My experience in data management systems and public engagement activities

Saumya Bhatnagar

March 5, 2020

Why DBMS!

Users

DBA	APP PROGRAMMERS	END USERS
DB Schema	App Software	Query App Interface

DBMS

Query Processor	Query Evaluation Engine (DDL Interpreter, DML Compiler, Application Object Code)
Storage Manager	Buffer Manager, File Manager, Transaction Manager

Database

Data files, Data Dictionaries, Indices

DBMS Types

	SQL	NoSQL
High Level Model	ER Model	
Representational Model	Hierarchical (IMS), Relational (Oracle, DB2, SQL Server), Network (IDMS, IMAGE)	
Low-Level Model		

DB Architectures

- Centralized DBMS Architecture
- Client-Server Architecture
- Distributed Database Architecture

Schema Types

- Internal Schema
- Conceptual Schema
- External Schema

Glossary on Keys

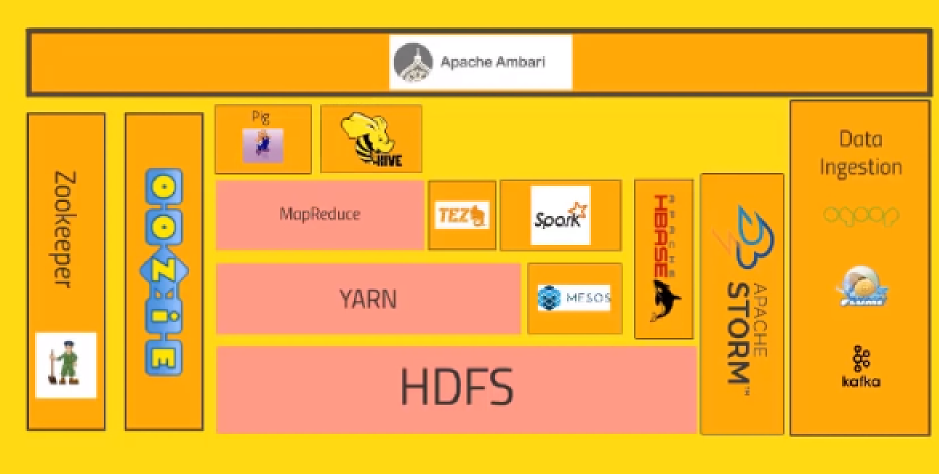
Key Types

- Super Key
- Candidate Key
- Primary Key
- Secondary Key
- Foreign Key
- Composite Key
- Compound Key (Composite key with foreign key)
- Alternate Key
- Sort/Control Key
- Surrogate key

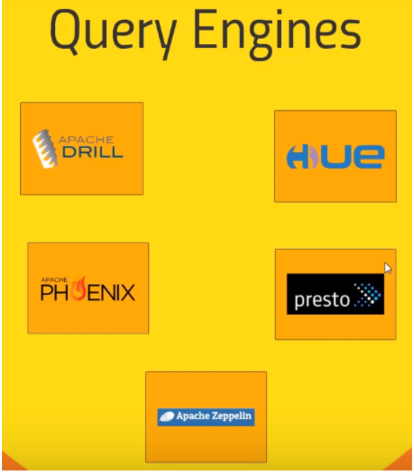
Overview

1. DBMS
   - MySql, Oracle, Cassandra, HBase, MongoDB
2. Hadoop
   - Hadoop Ecosystem
   - External Data Storages
   - Query Engines
3. Which Data Storage?
4. SQL
   - MySql, Vertica
5. NoSQL
   - Cassandra with solr
   - No one single point of failure
6. APIs
7. Microservices

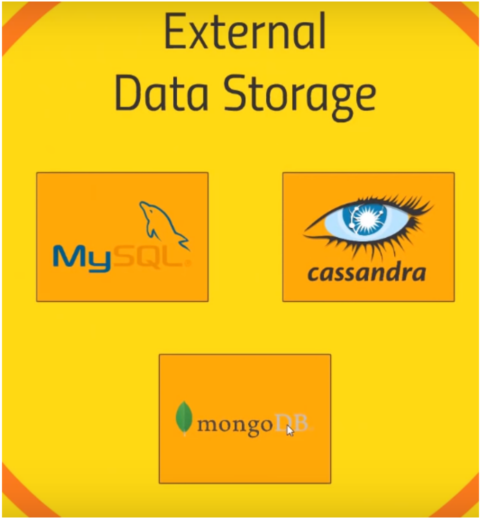
Hadoop Ecosystem



Query Engines And External data storage



Query Engines



External Data Storage

Clustered Computing Platforms (Mapreduce, Spark)

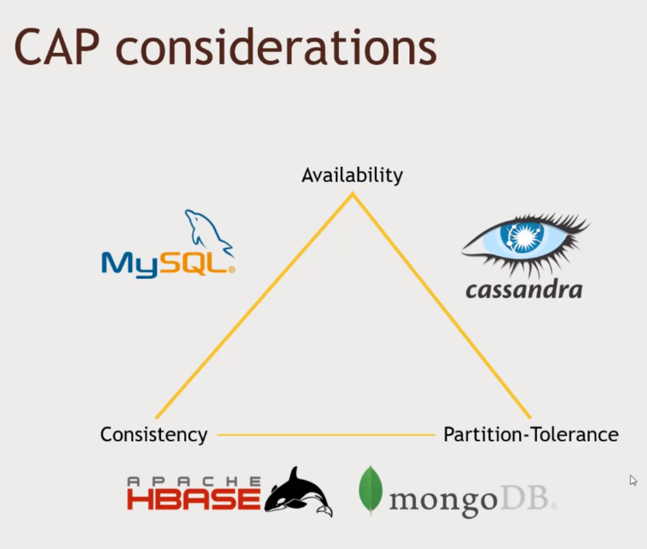
SPARK

- Distributing queries and trend analysis
- Microbatching for historical analysis
- Loading large datasets into memory
- Running queries against large datasets

Pros & Cons of the databases

Hadoop/Mapreduce	Slow for real time analytics
MongoDB	Global write lock performance concerns
Cassandra (w/o solr)	Query Limitations
Cassandra (w/o solr)	No bother about denormalizing, duplication, access pattern data modelling
Solr	Search capabilities, partial text search, facet queries, geospatial, etc.

Which Data Storage?



CAP considerations

Availability

Consistency

Partition-Tolerance

Vertica for Big Data Engineering

Command Type	
1. DDL	1. create, alter, drop, truncate, rename
2. DML	2. select, insert, update, delete
3. DCL	3. grant, revoke
4. TCL	4. commit, rollback

Example (Vertica Code Example)

```
SELECT name, class, date,
RANK() OVER (PARTITION BY class ORDER BY marks desc) AS rank
FROM student
WHERE name IS NOT NULL
AND subject like ’math%’
AND date > ’01/01/2007’
ORDER BY class;
```

SQL Glossary

- bandwidth=rate of data transfer
- latency=time of date transfer
- 1NF, NF, 3NF, BCNF
- ACID Properties (atomicity, consistency, isolation, durability)
- Lossless Decomposition
- Data Independence
-
-
-
-
-
-

DSE provides integration between Cassandra with Solr

- Storage grid (cassandra) + Search grid(solr)
- Devcenter or cqlsh
- Cassandra cluster handling over 1TB data
- 2 Data Centers
- 3 Servers, with RF of 3
- configure dse.yaml or vassandra.yaml
- Opscenter
- Solr Admin UI gives Solr Index Size
- All Nodes should have solr enabled within DC
- Map collection to dynamic fields
- solr queries have consistency levels

CQL syntax similar to SQL syntax

Example (CQL Code Example)

```
/*create table defining partition, clustering keys*/
CREATE TABLE student (
name text, class text, subject text, date timestamp,
PRIMARY KEY ((name, class), date)
);
```

Primary key is defined as ((partition keys), clustering/sorting keys)

Example (CQL Code Example)

```
SELECT name, class, date, rank FROM student
WHERE name IS NOT NULL
AND subject CONTAINS ’math’
AND date > ’01/01/2007’
ORDER BY class
PER PARTITION LIMIT 2;
```

Solr provides full text search, term-search

Example (CQL + Solr Code Example)

```
SELECT name, class, date
FROM student WHERE
solr_query=’{"q":"name:[* TO *] AND subject:math*",
             "fq":"date:[2007-01-01T00:00:00Z TO NOW]",
             "facet":{"field":"class"},
             "sort":"class, marks desc",
             "paging":"driver",
             "timeAllowed":30000 }’
ALLOW FILTERING;
```

Clustering columns can be defined in WHERE clauses if ALLOW FILTERING is also used even if a secondary index is not created

Cassandra Glossary

- snitch
- Gossip
- Quorum
- num_tokens
- max_solr_concurrency_per_core = cpu code / num solr cores
- partitioner
- auto_bootstrap
-
-
-
-
-

SOAP vs REST

Client (Machine Devices - Mobile, desktop) → API Binding → Server
SOAP:

- Stateless
- Slow
- XML

REST:

- Public
- Fast
- Multiple formats

REST:
NODE.js
MongoDB (native js code) - JS based
json format
MongoDB: 2 collection joins, aggregation in mongoDB
instead js for loop can be used

REST vs Bulk

Bulk is built on top of REST
Bulk:

- async
- batches

Email API

Email uses SMTP and Port number
Tight coupling
IOC (Inversion of Control):
Inject email in customer: 1. Property in class 2. Parameter

CQRS

command

...

read

...

API gateway

Swagger
APIGEE:

- authentication control
- traffic control

Server info... API gateway provides URL

Microservices

Microservices architecture runs on top of STORM/JMS/KAFKA
Storm (handles clustering/distribution)
Kafka (messaging between the grids)
Kafka or Rabbit NQ are message broker URIs
for cache use Redis. Redis is a cache DB
JWT (Json Web token): network calls to DB should be least → Resource Management
YAML → dependent on other services. has details such as name, port, URL, env variables, etc.

Docker - Container

Docker is OS
Containers are VM ware
Cluster has nodes. Nodes has pods. e.g. Pod1, Pod2, Pod3, Pod4 are 4 containers. Pod1 may act as Inst of Service
Dockerfile is image of service and is "Built, deployed and ran" by DevOps

Domain Driven Design

Service bus
Rabbit MQ
Order Service & Domain Service
DDD: Command (message) $\rightarrow$ Event [Eventual Consistency]
Service bus... message sent to exchange queue via routing key

Pub/Sub Design Pattern

content...

title

content...

Microservices on Docker

content...

Microservices on Kubernetes

content...

Serverless

content...

Thank You!