NoSQL

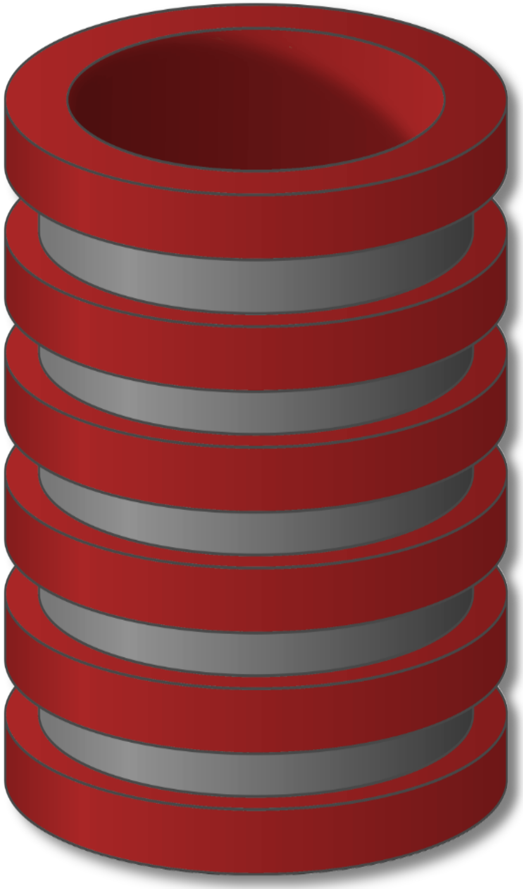
00

Presented by Subhash EP

Subhash

SQL

Not
Only SQL



NoSQL Systems

Motivation

NoSQL: The Name

- "SQL" = Traditional relational DBMS
- Recognition over past decade or so:
  Not every data management/analysis problem is best
  solved using a traditional relational DBMS
- "NoSQL" = "No SQL" =
  Not using traditional relational DBMS
- "No SQL" ≠ Don't use SQL language

NoSQL: The Name

- "SQL" = Traditional relational DBMS
- Recognition over past decade or so:
  Not every data management/analysis problem is best
  solved using a traditional relational DBMS
- "NoSQL" exclusively = "No SQL" =
  Not using traditional relational DBMS
- "No SQL" ≠ Don't use SQL language

✱ "NoSQL" = "Not Only SQL"

Not every data management/analysis problem is best solved using a traditional DBMS

Database Management System (DBMS) provides….

… efficient, reliable, convenient, and safe multi-user storage of and access to massive
amounts of persistent data.

NoSQL Systems

Alternative to traditional relational DBMS

+ Flexible schema
+ Quicker/cheaper to set up
+ Massive scalability
+ Relaxed consistency → higher performance & availability

– No declarative query language → more programming
– Relaxed consistency → fewer guarantees

NoSQL Systems

Several incarnations

- MapReduce framework
- Key-value stores
- Document stores
- Graph database systems

MapReduce Framework

Originally from Google, open source Hadoop

- No data model, data stored in files
- User provides specific functions

  map() reduce()

- System provides data processing “glue”, fault-tolerance,
  scalability

Map and Reduce Functions

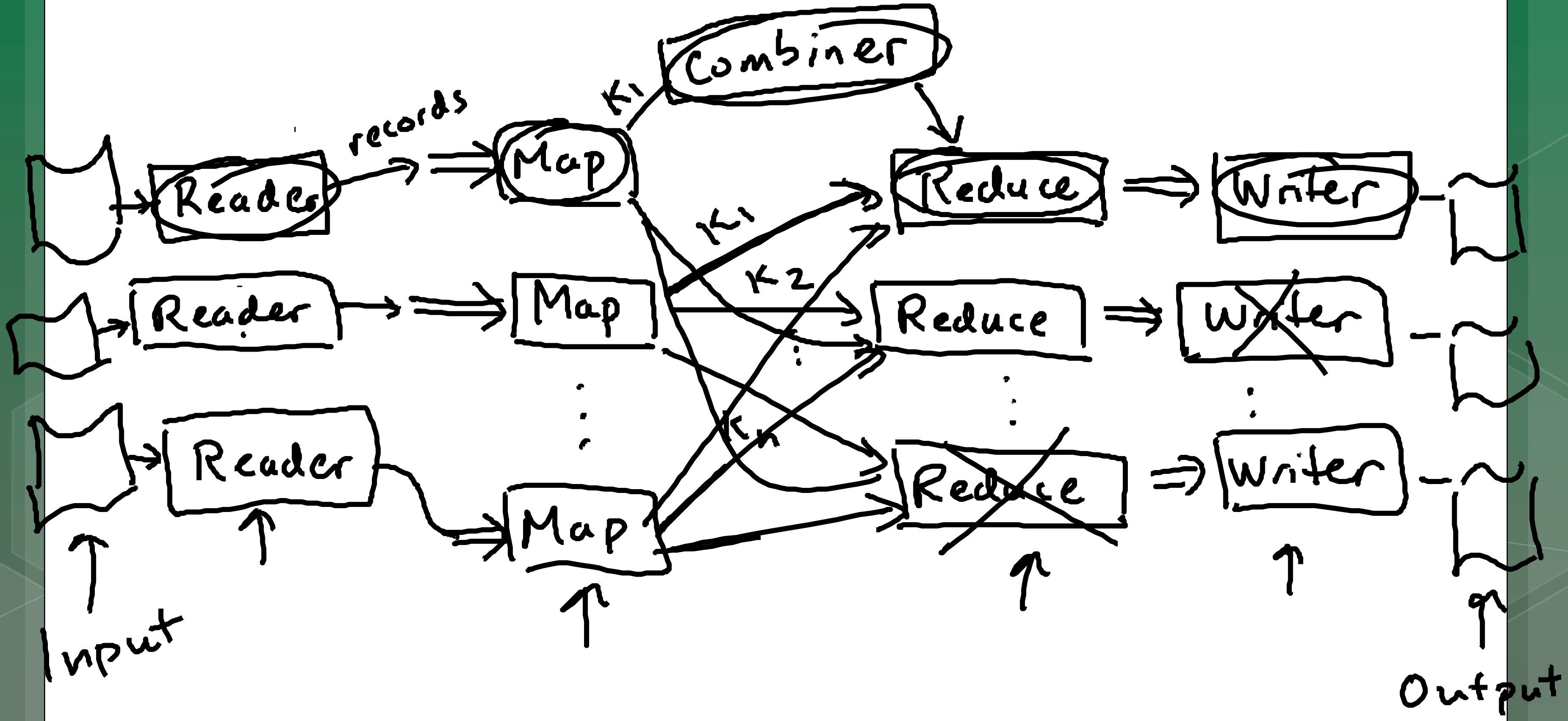
Map: Divide problem into subproblems

map(item) → 0 or more <key, value> pairs

Reduce: Do work on subproblems, combine results

reduce(key, list-of-values) → 0 or more records

MapReduce Architecture



MapReduce Example: Web log analysis

- Each record: UserID, URL, timestamp, additional-info
- Task: Count number of accesses for each domain (inside URL)

MapReduce Example (modified #1)

- Each record: UserID, URL, timestamp, additional-info
- Task: Total "value" of accesses for each domain based on additional-info

map(record) → ⟨domain, score⟩

reduce (domain, list of scores) → ⟨domain, sum⟩

MapReduce Framework

- No data model, data stored in files
- User provides specific functions
- System provides data processing “glue”, fault-tolerance,
  scalability

MapReduce Framework

Schemas and declarative queries are missed

- **Hive –** schemas, SQL-like query language
- **Pig –** more imperative but with relational operators
- Both compile to “workflow” of Hadoop (MapReduce) jobs

Key-Value Stores

Extremely simple interface

- Data model: (key, value) pairs
- Operations: Insert(key,value), Fetch(key),
  Update(key), Delete(key)

Implementation: efficiency, scalability, fault-tolerance

- Records distributed to nodes based on key
- Replication
- Single-record transactions, "eventual consistency"

Key-Value Stores

Extremely simple interface

- Data model: (key, value) pairs
- Operations: Insert(key,value), Fetch(key),
  Update(key), Delete(key)
- Some allow (non-uniform) columns within value
- Some allow Fetch on range of keys

Example systems

- Google BigTable, Amazon Dynamo, Cassandra, Voldemort, HBase, …

Document Stores

Like Key-Value Stores except value is document

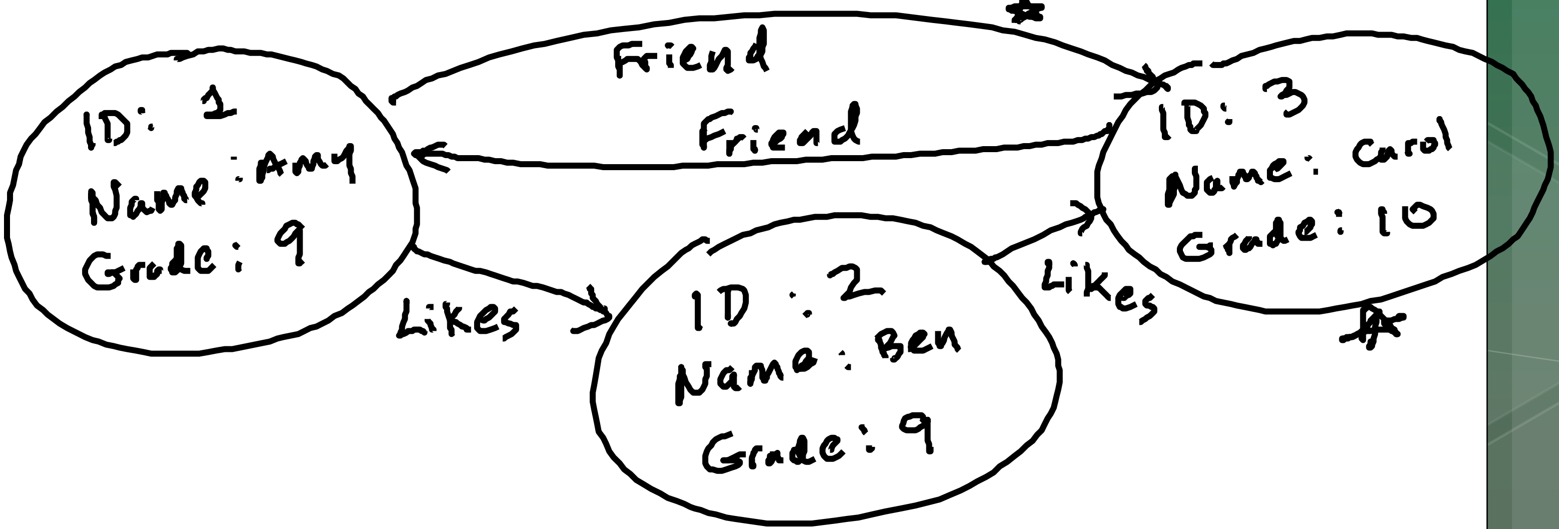
- Data model: (key, document) pairs
- Document: JSON, XML, other semistructured formats
- Basic operations: Insert(key,document), Fetch(key),
  Update(key), Delete(key)
- Also Fetch based on document contents

Example systems

- CouchDB, **MongoDB**, SimpleDB, …

Graph Database Systems

- Data model: nodes and edges
- Nodes may have properties (including ID)
- Edges may have labels or roles



Graph Database Systems

- Interfaces and query languages vary
- Single-step versus "path expressions" versus full recursion
- Example systems

  Neo4j, FlockDB, Pregel, …

That's all

Technology Series

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep