Lecture 9. NoSQL Databases, Big Data Technologies (Chapter 24 and 25)

alisa.lincke@lnu.se

Outline

- Data and Big Data
- NoSQL Databases
  - Document-Based (MongoDB)
  - Key-Value Stores
  - Graph-based Stores (Neo4j, OrientDB)
- Break 10 min
- Big Data Technology: MapReduce and Hadoop

Data

- The amount of data worldwide has been growing since 1994, as the result there is an explosive growth in the amount of data generated and communicated over networks worldwide.
- The applications collecting/generating every day information:
  - The social media websites (LinkedIn with more than 250 million users, Facebook with 1.3 billion and 800 million active users everyday, Twitter has ca 980 million with ca. 1 billion tweets per day)
  - Satellite imagery
  - Communication Networks (Telenor, Telia, etc.)
  - Banking
  - Other

Data Examples

- **Network data:**
  - Facebook: 500 million users
  - Twitter: 300 million users
  - Tele Communication data
  - Transport data
- **Document data:**
  - Web as a document repository ca 50 billions of web pages
  - Wikipedia: 4 million articles
  - Archives
- **Financial data:**
  - Banking, Accounting
- **Transaction data:**
  - Credit card companies: billions of transactions per day.
  - Queries in search engines (e.g., Google)
  - Membership cards allows to collect information about customer preferences/needs
- **Sensors data:**
  - Mobile sensors
  - Internet of Things (IoT) sensors network
  - Climate data: thousands of station
- **Linked data:**
  - Subtype of network data with semantics
- **Geographical data:**
  - Maps, geodata
- **Event-data:**
  - App log data if user interaction with App
- **Video data:**
  - Human movements in Sport,
- **Image data:**
  - Satellite imagery
  - Medical Images

Characteristics of Data

- Dependencies:
    - nondependencies (e.g., text), and
    - dependency-oriented data having relationship in time (time series, sequential data, spatial data)
- Data structure:
    - Structured: table (column, rows) or CSV file, network/graph (nodes, edges), objects with nested objects (JSON files)
    - Unstructured: image (pixels in rows,columns), voice data, text data

Characteristics of Big Data

The Gartner Group introduced five V’s characteristics for Big Data:

- **Volume**: refers to the size of the data stored and managed by the system. Examples: sensors, social media, environmental recording devices, credit card readers (transactional data), and more.
- **Velocity**: refers to frequency or speed of data to be generated, stored, processed. For example, *streaming data* (sensors, telecommunication data, health vital signals data, stock exchanges,)
- **Variety**: refers to structure/type of data, event data (clickstream, social media), location data (e.g., geospatial data, maps), images (surveillance, satellites, medical scanning), supply chain data, sensors data, video data (movies, YouTube streaming, etc.)
- **Veracity**: refers to the credibility of the source, and the suitability of data for its target audience (trust, and availability)
- **Value**: refers to what can we do with this data (to solve some problem, need, statistics, quality)

SQL-based Data and NoSQL based Data

- Data for SQL-based databases are:
  - University database
  - Hospital database
  - Traveling Agency database
  - Accounting database
  - Banking databases
  - Other…
- Data for NoSQL based databases are:
  - Social media data (network structure + document-based structure)
  - Archives (text data), images, videos (stored as files + document-based database)
  - Event-data or user interaction data with App, usually stored in JSON format, thus document-based database)
  - Sensors data (stored in files or in time-series databases TSBD (e.g.,InfluxData))

NoSQL Databases

- NoSQL (Not Only SQL) are other databases to suit the particular data and its characteristics (5 Vs), and application domain.
- NoSQL characteristics:
  - **Scalability:** where usually horizontal scalability is used by adding more nodes for data storage and processing as the volume of data grows
  - **Availability :** guaranties high availability due to using the distributed approach. In addition, using two access techniques: *hashing* and *range partitioning*
  - **Replication:** support master-slave, and master-master replication.
  - **Consistency:** Horizontal partitioning of the files records in NoSQL is usually used to access concurrently the records. In addition, many NoSQL applications does not require serializability.
  - **Not Required Schema:** allows semi-structured, self-described data (JSON objects). All constrains should be programmed in the application program
  - **Less Powerful Query Language:** only a subset of SQL based language is used (no JOIN operations)
  - **Versioning**: some NoSQL databases allows to store multiple versions of the data items, with the timestamps of when the data version was created.

NoSQL Databases Categories

- 1. Document-based NoSQL database
- 2. NoSQL Key-Value Stores
- 3. Column based or wide column NoSQL:
- 4. Graph-based NoSQL

1. Document-based NoSQL database

- Stores data in the form of collections of similar documents/objects
- Document is self-described data usually in BJSON format (Binary JavaScript Object Notation)
- Documents are accessible via their document id, or also indexes.
- Example JSON document/object:

{
‘id’: this.gameID,
‘type’: "playmode",
‘event"’: "point_selection",
‘state’: {‘game_progress’: {‘fields’: {Money: 10, Joy: 50, Health: 30}, ‘score’: 10} ‘value’:{name: ‘Banana’, times: 5}, event_count: 4},
‘timestamp’ : 1667736467
}

- Examples of well-known databases: MongoDB, CouchDB, DocumentDB, other

MongoDB

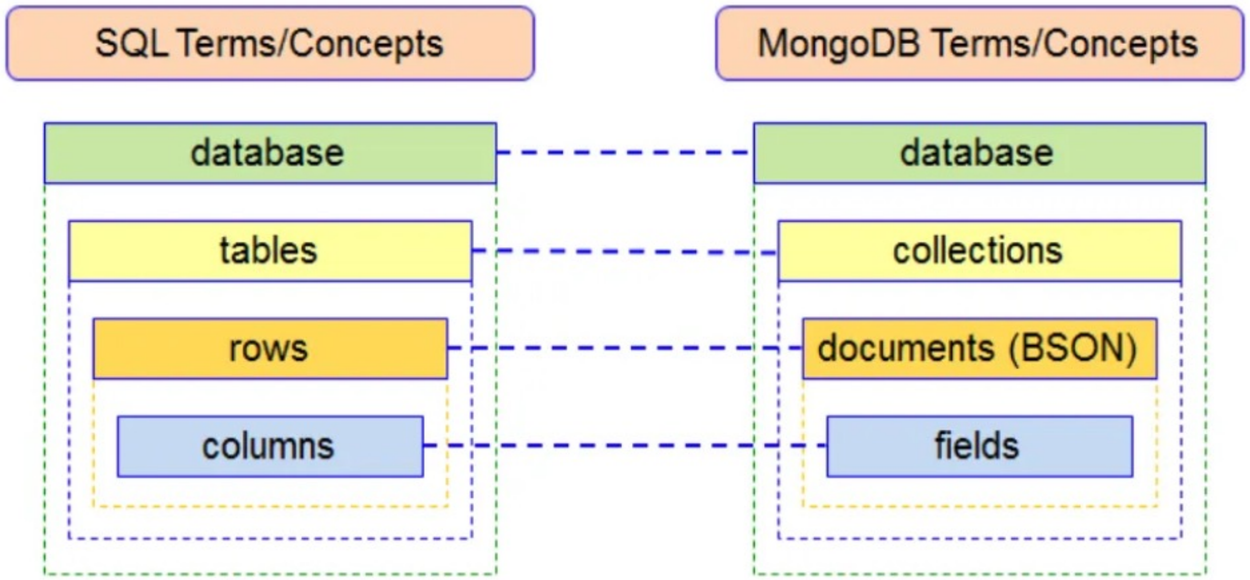
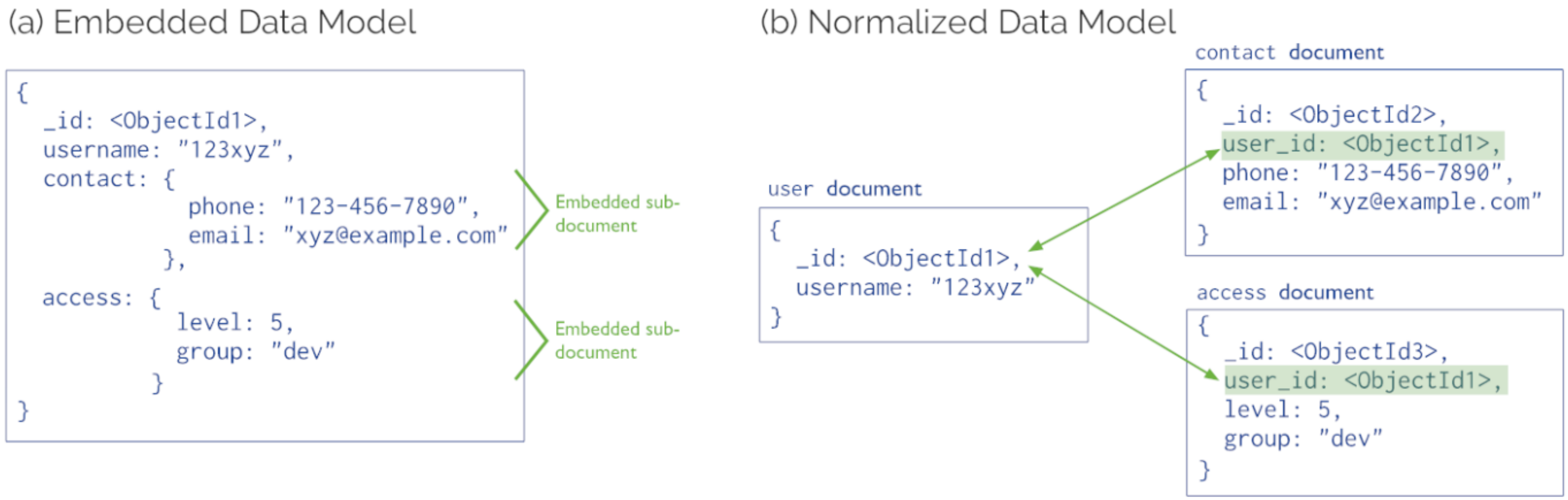


Image taken from: https://medium.com/zenofai/scaling-dynamodb-for-big-data-using-parallel-scan-1b3baa3df0d8

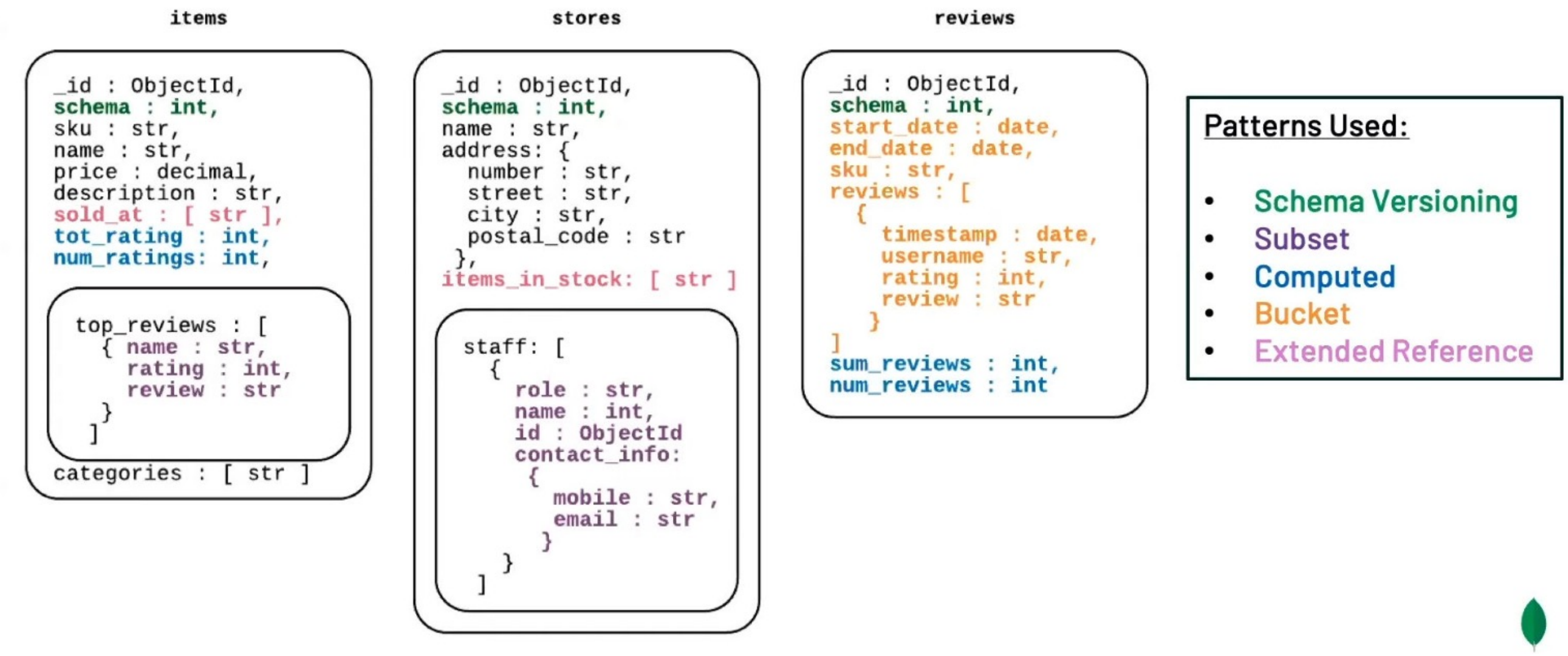
MongoDB Data Model (Flexible Schema)



Model relationships in MongoDB: https://devopedia.org/data-modelling-with-mongodb

Example MongoDB Schema (Model)

Web application, JavaScript, Mongoose library (DB-API)



An example MongoDB data model using various design patterns.
Source: [Genkina 2020, 31:38]

MongoDB Operations

Using MongoDB CLI (Command-line interface)

- Create database:
  - use “db_name”
- Create collection:
  - db.createCollection(name,structure)
  - For example: db.createCollection(“project”,{capped:boolean, size:int,max:int})
- CRUD operations:
  - db.collection_name.insert(<document(s)>)
  - db.collection_name.remove(<condition>)
  - db.collection_name.find(<condition>)

MongoDB

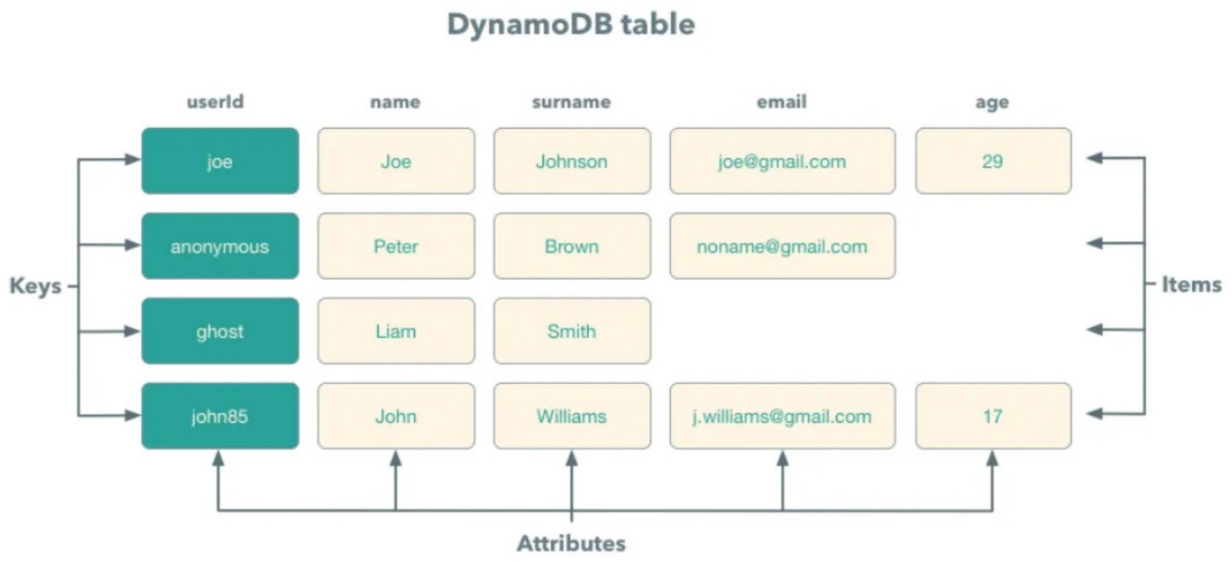
- MongoDB Server should be installed OR
- Using MongoDB Atlas cloud (no need to install Mongodb server)
- The DB-API is used to access mongodb from application
- For example, for web applications the JavaScript based Mongoose library is used

2. Key-Value Stores

- These systems have a simple data model based on fast access by the key to the value associated with they key
- The key is a unique identifier associated with a data item (value)
- The value can be a record, an object or a document, or even more complex data structure. Support different data types (strings of bytes, arrays of bytes, tuples, JSON objects)
- No query language
- Set of operations that can be used by the application programmers (GET,PUT,DELETE).
- Main characteristic: is that every value (data item) must associate with unique key and that retrieving the value by using key must be very fast.
- Usability/Applicability Examples: for streaming data, for real-time data processing and analyzes.
- Databases: Redis, Apache Kafka, Apache Cassandra, DynamoDB, other

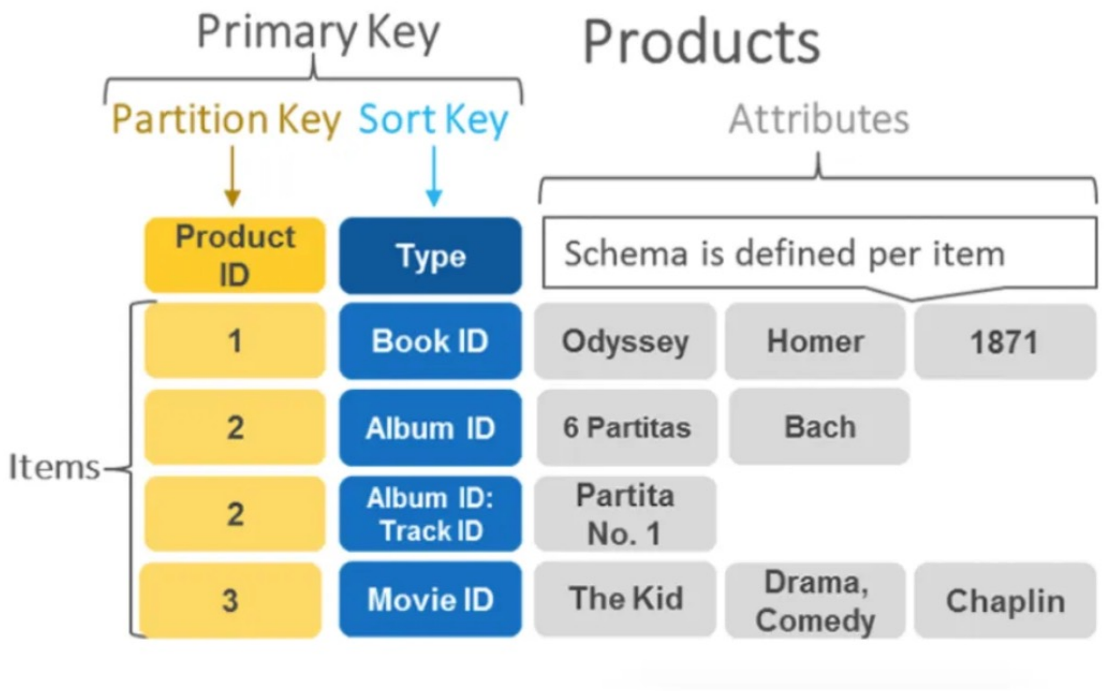
DynamoDB (1)

- Provided by Amazon Web Services (AWS)
- Uses concepts of table, items, and attributes
- Item is a value



DynamoDB (2)

- Table has a ***name*** and ***primary key***
- A primary key consists from two attributes (partition key, sort key).
- Partition key is used for hashing, and because there are will be same partition key, additional sorting key is used for ordering records in the same partition.



Graph-based Databases

- Graph databases is represented as a graph, which is a collection of vertices (nodes) and edges.
- Nodes and edges can be labeled to indicate the types of entities and relationships they represent
- Uses graph theory and algorithms for optimizing the data search
- Own query language (e.g., Cypher)
- Applications: analyzing social networks data, recommendations, geospatial data, postal delivery network
- Databases: Neo4j, OrientDb,

Neo4j

- Uses concepts of ***nodes*** and ***relationships (edges)***
- Separate structure for *data structure* and *graph structure*.
- Every node has a label (name) and properties (attributes)
- Relationships are edges
- Paths used for traversal in a graph (has start and end node)
- Indexing and node identifier. Each node has unique identifier, in addition user can create indexes for collection of nodes that have a particular label.
- https://console.neo4j.org/

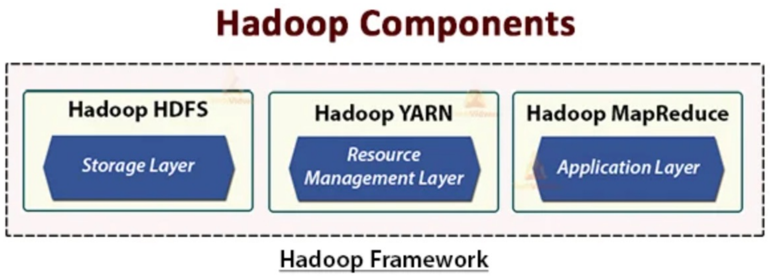
NoSQL Playgrounds

- Mongodb:
  - https://mongoplayground.net/
- Monogodb, Neo4j, Cassandra:
  - https://bitbucket.prodyna.com/projects/NOS/repos/nosql-playground/browse
- Kafka:
  - https://kafka-docker-playground.io/#/
  - https://www.conduktor.io/blog/kafka-playground-two-free-kafka-clusters-without-operational-hassles/
- Redis:
  - https://try.redis.io/
- Neo4j:
  - https://neo4j.com/sandbox/
  - https://console.neo4j.org/

10 min break

Hadoop

- Hadoop is an open-source Apache software for solving a problem involving massive amounts of data and computation.
- Was crated for finding a fast and scalable approach for web search engines.
- Consists of three main modules
  - MapReduce is programming model for parallel processing of large data sets
  - Hadoop Distributed File System (HDFS) for storage and provides high-throughput access to a data
  - Hadoop YARN: a framework for job scheduling and cluster resource management



MapReduce

- Developed by Dean and Ghemawat at Google in 2004
- Fault-tolerant implementation and runtime environment
- Programming style: *map* and *reduce* tasks
- Allows programmers to analyze very large datasets
- Underlying data mode assumed: key-value pairs

MapReduce Programming Model

- The general form of *map* and *reduce* functions:
    - Map[K1,V1] which is (key,value):List[K2,V2] and
    - reduce(K2,List[V2]):List[K3,V3]
- where, ***map*** is a generic function that takes a *key* of type K1 and a *value* of a type V1 and returns a *list* of key-value pairs of type K2 and V2. and
- ***reduce*** is a generic function that takes a *key* pf type K2 and a *list of values* of type V2 and returns a *list of key-value pairs* of type (K3,V3)
- In general, key types K1,K2,K3, etc. are different, with the only **one requirement** that the output types from Map function must match the input type of the Reduce function.

MapReduce Example

Count word frequency in a document

- **Input Splits**: divides the input into fixed-sized *n jobs* or input splits.
- **Mapping**: in this example a job of mapping phase is to count number of occurrences of each word from input splits and prepare a list of [(word, frequency)] (List[K2,V2])
- **Shuffling**: is to consolidate (sort/order) the relevant records from Mapping phase.
- **Reducing**: in this example performs aggregation (sum) function and combines values from Shuffling
- **Final output:** is single output value.

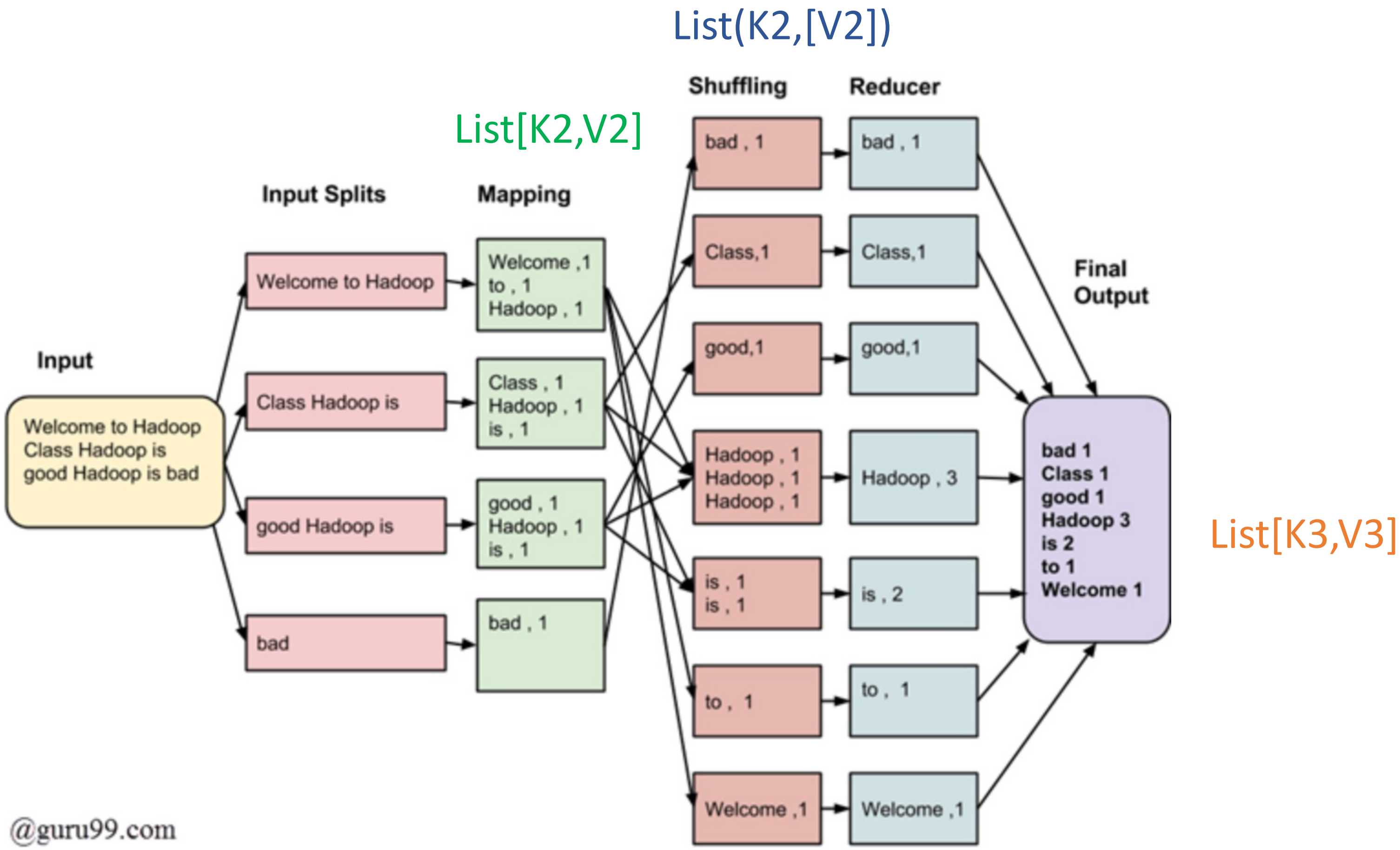


Image taken from: https://www.guru99.com/introduction-to-mapreduce.html

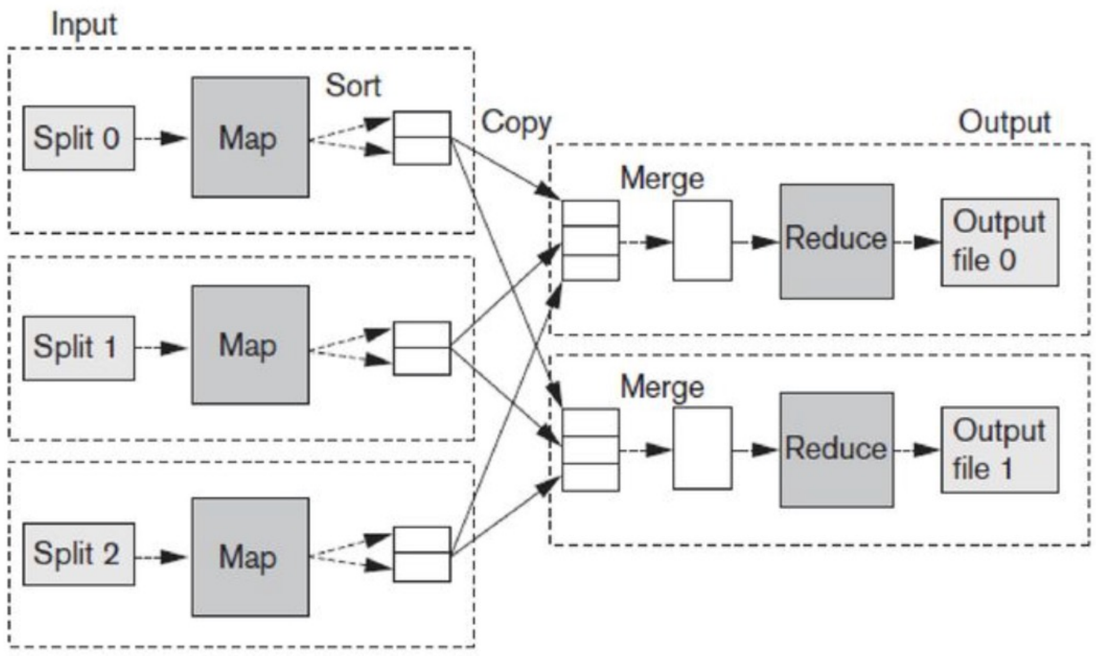
MapReduce Pceudocode

Example: Count word frequency in a document

```
map (String key, String value):
        for each word in w in value Emitintermediate(w, "1");
reduce (string key, Iterator values):
        Int result = 0;
        For each v in values:
            result+ =Parseint(v);
        Emit (key, Asstring(result));
```

General MapReduce Architecture

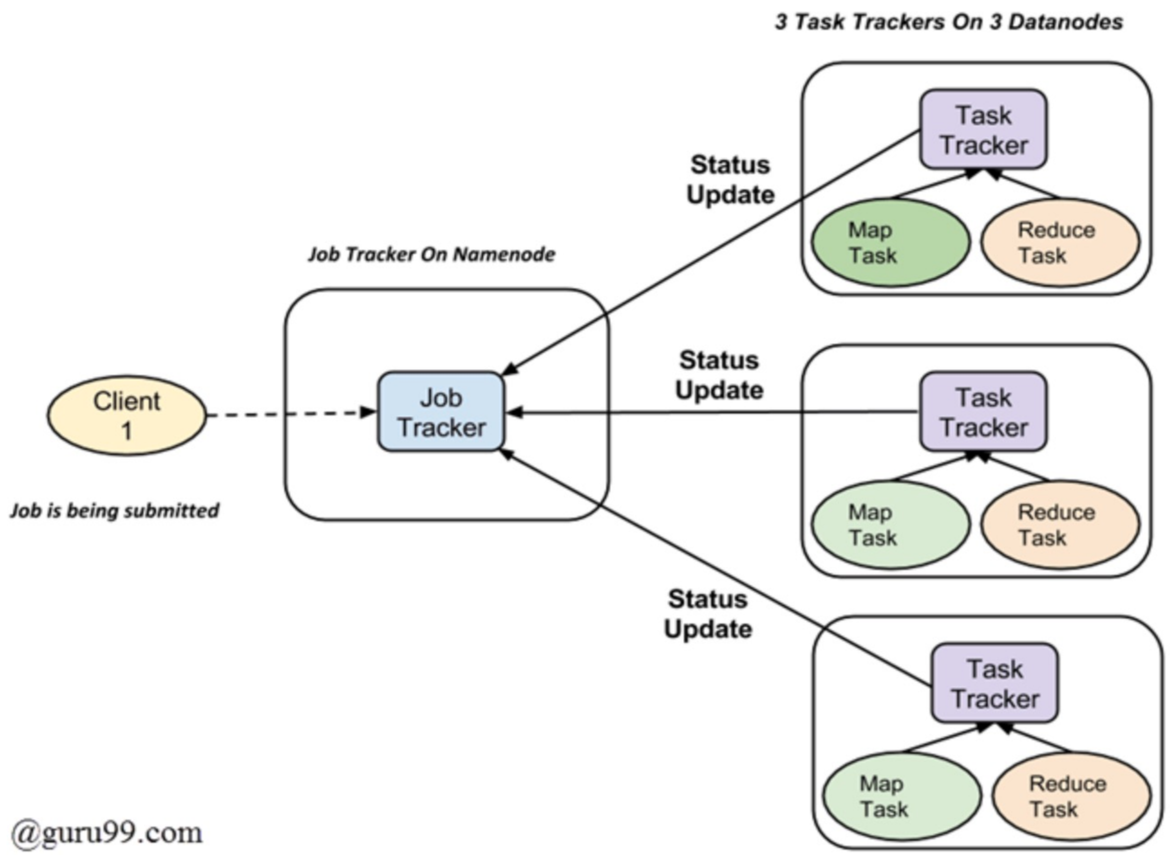
- It is beneficial to have multiple splits with an appropriate size (default 128 MB).
- One map task is created for each split.
- After mapping, the Sort operation is executed to make all tuples with the same key are sent to the corresponding reducer.
- The key-value list is a tuple type
- The output of the mapping is stored on local disk and removed automatically after the reducer has finished its task
- The mapping can be run in one machine and reducer on another machine to optimize the overload
- Shaffling (copy and merge) is done in another machine (where is the reducer is located) to create a list with (key, [value]) pairs where each reducer will have one unique key
- The map and reduce executes the user defined-functions (e.g., sum, count, etc.)
- The reduce output is stored in HDFS



Machine 1 Machine 2

MapReduce Runtime Environment

- The complete execution process (of Map and Reduce tasks) is controlled by two types of entities called:
  - Job Tracker: acts like a master
  - Multiple Task Trackers: acts like a slave, each of them performing the job. Run on DataNodes of the cluster
- For every job submitted for execution in the system, there is one Jobtracker
- Overall flow of MapReduce job:
  - Job submission
  - Job initialization
  - Task assignment
  - Task execution
  - Job completion

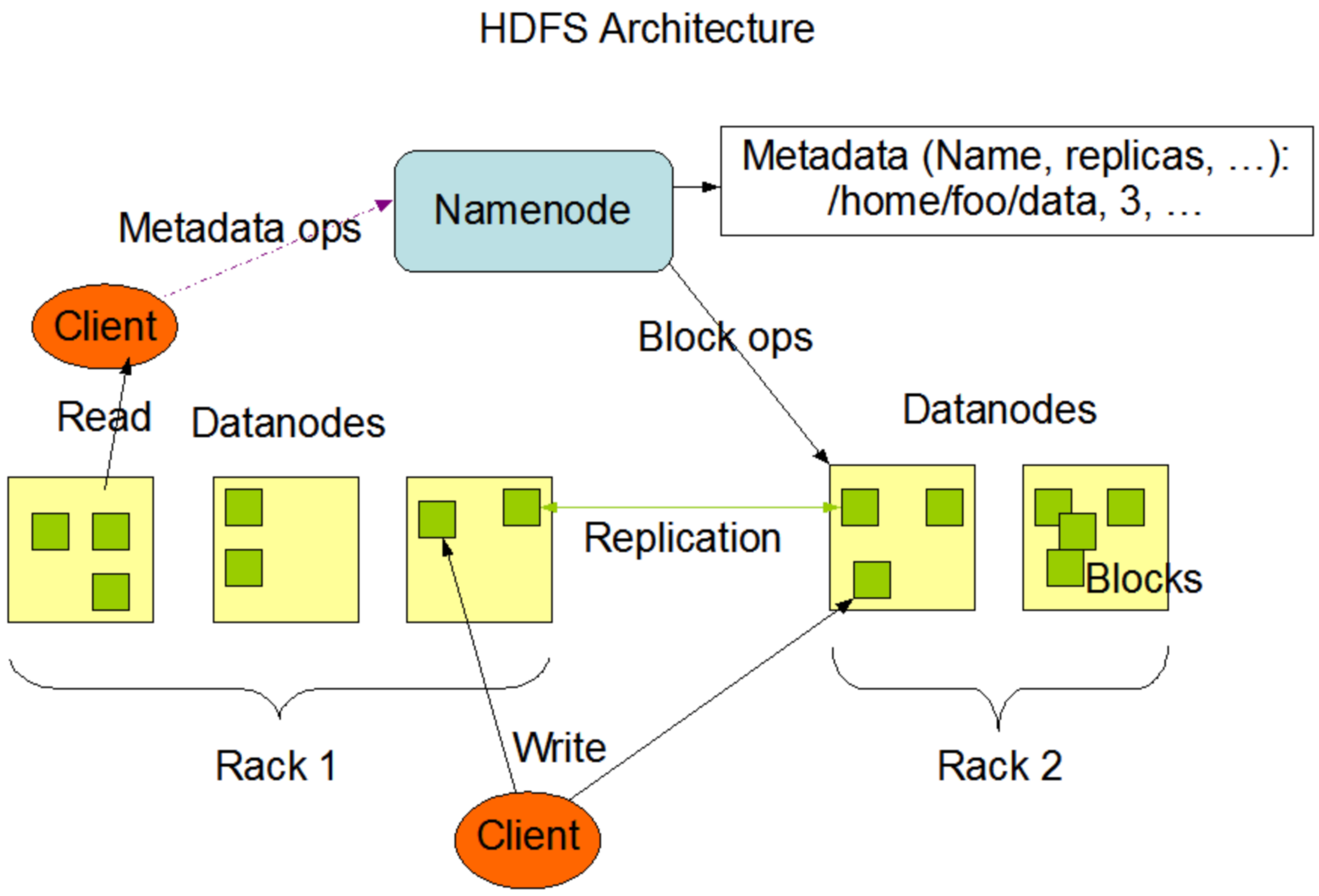


Hadoop Distributed File System (HDFS)

- HDFS:
  - is designed to run on a cluster of commodity hardware
    - Commodity hardware is involves the use of large numbers of already-available computing components for parallel computing, to get the greatest amount of useful computation at low cost.
  - provides high-throughput access to large datasets
  - is build using Java language and requires java to be installed to use this storage system.
  - stores file system metadata and application data separately on different servers (NameNode and DataNodes)
  - All servers are fully connected and communicate with each other using TCP-based protocol
  - Replication is done for DataNodes

HDFS architecture

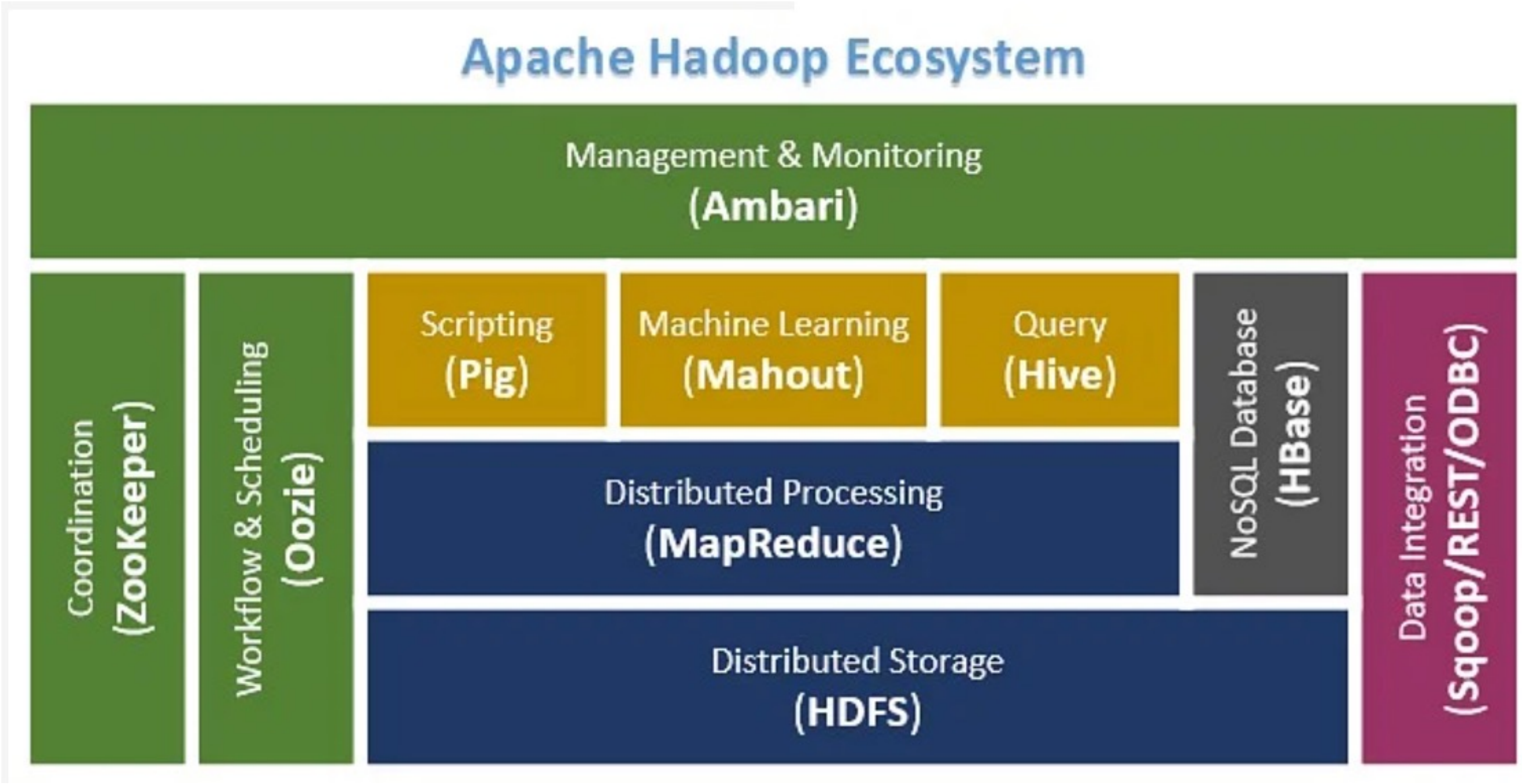
- Master-slave architecture
- NameNode and DataNodes are software designed to run on commodity machines (with GNU/Linux OS).
- Namenode is running on dedicated machine
- Other machines in the cluster runs one instance of the DataNode software.
- User data never flows through the NameNode.
- The files are broken into block-size chunks called data blocks
- Rack is a collocation of 30-40 DataNodes
- NameNode stores the filesystem metadata, such as files names, information about blocks of a file, blocks locations, permissions, etc. and used for managing the Datanodes
- Datanodes are storing the application data, it servers the client read/write requests based on the NameNode instructions.
- A cluster can have thousands of DataNodes, and tens of thousands of HDFS clients simultaneously connected
- Each block is replicated (default three copies) to a number of nodes in a cluster.



File I/O operations in HDFS

- HDFS supports a traditional hierarchical file organization. The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode.
- Provides single-writer, and multiple-reader model
- Files cannot be updated, only appended or removed
- A file consists of blocks
- Block placement:
  - Nodes of Hadoop cluster typically spread across many racks
  - Nodes on a rack share a switch

The Hadoop Ecosystem



References

- https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html