ENABLING EXTREME SCALABILITY WITH NOSQL

An introduction to NoSQL databases

Presented By: Ashwani Kumar
B.Tech. (I.T.) 3rd year
1303213015
What is covered in this presentation?

- A brief history of databases
- NoSQL WHY, WHAT & WHEN?
- Characteristics of NoSQL databases
- Aggregate data models
- CAP theorem
Database - Organized collection of data

DBMS - Database Management System: a software package with computer programs that controls the creation, maintenance and use of a database

Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information.
A brief history

- Codd's Relational Model
- 1977: IBM's System R
- Oracle
- Postgres
- Teradata ITB
- Sybase IQ
- DB2 Parallel Edition
- Greenplum
- Netezza
- Aster Data
- Vertica
- Oracle Exadata
- Explosion of DW's
- VoltDB
- MapReduce Paper
- BigTable Paper
- Dynamo Paper
- NoSQL Term
- Hadoop
- Big Three: CouchDB, MongoDB, HBase (many more smaller)
• **Benefits of Relational databases:**

- Designed for all purposes
- ACID
- Strong consistancy, concurrency, recovery
- Mathematical background
- Standard Query language (SQL)
- Lots of tools to use with i.e: Reporting services, entity frameworks, ...
SQL databases
But...
- Relational databases were not built for distributed applications.

Because...
- Joins are expensive
- Hard to scale horizontally
- Impedance mismatch occurs
- Expensive (product cost, hardware, Maintenance)

Era of Distributed Computing
But...

- Relational databases were not built for **distributed applications**.

Because...

- Joins are expensive
- Hard to scale horizontally
- Impedance mismatch occurs
- Expensive (product cost, hardware, Maintenance)

And....

It’s weak in:

- Speed (performance)
- High availability
- Partition tolerance
Why NOSQL now??  Ans. Driving Trends

New Trends

- Big data
- Connectivity
- P2P Knowledge
- Concurrency
- Diversity
- Cloud-Grid
Side note: RDBMS performance
But.. What’s NoSQL?

- A No SQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational database

- No SQL systems are also referred to as "NotonlySQL“ to emphasize that they do in fact allow SQL-like query languages to be used.
NoSQL avoids:

- Overhead of ACID transactions
- Complexity of SQL query
- Burden of up-front schema design
- DBA presence
- Transactions (It should be handled at the application layer)

Provides:

- Easy and frequent changes to DB
- Fast development
- Large data volumes (e.g., Google)
- Schema less
NoSQL why, what and when?

When and when not to use it?

**WHEN / WHY ?**
- When traditional RDBMS model is too restrictive (flexible schema)
- When ACID support is not "really" needed
- Object-to-Relational (O/R) impedance
- Because RDBMS is neither distributed nor scalable by nature
- Logging data from distributed sources
- Storing Events / temporal data
- Temporary Data (Shopping Carts / Wish lists / Session Data)
- Data which requires flexible schema
- **Polyglot Persistence** i.e. best data store depending on nature of data.

**WHEN NOT ?**
- Financial Data
- Data requiring strict ACID compliance
- Business Critical Data
NoSQL is getting more & more popular
In relational Databases:

- You can’t add a record which does not fit the schema
- You need to add NULLs to unused items in a row
- We should consider the datatypes. i.e.: you can’t add a string to an integer field
- You can’t add multiple items in a field (You should create another table: primary-key, foreign key, joins, normalization, ... !!!)
In NoSQL Databases:

- There is no schema to consider
- There is no unused cell
- There is no datatype (implicit)
- Most of considerations are done in application layer
- We gather all items in an aggregate (document)
NoSQL databases are classified in four major datamodels:

- Key-value
- Document
- Column family
- Graph

Each DB has its own query language
Simplest NOSQL databases

The main idea is the use of a hash table

Access data (values) by strings called keys

Data has no required format data may have any format

Data model: (key, value) pairs

Basic Operations:
Insert(key, value), Fetch(key), Update(key), Delete(key)
The column is lowest/smallest instance of data.

It is a tuple that contains a name, a value and a timestamp.
Some statistics about Facebook Search (using Cassandra)

- MySQL > 50 GB Data
  - Writes Average: ~300 ms
  - Reads Average: ~350 ms

- Rewritten with Cassandra > 50 GB Data
  - Writes Average: 0.12 ms
  - Reads Average: 15 ms
Graph data model

- Based on Graph Theory.
- Scale vertically, no clustering.
- You can use graph algorithms easily
- Transactions
- ACID
Document based data model

- Pair each key with complex data structure known as data structure.
- Indexes are done via B-Trees.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
Document based data model

SELECT name, pic, profile_url
FROM user
WHERE uid = me()

SELECT message, attachment
FROM stream
WHERE source_id = me() AND type = 30

SELECT name
FROM friendlist
WHERE owner = me()

SELECT name
FROM group
WHERE gid IN ( SELECT gid
FROM group_member
WHERE uid = me() )

SELECT name, pic
FROM user
WHERE online_presence = "active"
AND uid IN ( SELECT uid2
FROM friend
WHERE uid1 = me() )

https://developers.facebook.com/docs/reference/fql/
## SQL vs NOSQL

### Differences

<table>
<thead>
<tr>
<th></th>
<th>SQL Databases</th>
<th>No SQL Database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Oracle, mysql</td>
<td>Mondo DB, CouchDB, Neo4J</td>
</tr>
<tr>
<td><strong>Storage Model</strong></td>
<td>Rows and tables</td>
<td>Key-value. Data stored as single document in JSON, XML</td>
</tr>
<tr>
<td><strong>Schemas</strong></td>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>Vertical &amp; Horizontal</td>
<td>Horizontal</td>
</tr>
<tr>
<td><strong>Transactions</strong></td>
<td>Yes</td>
<td>Certain levels</td>
</tr>
<tr>
<td><strong>Data Manipulation</strong></td>
<td>Select, Insert, Update</td>
<td>Through Object Oriented API's</td>
</tr>
</tbody>
</table>
• We need a distributed database system having such features:
  • – Fault tolerance
  • – High availability
  • – Consistency
  • – Scalability

Which is impossible!!!
According to CAP theorem
The CAP Theorem

- Impossible for any shared data-system to guarantee simultaneously all of the following three properties:
  - Consistency – once data is written, all future read requests will contain that data
  - Availability – the database is always available and responsive
  - Partition Tolerance – if part of the database is unavailable, other parts are unaffected

We can not achieve all the three items in distributed database systems (center)
CAP theorem

Traditional vs. NoSQL

Relational

Consistency

Partition Tolerance

CAP Theorem

Availability

NoSQL
In Conclusion!

• RDBMS is a great tool for solving ACID problems
  • When data validity is super important
  • When you need to support dynamic queries
• NoSQL is a great tool for solving data availability problems
  • When it’s more important to have fast data than right data
  • When you need to scale based on changing requirements
• Pick the right tool for the job
References..

- nosql-database.org/
- https://www.mongodb.com/nosql-explained
- www.couchbase.com/nosql-resources/what-is-no-sql
- http://nosql-database.org/ "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable"
- NoSQL distilled, Martin Fowler
- Please like and follow at www.slideshare.net/AshwaniKumar274
Thanks...

Any Questions??