NoSQL Databases

Recap

Data Management: Trends & Requirement

- Volume of data, requires:
  - Database scalability
  - massive data distribution
- Velocity of data, requires:
  - frequent update operation
- Variety of data, requires:
  - flexible database schema
- Big Users, requires:
  - massive read throughput

NoSQL

- a movement for finding an alternative to solve problems that RDBMS is not able to solve
- NoSQL databases is
  - not using relational model
  - designed to run on clusters
  - scale horizontally
  - No Schema
    - fields can be added easily at anytime
  - easy replication support

Types of NoSQL databases

- Key-value stores
  - Data Model: Key-value
  - Examples: Redis, Amazon DynamoDB, RocksDB
- Document stores
  - Data Model: Document such as XML or JSON
  - Examples: MongoDB, CouchDB

Types of NoSQL databases

- Column oriented databases
  - Data model: rows that are associated with multiple columns which can be grouped in families
  - Examples: BigTable, Hbase, Cassandra
- Graph databases
  - Data Model: entities and relationships between them
  - Examples: Apache Giraph, Stardog

End of RDBMS?

- Relational databases are not going anyway
    - are still good fit and ideal for structured, mature , reliable data
- Polyglot persistence = Usually different databases are used in different circumstances
- Two trends
    - NoSQL implements RDBMS standards
    - RDBMS are adopting NoSQL principles

Facebook: Database Tech. Behind

- Apache Hadoop http://hadoop.apache.org/
  - Hadoop File System (HDFS)
  - MapReduce for batch processing
- Apache Hive http://hive.apache.org/
  - SQL-like access to Hadoop-stored data

Facebook: Database Tech. Behind

- Apache HBase http://hbase.apache.org/
    - a Hadoop column-family database
    - used for e-mails, and SMS
- Memcached http://memcached.org/
    - distributed key-value store
    - used as a cache between web servers

      and MySQL servers in the beginning of FB

Facebook: Database Tech. Behind

- Apache Giraph http://giraph.apache.org/
  - graph database
  - facebook users and connections is one very large graph
  - used since 2013 for various analytic tasks (trillion edges)
- RocksDB http://rocksdb.org/
  - high-performance key-value store
  - developed internally in FB, now open-source

NoSQL Databases

Principles

Different Aspect of Data Distribution

- Scaling
  - vertical vs. horizontal
- Distribution model
  - sharding
  - replication
- CAP properties
  - Consistency, Availability, and Partition tolerance

Data Model

- The model represent the way by which the system organizes the data

- Each noSQL DB has a different data model

  - different noSQL: key-value, document, column-family, graph

  - key-value, document, column-family are oriented on **Aggregates**

Aggregates

- An aggregate
    - is a data unit with complex structure
        - not simply a tuple (row) as in RDBMS
    - a collection of related objects treated as a unit
        - a unit for data manipulation & management

NoSQL databases: Aggregate Oriented

- Many noSQL databases are aggregate-oriented
    - there is no general strategy on how to set the aggregate boundaries
    - but the aggregate give the database information about which bits of data to be manipulated together
        - which data to be stored together (on the same node for example)
    - this will minimize number of nodes to be access at read time
    - impact on concurrency control
        - atomic manipulation of a single aggregate at a time

Scalability

- is the capability of the system to handle growing amount of data and/or queries without loosing performance
- it is potential to be enlarged in order to accommodate the growth
- Two general approaches
    - Vertical
    - Horizontal

Vertical Scaling

- Scaling up/down
  - adding resources to a single node
    - such as increasing number of CPUs, extending the memory, using larger disk storage
  - using larger and more powerful machines
- Traditional choice:
  - favor of strong consistency
  - easy to implement
  - don’t deal with issues related to data distribution
- works well in many cases but …

Vertical Scalability: drawbacks

- Performance limit
  - everything works fine until we reach the limits of the node
- Cost
  - the cost is higher than the sum of the cost of equivalent commodity machines
- Proactive provisioning
  - at the beginning, applications have no idea about the final scale
  - upfront budget is needed when deploying new machines
  - flexibility is not supported

Vertical Scalability: drawbacks

- vendor lock-in
  - producers of large machines are limited
  - which will make customers dependent on the vendors
- deployment downtime
  - to scale up, it is not possible to do without turning off (downtime)

Horizontal Scalability

- Scaling out/in
  - adding more nodes to the system
  - the system is running on multiple nodes, adding/removing nodes is easy
- this is the choice for many NoSQL databases
- Advantages:
  - commodity hardware; cost effective
  - flexible deployment
  - no single point of failure

Horizontal Scalability: False Assumptions

- Network is reliable
- latency is zero
- Bandwidth is infinite
- Network is secure
- transport cost is zero

Horizontal Scalability: Consequences

- Increases complexity of management
- introduces new issues
  - Synchronization
  - data distribution
  - data consistency
  - recovery from failure

Horizontal Scalability: Architecture

- runs on a cluster
- cluster consists of:
  - a collection of interconnected commodity machines
  - based on shared-nothing architecture
    - each node has its own CPU, memory, disk storage
    - each node runs its own operating system
- data, queries, workload is distributed among the nodes