Preparing Your Data For Cloud

Narinder Kumar

Inphina Technologies

inphina

IndicThreads.com Conference On

Upcoming Technology

2010: Cloud Computing

Pune, India

Agenda

- Relational DBMS's : Pros & Cons
- Non-Relational DBMS's : Pros & Cons
- Types of Non-Relational DBMS's
- Current Market State
- Applicability of Different Data-Bases in different environments

Relational DBMS - *Pros*

- Data Integrity
- ACID Capabilities
- High Level Query Model
- Data Normalization
- Data Independence

Relational DBMS - *Cons*

- Scaling Issues
  - By Duplication (Master-Slave)
  - By Sharding/Division (Not transparent)
- Fixed Schema
- Mostly disk-oriented (Performance)
- May fair poorly with large data

Non-Relational DBMS - *Pros*

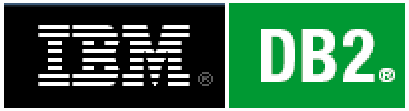
- Scalability
- Replication / Availability
- Performance
- Deployment Flexibility
- Modelling Flexibility
- Faster Development *(?)*

Non-Relational DBMS - *Cons*

- Lack of Transactional Support
- Data Integrity is Application's responsibility
- Data Duplication / Application Dependent
- Eventually Consistent *(mostly)*
- No Standardization
- New Technology

RDBMS's and Cloud

Cloud Capable RDBMS



Almost every RDBMS can run in a IAAS Cloud Platform

Cloud Native RDBMS

Types of Non-Relational DBMS

- Key Value Stores
- Document Stores
- Column Stores
- Graph Stores

Key Value Data-Bases

- Object is completely Opaque to DB
- Mostly GET, PUT & DELETE operations are supported
- There may be limits on size of Objects

Inspired by Amazon Dynamo Paper

Key Value DataBases & Cloud

amazon web services™
Amazon S3
Simple Storage Service

Project Voldemort

Redis

scalaris

MemCachedDB

Tokyo Tyrant

Document Data-Bases

- Object is not completely opaque to DB
- Every Object has it's own schema

    FirstName="Bob", Address="5 Oak St.", Hobby="sailing".

    FirstName="Jonathan", Children=("Michael,10", "Jennifer,8")

- Can perform queries based on Object's attributes
- Possible to describe relationships between Objects
- Joins and Transactions are not supported
- Good for XML or JSON objects

Document Data-Bases & Cloud

Column-Store Data-Bases

- Richer than Document Stores
- Multi-Dimensional Map
  - Tables
    - Row
    - Column
    - Time-Stamp
- Supports Multiple Data Types
- Usually use an Underlying DFS

Inspired by Google Big Table Paper

Column-Store Data-Bases & Cloud

Key Factors while Making a Choice

- Application Architecture Requirements
- Platform choices
- Non-Functional Requirements
  - Consistency
  - Availability
  - Partition
  - Security
  - Data Redemption

Different Requirements = Different Solutions

Scenario 1

- Feature First
  - Corporate Data
  - Consistency Requirements
  - Business Intelligence
  - Legacy Application

RDBMS on Amazon Cloud, RackSpace (IaaS) *or*

Microsoft Azure/Amazon RDS (PaaS)

Scenario 2

- Consumer Facing Application
  - Big Files (Images, BLOB's, Files)
  - Geographically Distributed
  - Mostly writes
  - Not heavy requirement on Rich Queries

Key-Value Data Stores (Amazon S3, Project Voldemort, Redis)

Scenario 3

- Hundreds Of Government Documents with different schemas
  - Need to serve on Web
  - Data Mining

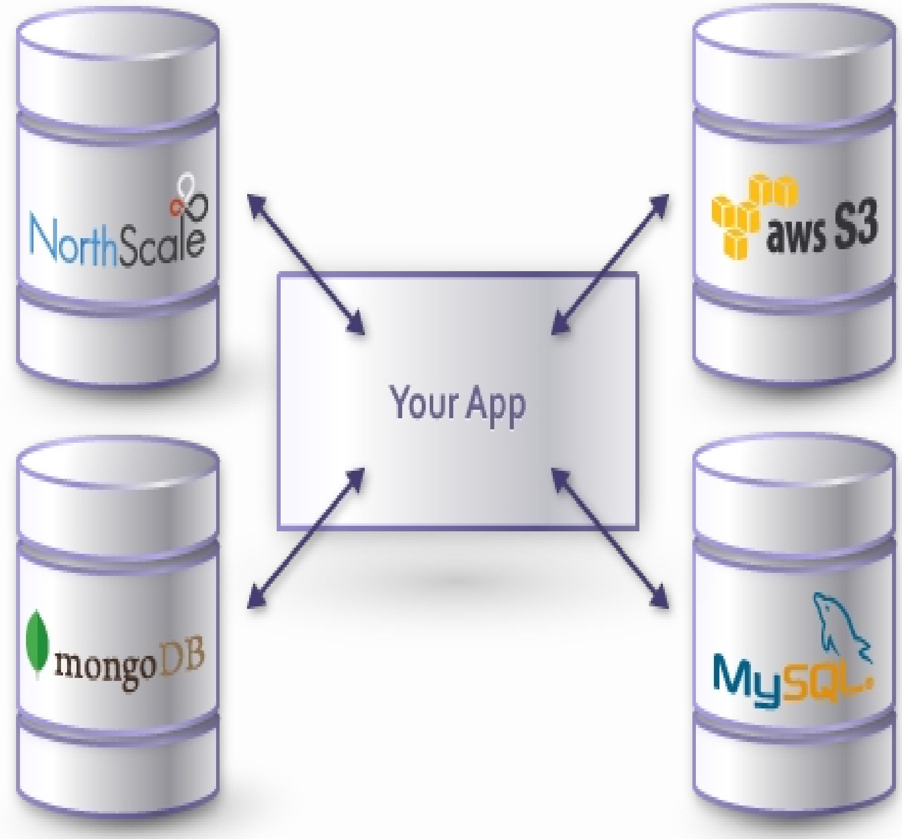
Document Data-Stores (Amazon SimpleDB, Apache Couche DB, MongoDB)

Scenario 4

- Scale First
  - Huge Data-Set
  - Analytical Requirements
  - Consumer Facing
  - High Availability over Consistency

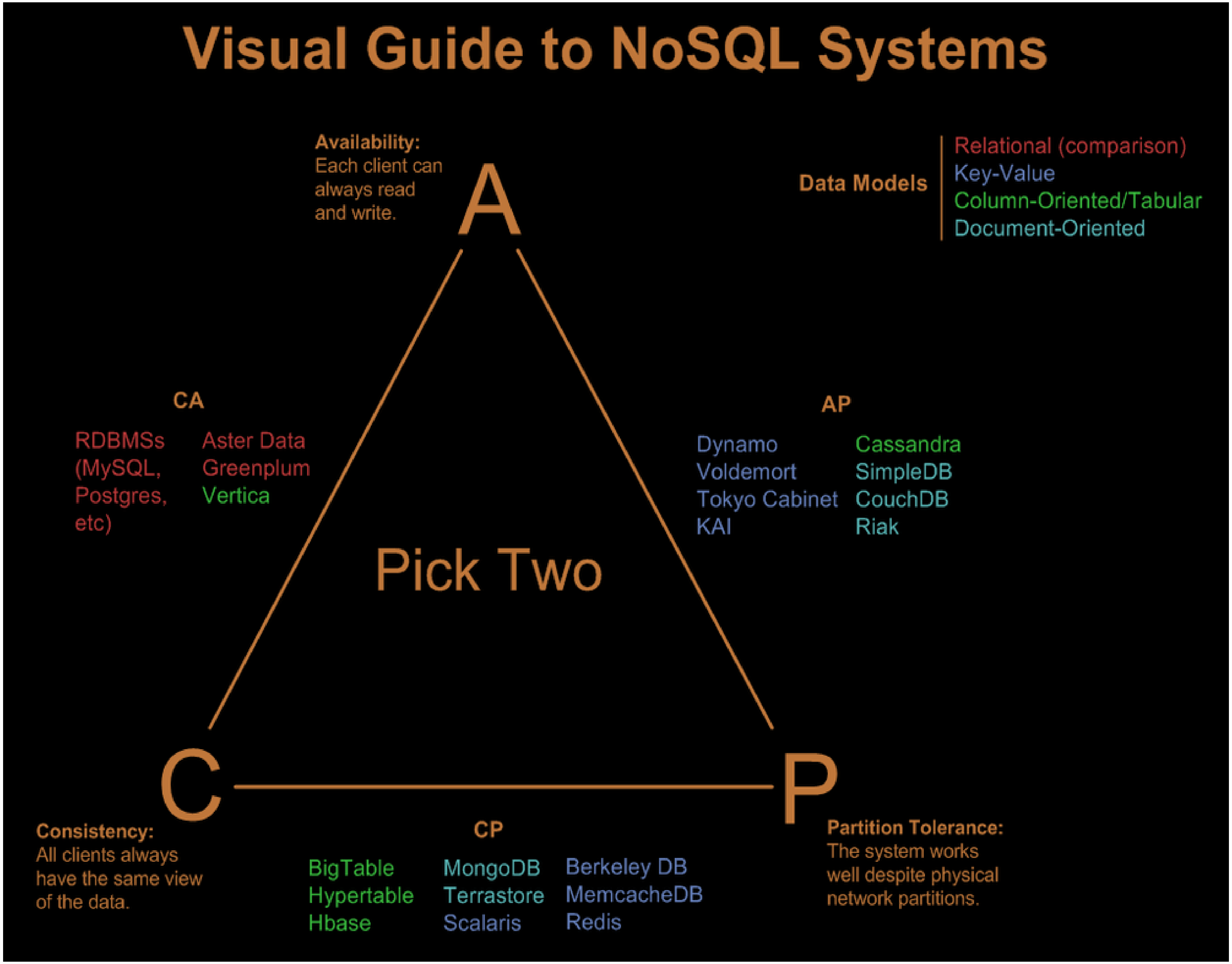
Column Data-Stores (Google App Engine, Hbase, Cassandra)

Mix & Match of Earlier Scenarios



- Polyglot Persistence
  - RDBMS for low-volume and high value
  - Key-Value DB for large files with little queries
  - Memcached DB for short-lived Data
  - Column DB for Analytics

CAP Theorem



Visual Guide to NoSQL Systems

Availability:
Each client can always read and write.

Data Models
- Relational (comparison)
- Key-Value
- Column-Oriented/Tabular
- Document-Oriented

CA

RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica

AP

Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak

Pick Two

Consistency:
All clients always have the same view of the data.

CP

BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis

Partition Tolerance:
The system works well despite physical network partitions.

Conclusions

- One Size does not Fit all
- Many choices
- No-SQL DB's providing Alternatives
- RDBMS serve useful purpose

inphina

nkumar@inphina.com

www.inphina.com
http://thoughts.inphina.com

References

- http://nosql-database.org/
- http://www.drdobbs.com/database/218900502
- http://perspectives.mvdirona.com/2009/11/03/OneSizeDoesNotFitAll.aspx
- http://blog.nahurst.com/visual-guide-to-nosql-systems
- http://blog.heroku.com/archives/2010/7/20/nosql/
- http://www.vineetgupta.com/2010/01/nosql-databases-part-1-landscape.html
- http://project-voldemort.com/
- http://code.google.com/p/redis/
- http://memcachedb.org/
- http://aws.amazon.com/simpledb/
- http://couchdb.apache.org/
- http://www.mongodb.org
- http://riak.basho.com/