- Large scale database with simple programming model

Hadoop vs RDBMS

	RDBMS	Hadoop
Data sources	Structured with schema	(Un)structured
Data type	Records,objects, XML	Files
Language	SQL & XQuery	Pig, Hive, Jaql
Processing type	Quick resp., rand. access	Batch processing
Data integrity	Data loss is not acceptable	Data loss can happen sometime
History	~40 years of innovations	< 5 years old

Map/Reduce

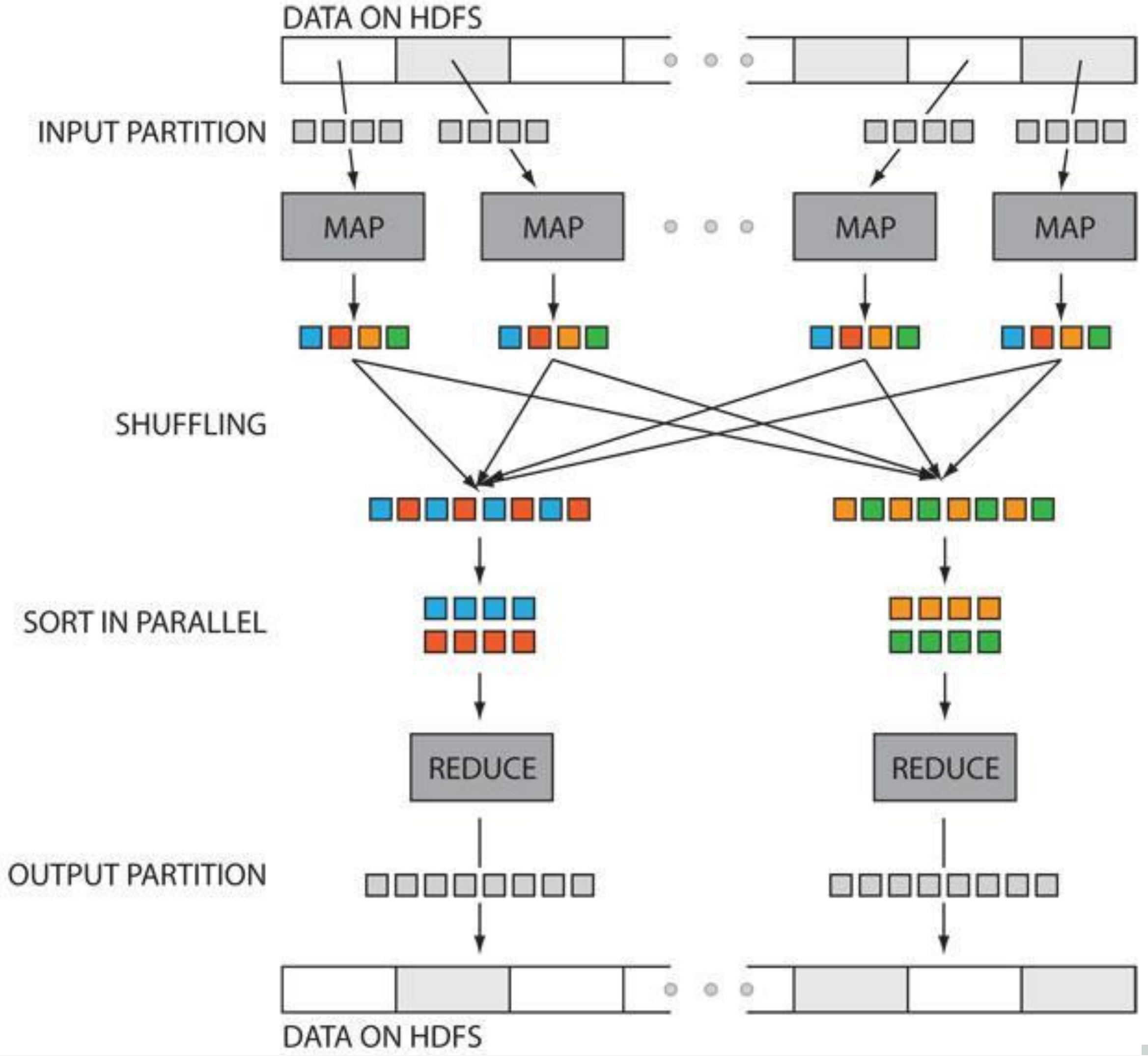
- Software framework for writing applications processing TBs+ datasets in parallel
- In reliable and fault tolerant manner
  - Use commodity hardware
- Forget taking care about:
  - parallel, semaphores, (dead) locks

Map & Reduce

- Map(k1,v1) → list(k2,v2)
- Reduce(k2, list (v2)) → list(v3)

```
function map(String name, String document):
    // name: document name
    // document: document contents
    for each word w in document:
      emit (w, 1)
   function reduce(String word, Iterator partialCounts):
    // word: a word
    // partialCounts: a list of aggregated partial counts
    sum = 0
    for each pc in partialCounts:
      sum += pc
    emit (word, sum)
```

Map & Reduce



DATA ON HDFS
INPUT PARTITION
MAP
MAP
MAP
MAP
SHUFFLING
SORT IN PARALLEL
REDUCE
REDUCE
OUTPUT PARTITION
DATA ON HDFS

Pig

- Goal: Reduce program size and complexity
- Data flow language
- Sample:

```
input = LOAD './all_web_pages' AS (line:chararray);
words = FOREACH input GENERATE FLATTEN(TOKENIZE(line)) AS word;
word_groups = GROUP words BY word;
word_count = FOREACH word_groups GENERATE COUNT(words) AS count, group;
ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO './word_count_result';
```

Hive

- Declarative
- Sample:

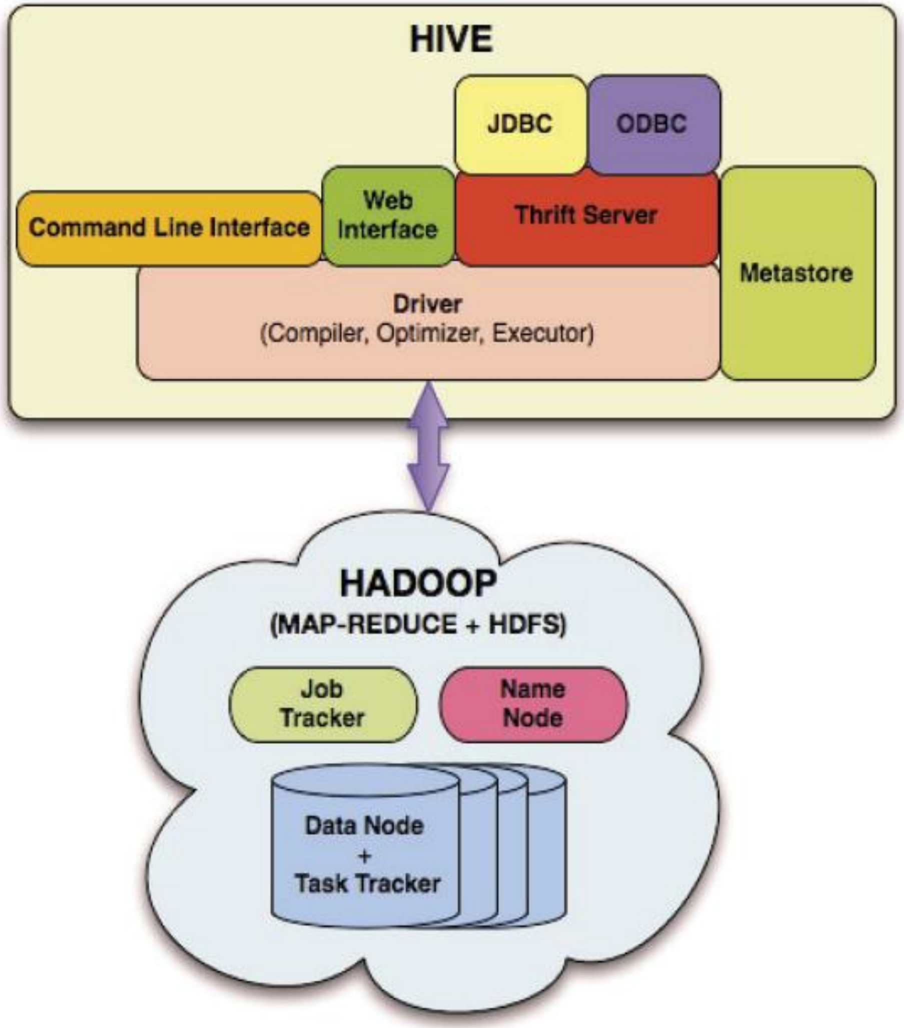
CREATE TABLE movie_ratings

(userid INT,movieid INT,rating INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE;



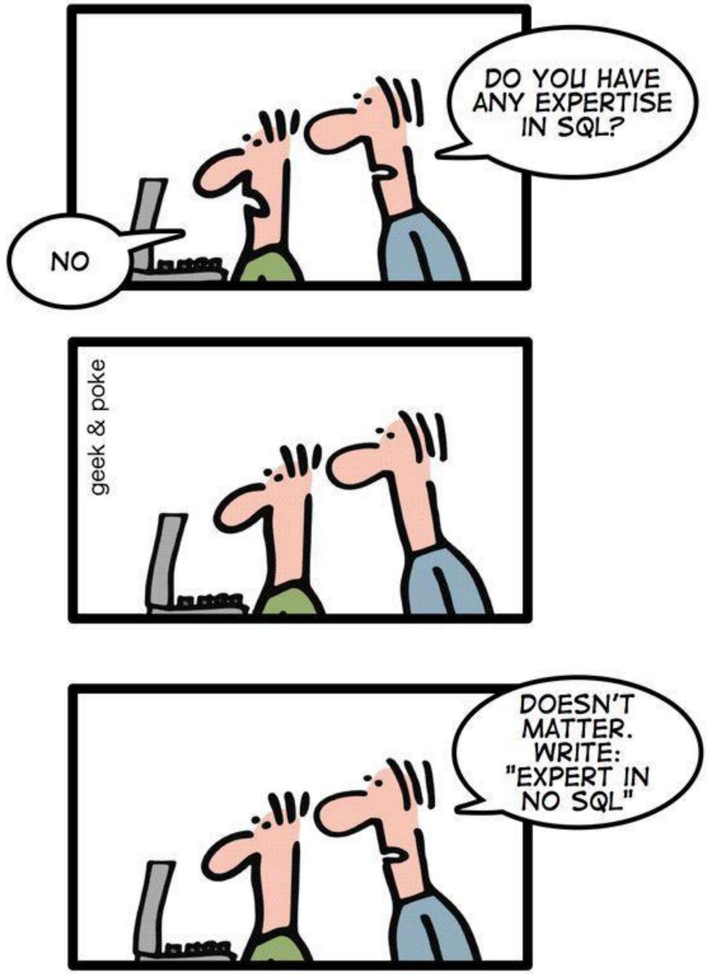
Use case

- IBM Jeopardy (Get answer to question before others)
- Use Hadoop to load large number of data and find the answer
- 200M pages loaded to memory

Use case - Facebook mail system

- HBase with HDFS (open source GFS)
  - High write throughput
  - Good random read performance
  - Small messages and metadata
  - Search index
- Stats:
  - 8B+ messages/day
  - Peak 1.5M ops/s (55% read, 46 write)
  - +250TB/month
- Two schema changes while in production

HOW TO WRITE A CV



Leverage the NoSQL boom