Overview of Hadoop

Igor Yakushin
ivy2@uchicago.edu

December 9, 2019

What is Hadoop?

- A framework to process huge amount of data
- Implemented in Java
- Runs on a cluster of commodity computers, from a few to a few thousand nodes
- Consists of two major components:
  - HDFS - distributed file system; everything you put there is automatically divided into blocks that are replicated and spread across the cluster;
  - MapReduce - an approach to perform calculations on such data in parallel.

Hadoop zoo

- There is a zoo of other Hadoop related tools
  - Hive - SQL interface to Hadoop built on top of HDFS and MapReduce
  - Impala - another SQL interface to Hadoop, allegedly much faster
  - HBase - noSQL fast distributed database on top of HDFS
  - Pig - data processing languague built on top of HDFS and MapReduce
  - Spark - for performance reasons no longer relies on MapReduce, can be used without Hadoop; its native language is Scala but it can also be used from Java, Python, R
  - Sqoop - copy data between relational database and Hadoop
  - Flume - copy data into Hadoop, typically used to store logs from a cluster for subsequent analysis
  - YARN - resource allocation manager, used to submit jobs
  - Zookeeper - centralized service for maintaining configuration information, naming, providing distributed synchronization
  - Hue - web GUI to many of the above tools
- In Big Data Platform class you'll mostly use pySpark via JupyterHub interface, HDFS, Hive, Pig, Hue.

Hadoop vs traditional RDBMS

RDBMS	**Hadoop**
do not scale well beyond a few terabytes	easily scales to petabytes by adding more computers to the cluster
for large datasets runs on very expensive enterprise server	data is scattered on a cluster of cheap commodity computers
works on structured data; one needs to predefine the data schema	can accept any unstructured data and worry about interpreting and reinterpreting it later
data needs to be normalized to avoid duplication and enforce constraints	works best on a single denormalized table

Hadoop vs traditional RDBMS

RDBMS	**Hadoop**
can be faster for queries on small subset of data	might introduce too much overhead for such queries
ACID - compliant	in general - not ACID-compliant
natively speaks SQL	natively implements MapReduce approach in Java on top of which some subset of SQL might be supported
is good for banks to keep track of transactions	is good for data scientists to try various ideas on a huge sets of data

ACID compliance

‘

- **A**tomicity - The database transaction must completely succeed or completely fail
- **C**onsistency - During the database transaction, RDBMS progresses from one valid state to another. The state is never invalid
- **I**solation - The client's database transaction must occur in isolation from other clients attempting to transact
- **D**urability - Once transaction is committed, it will remain so, even in the event of power loss, crashes, or errors. The data operation that was part of the transaction must be reflected in nonvolatile storage.

Hadoop has no concept of transaction so is not ACID compiliant.

Distributions

There are many Hadoop distributions that bundle different set of tools and add their own:

- Apache - free, open source;
- Cloudera - we are using Cloudera 6.3;
- Hortonworks - Hortonworks and Cloudera recently merged but they still maintain two separate distributions and working on the joint one
- HDInsight - Microsoft Azure's Cloud based Hadoop Distribution
- MapR
- ...

Many distributions, for example Cloudera, provide virtual machines to play with