Data Storage Technologies

Lecture 1

CAP theorem states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:

- Consistency: Every read receives the most recent write or an error
- Availability: Every request receives a (non-error) response, without the guarantee that it contains the most recent write
- Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

Lecture 2

File Systems:

- XML has key values, good for data validation, checking it's not corrupted
- RDF is a sub class of XML

ACID (Atomicity, Consistency, Isolation, Duration) properties provides the principles that database transactions should adhere to, they ensure that data doesn’t become corrupt as a result of a failure of some sort. A transaction is a single logical operation that may consist of one or many steps (such as in banking).

Lecture 3

Relational DBMS:

- Normal forms (1-5)
- Indexing
- Architectures: linking & foreign keys, one-to-many, many-to-many, EAV
- CRUD: Create Read Update Delete
- Transactions: an action or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database.

Tasks:

- Make short presentation (~5min) on indexes and Lucene Spatial
- Continue *game* task

Lecture 4

Persistences:

- In-Memory, will disappear if loss of power (e.g. RAM)
- Snapshot, save a state of data to be replaced by a new one
- Write Ahead Log (WAL), keep a log of every action perform since last snapshot
- Disk-based, permanent save

Tasks:

- Make general code that can be adapted to different schemes
- Maybe difficulty with `DISTINCT ON`
- Make sure code is working with easy example

Lecture 5

Redis (key-value)

Install:

```
wget https://download.redis.io/releases/redis-6.2.1.tar.gz
tar xzf redis-6.2.1.tar.gz
cd redis-6.2.1
make
make test
sudo make install
```

Create/modify the file *redis.conf* (optional)
Run in a terminal `redis-server <conf_file>` to run the server (*conf_file* optional)
In *conf_file* the important lines:

- `appendonly no` change to yes
- `dir ./` can be changed to set the directory to save the database
- `save 3600 1` to save a snapshot every 3600 seconds if at least 1 change has been done

Run `redis-cli` in another terminal to use the following commands:

- `set`, `get`, `mset`, `mget`
- `hset`, `hget`, `hmset`, `hmget`
- `rpush`, `lpush`, `llen`, `lrange`, `lpop`
- `sadd`, `spop`, `smembers`, `sunion`, `sdiff`
- `incr`, `decr`, `incrby`, `decrby`
- `exists`, `del`, `expire`, `persist`, `ttl`
- `multi`, `exec`

Column-oriented DBMS

Faster on HDD since usually more rows are present than columns

Lecture 6

Document Stores

MongoDB, CouchDB, PostgreSQL (it is a multi model DB, relational DBMS & document store); they can use file systems such as *JSON* and *BSON*.
REST (REpresentational State Transfer) API works good with document stores.

REST query examples:

```
curl -H "accept: application/json" -H "Content-Type: application/json" -H "x-
apikey: cc8b2c70b4c1271b49775b1bf6d27befbc929" -d 'q={"$or":[{"name":"Peter"},
{"profession":{"$regex":".*Data.*"}}]}' -G "https://dstnsu-
011f.restdb.io/rest/dst-nsu" | jq ''
curl -H "accept: application/json" -H "Content-Type: application/json" -H "x-
apikey: cc8b2c70b4c1271b49775b1bf6d27befbc929" -d 'q={"profession":
{"$exists":true}}' -G "https://dstnsu-011f.restdb.io/rest/dst-nsu" | jq ''
```

```
import requests

r = requests.get('https://dstnsu-011f.restdb.io/rest/dst-nsu?q=
{"name": {"$in": ["Peter", "Khue"]}}&h=
{"$orderby": {"profession": 1}}', headers={'x-
apikey': 'cc8b2c70b4c1271b49775b1bf6d27befbc929'})
```

REST operators: restdb.io/docs/querying-with-the-api#restdb

Lecture 7

RDF (Ressource Description Format) store or **triplestore** is a database built for the retrieval of triples: *subject-predicate-object*.
RDF vocabularies (ontologies):

- FOAF (Friend Of A Friend)
- Dublin Core, composed of 15 metadata elements to describe resources
- schema.org, promotes schemas for structured data
- SKOS (Simple Knowledge Organization System)

Recommended RDF store:

- Jena (link), a Java API which can be used to create and manipulate RDF graphs

Lecture 8

Graph DBMS (allows more flexibility ?): Neo4j

Pros and cons of different databases

Relational:

- pros: consistency
- cons: if the data changes you need to modify your schema which can lead to problems

Column-oriented:

- pros: faster to retrieve fields in data, faster to add/remove/modify fields
- cons: slower to add/remove/modify records

Key-Value:

- pros: fast research
- cons: no relation described between the data

Document-oriented:

- pros: flexibility of data by storing it discretely in groups of key-value pairs called documents
- cons: not consistent

OrientDB

Install:

- download from: orientdb.org/download-previous
- follow install from: tutorialspoint.com/orientdb/orientdb_installation.htm

To start the service run:

- `cd $ORIENTDB_HOME/bin && ./server.sh` to start the server (if the error `unrecognized option: -d64` appears then remove this option from `./server.sh`)
  `$ORIENTDB_HOME` is the path to the directory `orientdb-x.x.x` if not setup during installation
- `cd $ORIENTDB_HOME/bin && ./console.sh` to start the console (in a different terminal)
- to enter the studio, enter the link *http://localhost:2480/* in a browser

Some create commands:

- `create class myClass extends V` to create a vertex class (replace `V` by `E` for an edge class)
- `create vertex myVertex content {pty1:"v1", pty2:"v2"}` to create a vertex to (select from myVertex where pty1 = "v2")``
  to create an edge

Some query commands:

- `select from myClass where pty = "v"`
- `match {class: myClass, as: cls, where: (pty = "v")} return cls.pty`
- `traverse * from myClass while pty = "v"`
- `traverse in() from (select from myClass where pty = "v")` can also add `maxdepth int` or/and `strategy myStrategy`

Lecture 9

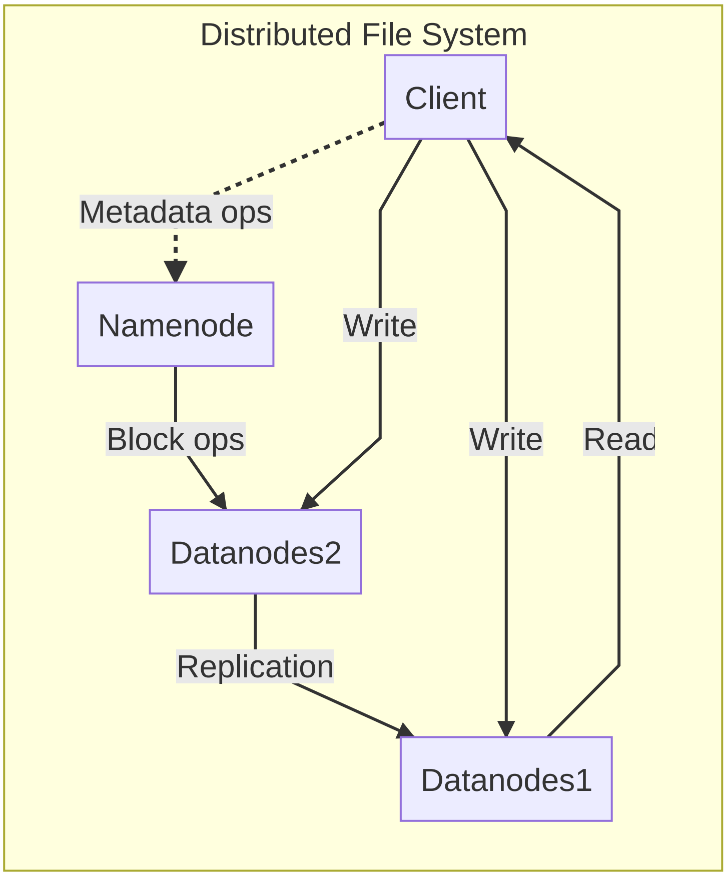
For next time:

- Get cloudera virtual machine ready, it is necessary for its programs preinstalled

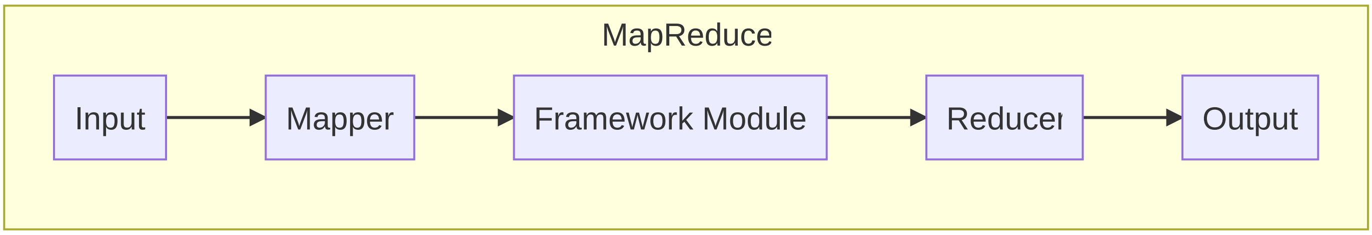
Lecture 10

What different DBMS approaches are best for:

RDBMS	**NoSQL**
Normalized	Big
Real time, transactions	Not real time, analytics
SQL	MapReduce
CA	AP, CP



MapReduce:



For next time:

- Prepare presentation on Mahout ✓
- Run MapReduce via hadoop ✓

```
docker pull cloudera/quickstart:latest
docker run -m 8G --memory-reservation 2G --memory-swap 8G --hostname=quickstart.cloudera --privileged=true -t -i -v $(pwd):/alix --publish-all=true -p8888 -p8088 cloudera/quickstart /usr/bin/docker-quickstart

# The following is to type inside the container cli
hadoop fs -copyFromLocal /alix/DST /user/root/DST
hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.7.0.jar \
    -D mapred.reduce.tasks=1 \
    -input /user/root/DST/dst-stu/d/mr/tf-idf \
    -output /user/root/DST/output \
    -mapper /alix/DST/mapper.sh \
    -reducer /alix/DST/reducer.sh
hadoop fs -cat /user/root/DST/output/part*
hadoop fs -rm -r /user/root/DST/output
```