# CSE 535 Asynchronous Systems
# Hadoop

YoungMin Kwon

# Size of Data

- We live in the data age
  - The New York Stock Exchange generates 1 terabyte of new trade data a day
  - Facebook hosts 10 billion photos
  - Ancestry.com stores around 2.5 petabytes of data
  - The Internet Archive stores around 2 petabytes of data and is growing by 20 terabytes per month
  - The Large Hardon Collider near Geneva, Switzerland produce about 15 petabytes of data per year

# Size of Data

- The volume of publicly available data increases every year
  - Organizations cannot simply manage their own data
  - They need to extract meaningful data from other organizations' data
- It has been said that more data usually beats better algorithms
  - Recommending a movie or a restaurant

# Data Storage and Analysis

- The issue: the storage capacity of HDD have increased massively, but access speeds have not kept up.
  - In 1990's a typical drive stores 1GB of data at a transfer speed of 4.4 MB/s
  - These days, 1 terabyte drive is the norm at a transfer speed of 100 MB/s (it takes more than 2.5 hours to read all the data)

# Data Storage and Analysis

- A solution: read from multiple disks at once
  - Suppose that the 1 TB of data are distributed in 100 disks
  - Working in parallel, the entire data can be read under 2 minutes

# Data Storage and Analysis
## Issues with the solution

- More hardware means more failures
  - Add redundancy to fix the issue
  - RAID (Redundant Array of Independent Disks)
  - HDFS (Hadoop File System)

- Combining the data from many sources
  - The data read from one disk may need to be combined with the data read from the other 99 disks.
  - MapReduce provides an abstraction that transforms disk reads and writes into a computation over sets of keys and values

# Data Storage and Analysis

- Hadoop provides
  - A reliable shared storage and analysis system
  - The storage is provided by HDFS
  - The analysis is provided by MapReduce

# Comparison with Other Systems

## : Relational Database Management System

- Seek time improves slower than transfer rate
- If seeks dominate the data access pattern
  - It will take longer to read/write large portions of data than streaming through it
- Updating a small portion of records works well in a B-Tree-based database
- Updating a majority of a database, B-tree is less efficient than MapReduce that sort/merge to rebuild database

# Comparison with Other Systems
## : Relational Database Management System

- RDBMS
  - Good for point queries and updates
  - Gigabytes of data
  - Structured data
- MapReduce
  - Good for analyzing the whole dataset in a batch fashion for ad-hoc analysis
  - Petabytes of data
  - Semi-structured and unstructured data

# Comparison with Other Systems
## : Grid Computing

- High Performance Computing (HPC) and Grid computing
  - Doing large-scale data processing using Message Passing Interface (MPI) API
  - HPC distributes the work across a cluster of machines that access a shared file system hosted by a Storage Area Network (SAN)
  - Works well for computing-intensive jobs
  - Not so efficient for larger data volumes (100+ GB)

# Comparison with Other Systems
## : Grid Computing

- MapReduce uses the data locality
- MPI API
  - It gives great control to the programmer
  - Programmers handle the data flow exposed via low-level C routines as well as high level algorithms
- Node failures
  - One of the difficult challenges in coordinating large scale distributed systems
  - MapReduce's shared-nothing architecture provides the robustness against the node failures

# Install Hadoop

- Install Hadoop
  - https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm
- SSH Setup and Key Generation
  - `ssh-keygen -t rsa`
  - `cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`
  - `chmod 0600 ~/.ssh/authorized_keys`
- Java setup
  - `Add the following lines to ~/.bashrc`
  - `export JAVA_HOME=/usr/lib/jvm/java-8-oracle`
  - `export PATH=$PATH:$JAVA_HOME/bin`

# Install Hadoop

- Downloading hadoop
  - `sftp` hadoop@10.12.9.155
  - `get hadoop-2.8.0.tar.gz`
  - (alternatively) `wget` http://apache.claz.org/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz

  - `tar –xzvf hadoop-2.8.0.tar.gz`
  - `mv hadoop-2.8.0/* /usr/local/hadoop/`

# Install Hadoop

- Add the following lines to ~/.bashrc

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib
export
PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

- Run

```
source ~/.bashrc
```

# Install Hadoop

- Update hadoop-env.sh with

```
cd $HADOOP_HOME/etc/hadoop
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

- Update core-site.xml with

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

# Install Hadoop

- Update hdfs-site.xml with

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.name.dir</name>
<value>file:///home/hadoop/hadoopinfra/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
<value>file:///home/hadoop/hadoopinfra/hdfs/datanode</value>
    </property>
</configuration>
```

# Install Hadoop

- Update yarn-site.xml with

```
<configuration>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

- Update mapred-site.xml by

```
cp mapred-site.xml.template mapred-site.xml
```

  - Update mapred-site.xml with

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

# Verify Hadoop Installation

- Name Node Setup
  - `cd ~`
  - `hdfs namenode -format`
- Hadoop dfs
  - `start-dfs.sh`
- Yarn script
  - `start-yarn.sh`
- Accessing Hadoop on Browser
  - http://localhost:50070
- All applications for cluster
  - http://localhost:8088