Hive & Pig

By Abhishek Roy

About Me

Abhishek Roy

- Around 11 years of tech experience with services, product and startups
- Was leading the data team at Qyuki Digital Media, a social network for artists
- Involved in Big data and Hadoop with emphasis on recommendation systems and machine learning approaches for the last 3 years
- Training Big Data (Hadoop, Mahout, Cassandra) for 3 years. I have trained around 1500 students
- Co-founder@ **www.feetapart.com,** a social wellness platform

What do you get?

- Learn the Big Data/Hadoop concepts with focus on the hands on approach
- Learn Hive & Pig and start playing around
- Related material (slides, code, data)
- Lifetime help and support

Agenda

- What is Big Data
- History and Background
- Hadoop Introduction
- HDFS Introduction
- Hadoop Setup *(Hands On)*
- HDFS architecture & Deep Dive *(Hands On)*
- Map Reduce Introduction

Agenda (cont.)

- Hive *(with Hands On examples)*
- Pig *(with Hands On examples)*

What Is Big Data?

Like the term Cloud, it is a bit ill-defined/hazy

Big Data – A Misnomer

- Misleading to quick assumptions
  - Current challenges are driven by many things, not just the size of data
- ANY company can use the Big Data principles to improve specific business metrics
  - Increased data retention
  - Access to all the data
  - Machine learning for pattern detection, recommendations
- But what has happened to cause this all?

What's the "BIG" hype about Big Data?

There may be hype, but problems are real and of big value. How?

- We are in the age of advanced analytics (that's where all the problem is ,we want to analyze the data) where **valuable business insight** is mined out of historical data.
- We live in the age of crazy data where every **individual , enterprise and machine** leaves so much data behind summing up to many terabytes & petabytes , and it is only expected to grow.

What's the "BIG" hype about Big Data?

- Good news. Blessing in disguise. **More data means more precision.**
    - **More data usually beats better algorithms.**

- But how are we going to analyze?
    - Traditional database or warehouse systems crawl or crack at these volumes.
    - Inflexible to handle most of these formats.
    - This is the very characteristic of Big Data.

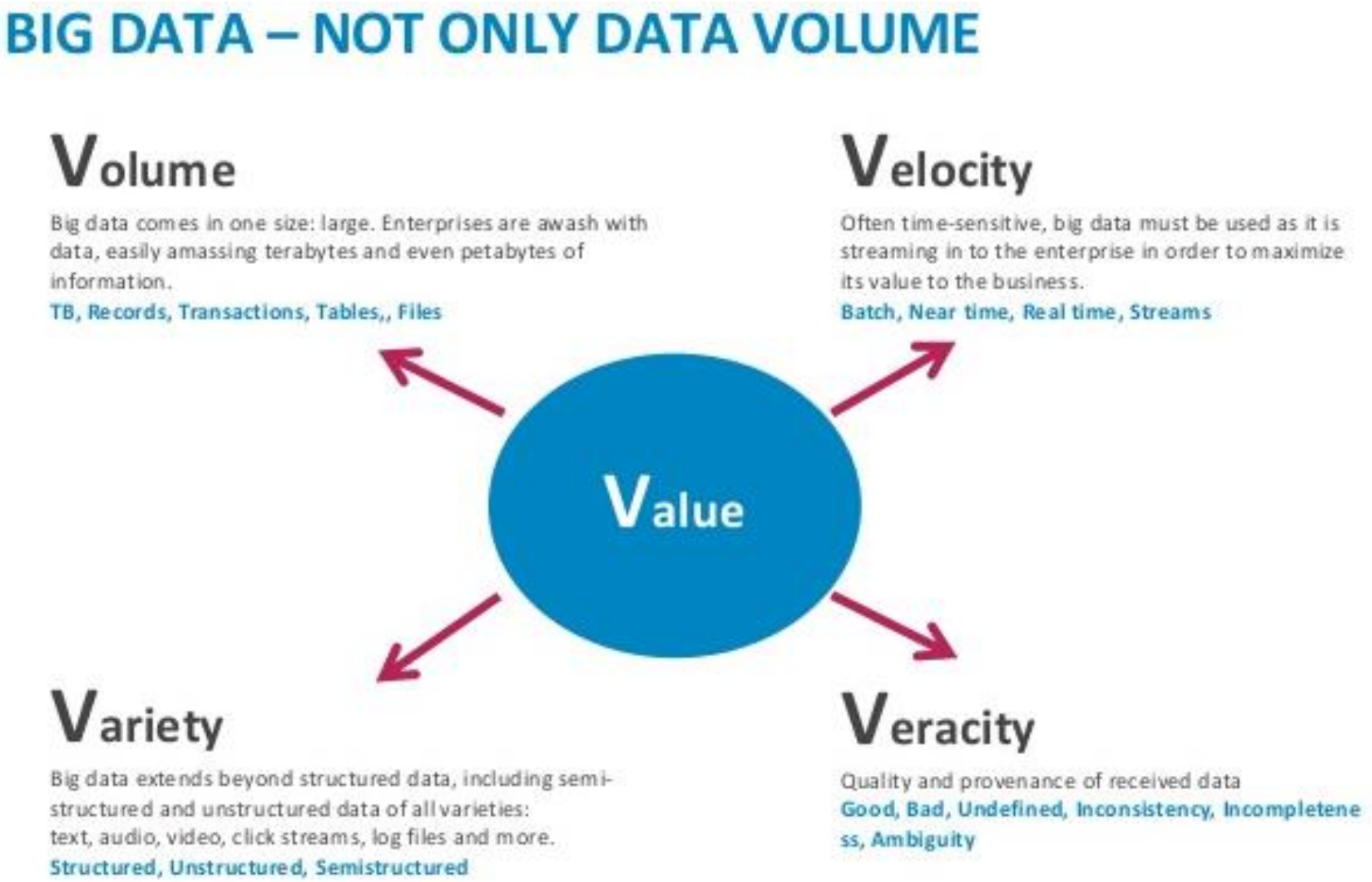
Key Hadoop/Big Data Data Sources

- Sentiment
- Clickstream
- Sensor/Machine
- Geographic
- Text
- Chatter from social networks
- Traffic flow sensors
- Satellite imagery
- Broadcast audio streams

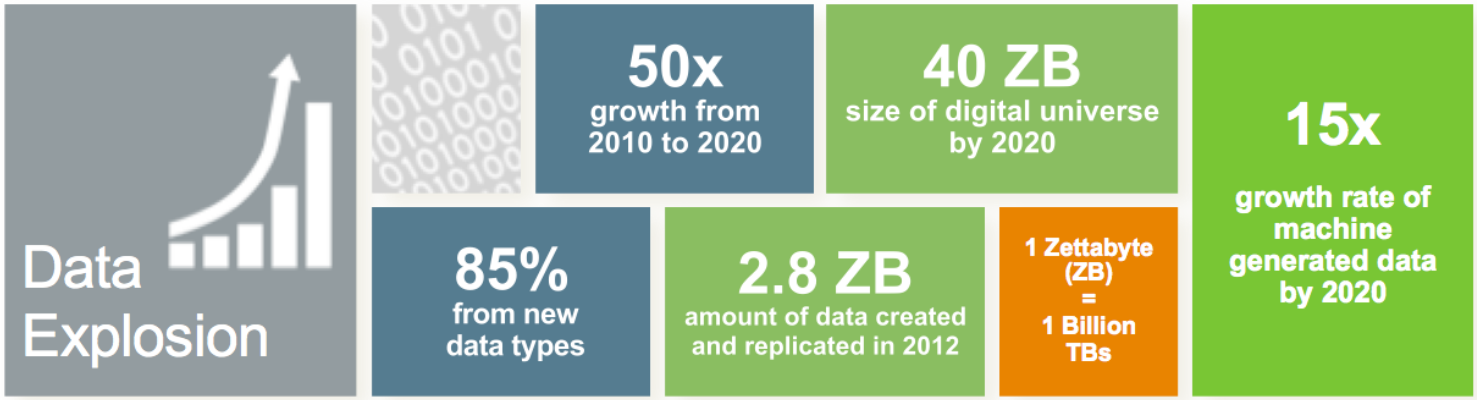
Introduction cont.

- Nature of Big Data
  - Huge volumes of data that cannot be handled by traditional database or warehouse systems.
  - It is mostly machine produced.
  - Mostly **unstructured** and grows at high velocity.
  - Big data doesn’t always mean huge data, it means “difficult” data.

The 4 Vs


BIG DATA – NOT ONLY DATA VOLUME
Volume
Big data comes in one size: large. Enterprises are awash with data, easily amassing terabytes and even petabytes of information.
TB, Records, Transactions, Tables,, Files
Velocity
Often time-sensitive, big data must be used as it is streaming in to the enterprise in order to maximize its value to the business.
Batch, Near time, Real time, Streams
Value
Variety
Big data extends beyond structured data, including semi-structured and unstructured data of all varieties: text, audio, video, click streams, log files and more.
Structured, Unstructured, Semistructured
Veracity
Quality and provenance of received data
Good, Bad, Undefined, Inconsistency, Incompleteness, Ambiguity

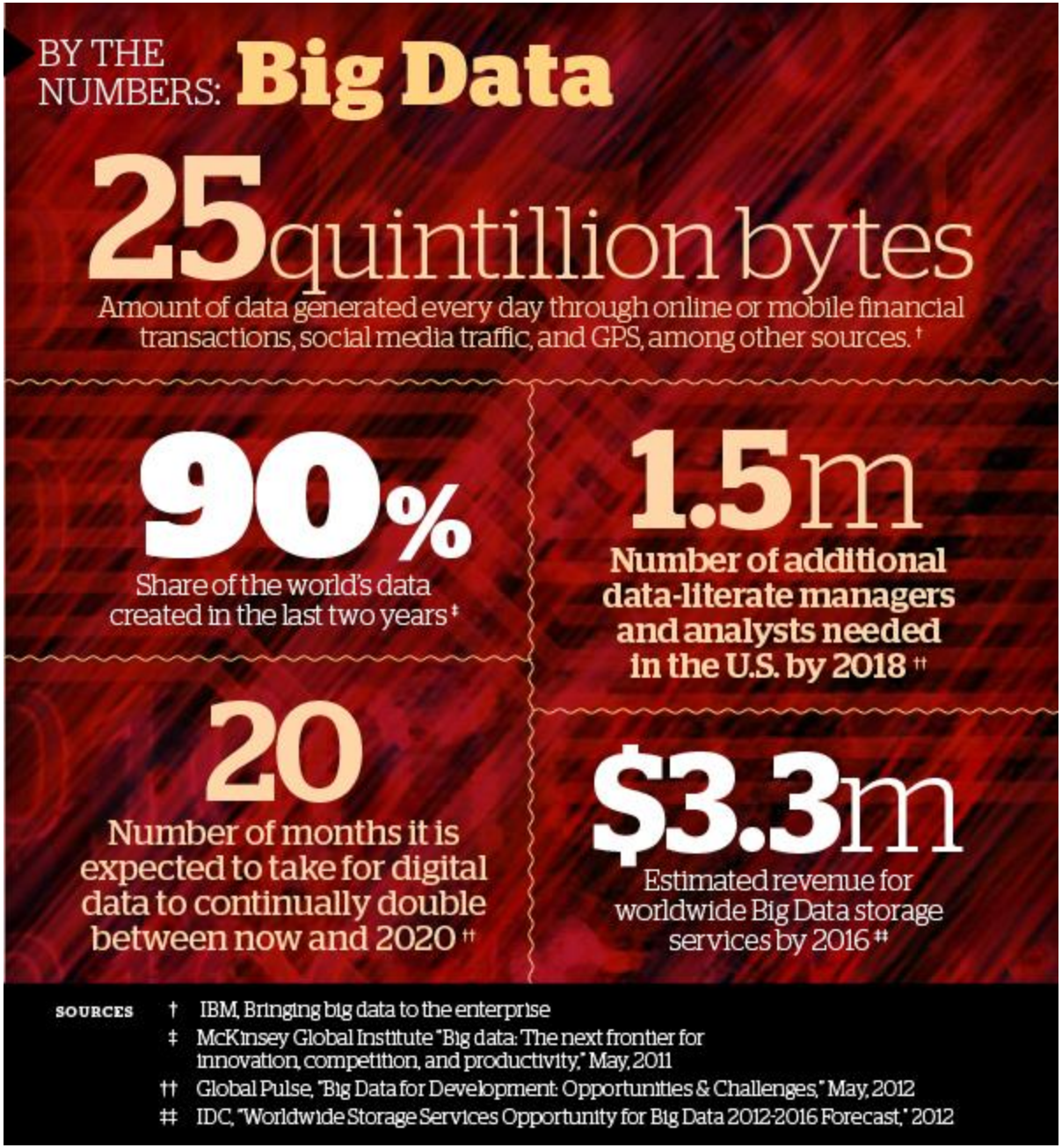
Data Explosion



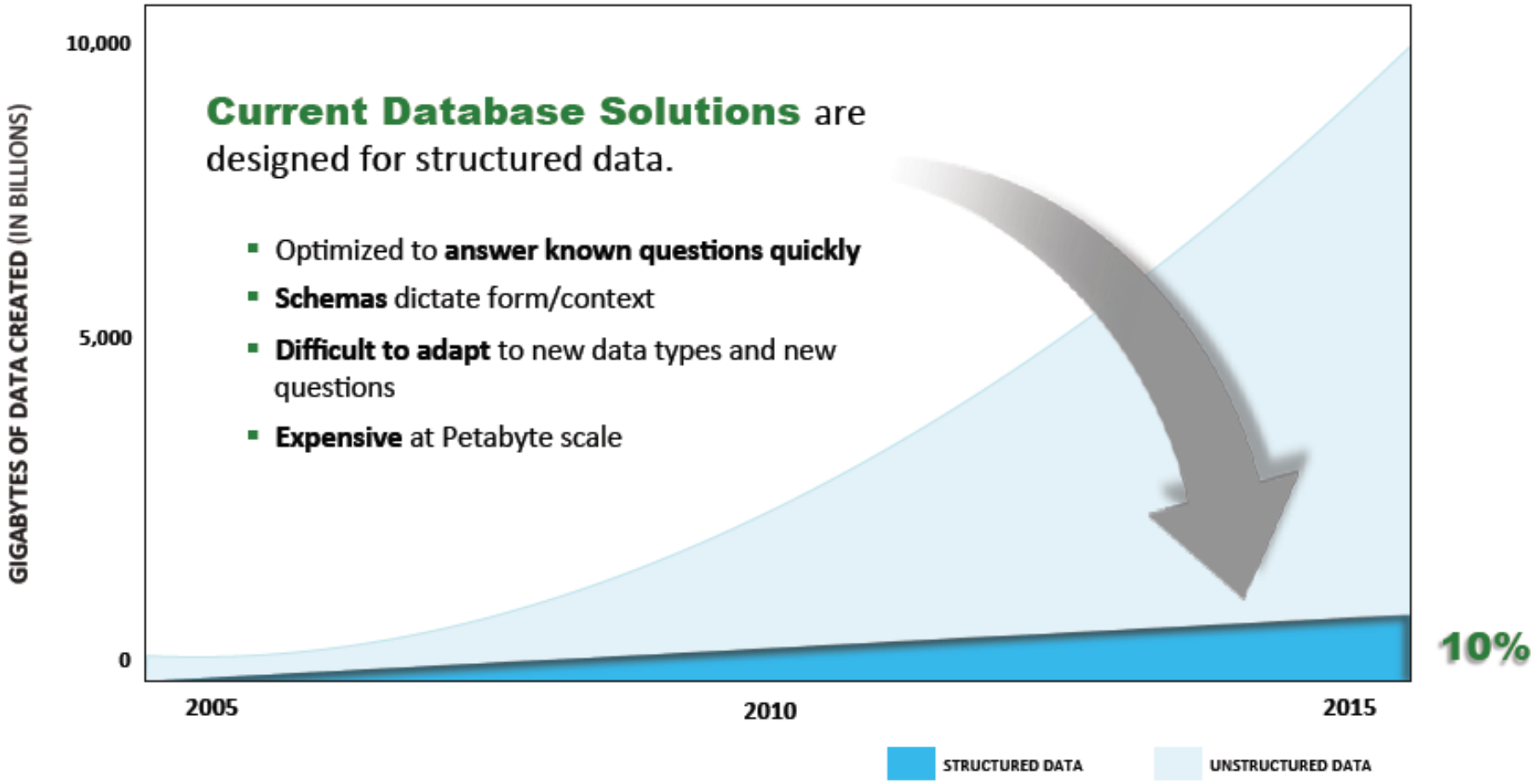
Source: IDC

By 2015, organizations that build a modern information management system will outperform their peers financially by 20 percent.

– Gartner, Mark Beyer, "Information Management in the 21st Century"


BY THE NUMBERS: Big Data
25 quintillion bytes
Amount of data generated every day through online or mobile financial transactions, social media traffic, and GPS, among other sources. †
90%
Share of the world's data created in the last two years ‡
1.5m
Number of additional data-literate managers and analysts needed in the U.S. by 2018 ††
20
Number of months it is expected to take for digital data to continually double between now and 2020 ††
$3.3m
Estimated revenue for worldwide Big Data storage services by 2016 ‡‡
SOURCES
† IBM, Bringing big data to the enterprise
‡ McKinsey Global Institute "Big data: The next frontier for innovation, competition, and productivity," May, 2011
†† Global Pulse, "Big Data for Development: Opportunities & Challenges," May, 2012
‡‡ IDC, "Worldwide Storage Services Opportunity for Big Data 2012-2016 Forecast," 2012

Traditional Solutions



Inflection Points

- Is divide the data and rule a solution here?
  - Have a multiple disk drive , split your data file into small enough pieces across the drives and do parallel reads and processing.
  - Hardware Reliability (Failure of any drive) is a challenge.
  - Resolving data interdependency between drives is a notorious challenge.
  - Number of disk drives that can be added to a server is limited

Inflection Points

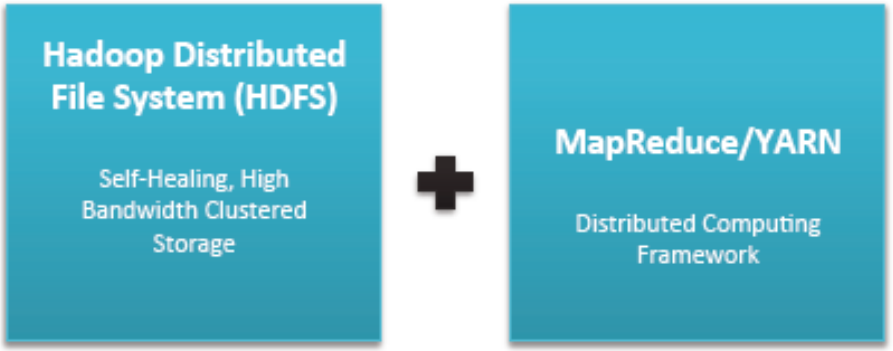
- We need a Drastically different approach
  - A distributed file system with high capacity and high reliability.
  - A process engine that can handle structure /unstructured data.
  - A computation model that can operate on distributed data and abstracts data dispersion.

What is Apache Hadoop

Apache Hadoop is an open source platform for data storage and processing that is...

- ✓ Scalable
- ✓ Fault tolerant
- ✓ Distributed

CORE HADOOP SYSTEM COMPONENTS



Has the Flexibility to Store and Mine Any Type of Data

- Ask questions across structured and unstructured data that were previously impossible to ask or solve
- Not bound by a single schema

Excels at Processing Complex Data

- Scale-out architecture divides workloads across multiple nodes
- Flexible file system eliminates ETL bottlenecks

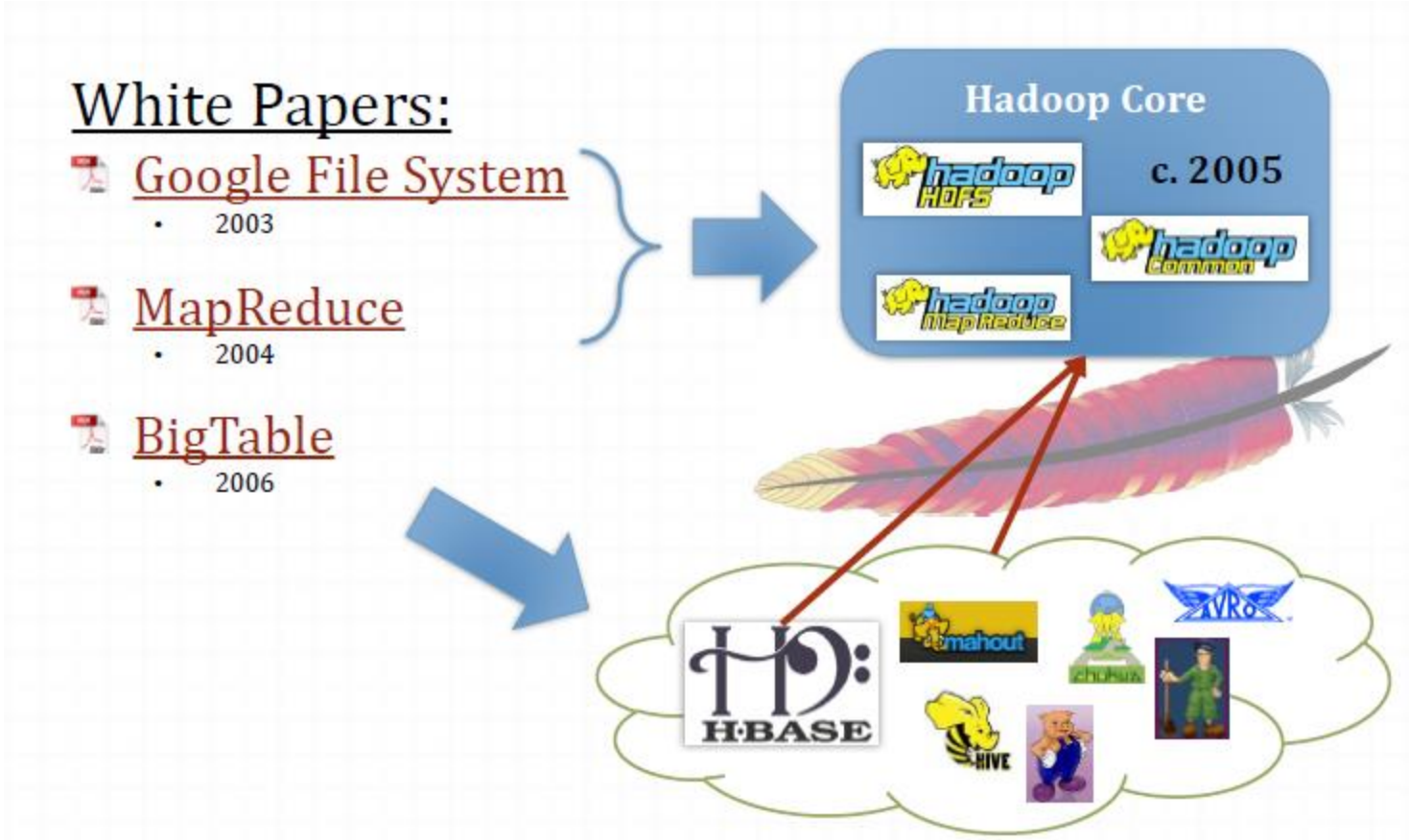
Scales Economically

- Can be deployed on commodity hardware
- Open source platform guards against vendor lock

What is Hadoop?

- “framework for running [distributed] applications on large cluster built of commodity hardware” –from Hadoop Wiki
- Originally created by Doug Cutting
  - Named the project after his son’s toy
- The name “Hadoop” has now evolved to cover a family of products, but at its core, it’s essentially just the MapReduce programming paradigm + a distributed file system

History


White Papers:
Google File System
2003
MapReduce
2004
BigTable
2006
Hadoop Core
c. 2005
hadoop HDFS
hadoop common
hadoop MapReduce
HBASE
mahout
HIVE
AVRO

Core Hadoop : HDFS

Self-healing, high bandwidth **clustered storage.**

HDFS breaks incoming files into blocks and stores them redundantly across the cluster.

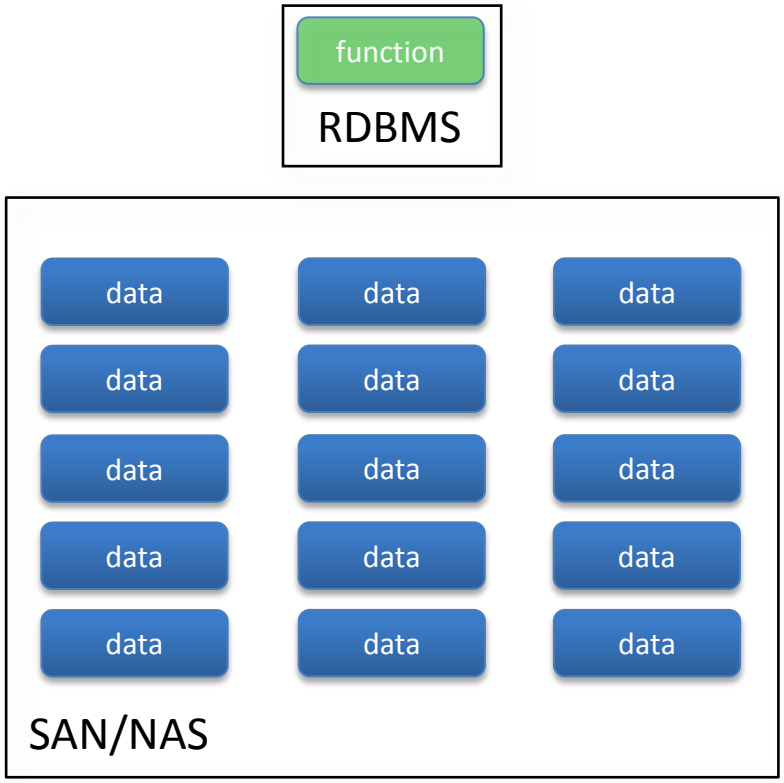
Core Hadoop: MapReduce

Distributed computing framework.

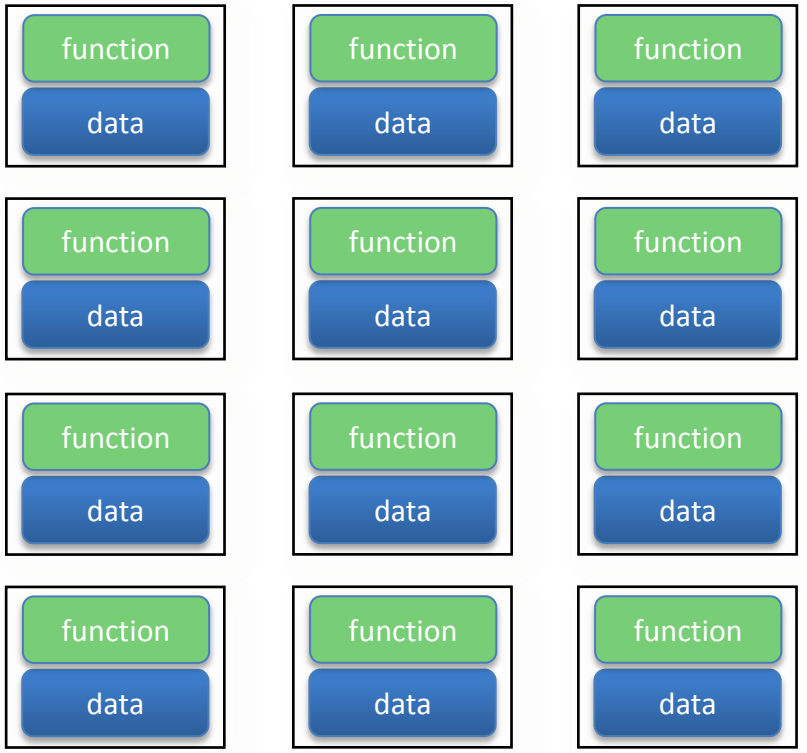
Processes large jobs in parallel across many nodes and combines the results.

Ship the Function to the Data

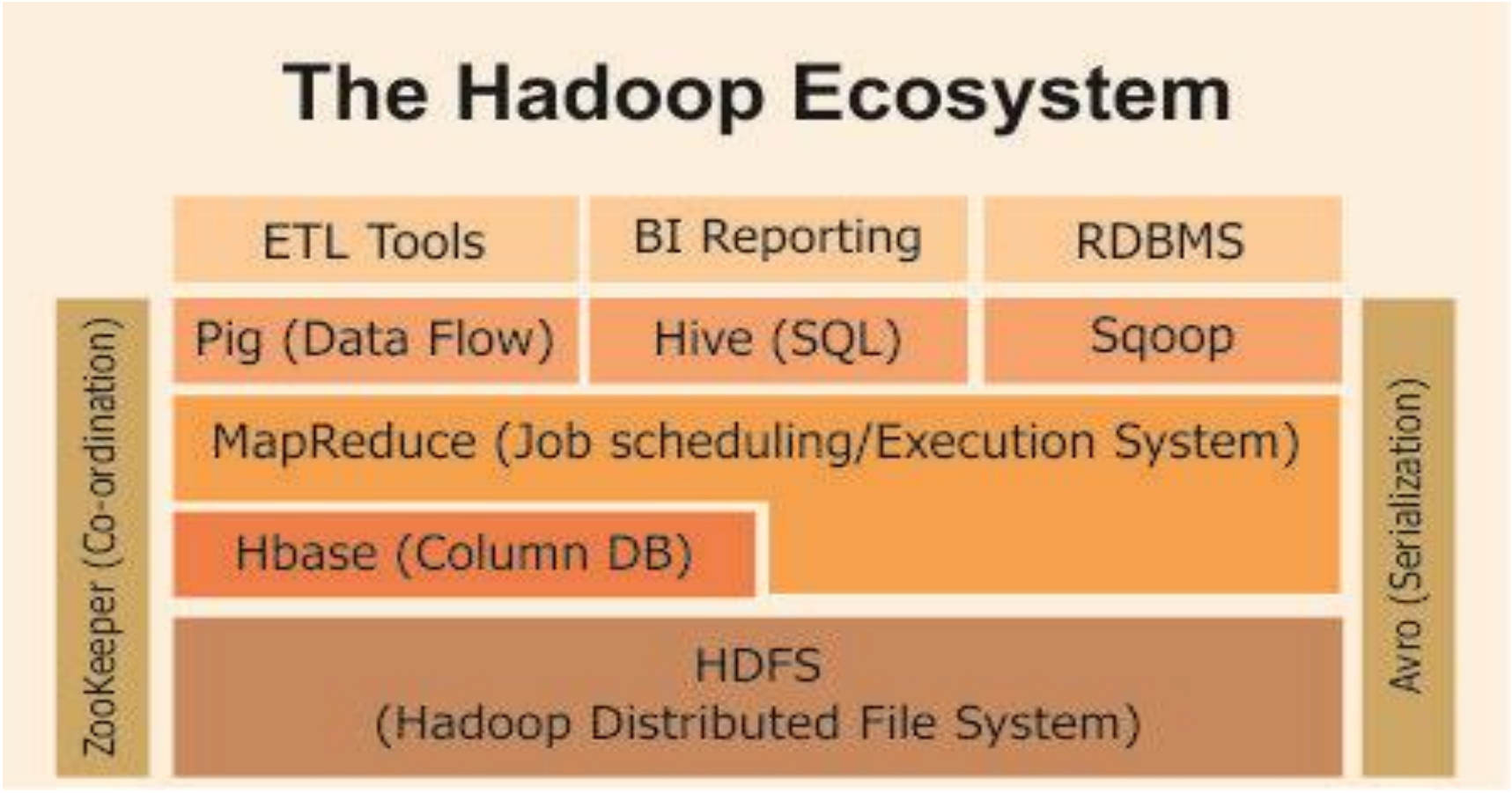
Traditional Architecture



Distributed Computing



Hadoop and it’s ecosystem



Pig

A platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.

Mahout

A machine learning library with algorithms for clustering, classification and batch based collaborative filtering that are implemented on top of Apache Hadoop.

Hive

Data warehouse software built on top of Apache Hadoop that facilitates querying and managing large datasets residing in distributed storage.

Sqoop

A tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases.

Apache Flume

A distributed service for collecting, aggregating, and moving large log data amounts to HDFS.

Apache HBase

HBase is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java. It runs on top of HDFS.

Big Data Landscape

Log Data Apps

splunk>
loggly
sumologic

Vertical Apps

PREDICTIVE POLICING
bloomreach. GET FOUND.

Business Intelligence

ORACLE | Hyperion
SAP BusinessObjects
Microsoft | Business Intelligence
IBM COGNOS
sas MicroStrategy
GoodData

Analytics and Visualization

tableau
Palantir
METAMARKETS
TERADATA ASTER
visual.ly
KARMASPHERE
EMC² GREENPLUM.
platfora
ClearStory DATA Now You See It

Data Providers

GNIP
DATASIFT
SPACE CURVE
INRIX

Analytics Infrastructure

Hortonworks
cloudera
MAPR TECHNOLOGIES
VERTICA An HP Company

Operational Infrastructure

CouchBase
TERADATA
10gen the MongoDB company
HADAPT

Infrastructure As A Service

amazon web services
infochimps
Windows Azure

Structured Databases

ORACLE
MySQL
Microsoft SQL Server
PostgreSQL
memsql

Technologies

hadoop
MapReduce
APACHE HBASE
Cassandra

dave@vcdave.com
http://blogs.forbes.com/davefeinleib/

Big Data Application Domains

- Healthcare
- The public sector
- Retail
- Manufacturing
- Personal-location data
- Finance
- Telecom

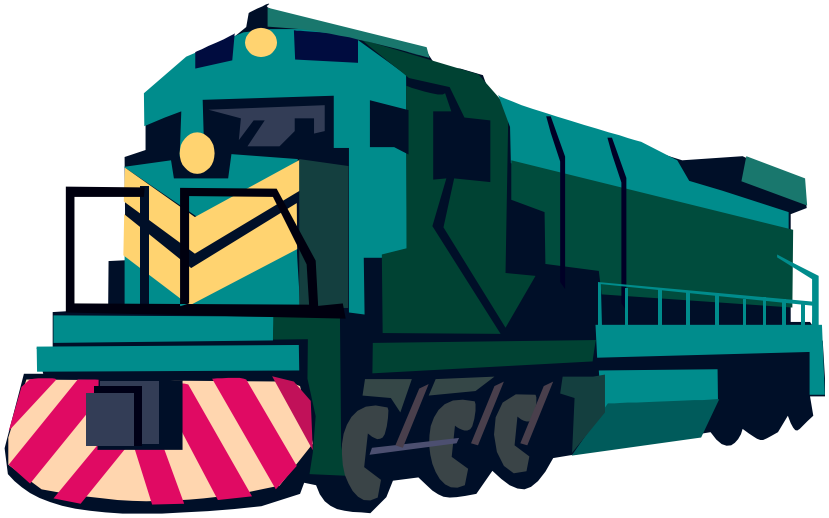
Use The Right Tool For The Right Job

Relational Databases:

When to use?

- Interactive Reporting (<1sec)
- Multistep Transactions
- Lots of Inserts/Updates/Deletes

Hadoop:



When to use?

- Affordable Storage/Compute
- Structured or Not (Agility)
- Resilient Auto Scalability

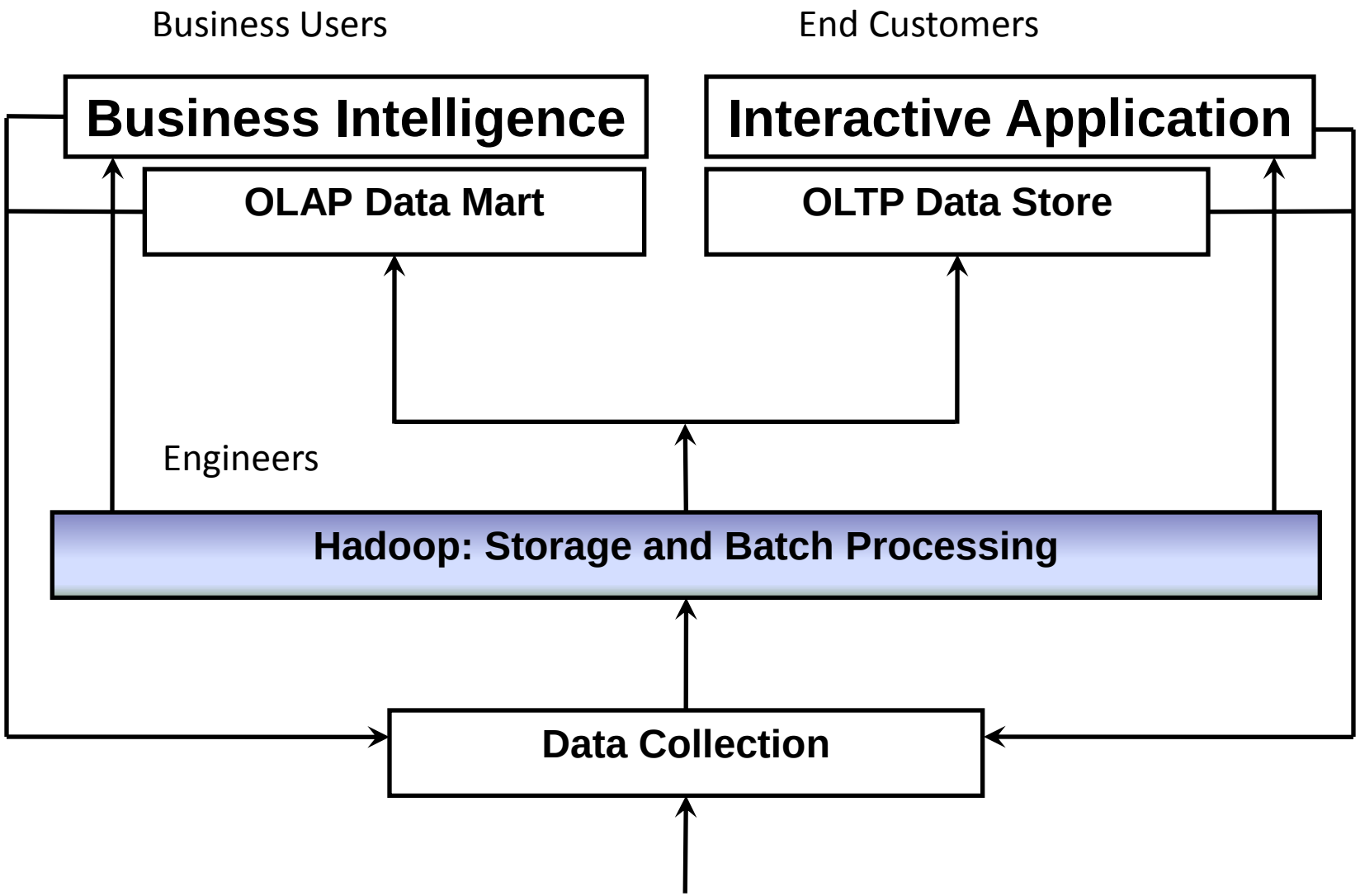
Economics of Hadoop Storage

- **Typical Hardware:**
  - Two Quad Core Nehalems
  - 24GB RAM
  - 12 * 1TB SATA disks (JBOD mode, no need for RAID)
  - 1 Gigabit Ethernet card
- **Cost/node:** $5K/node
- **Effective HDFS Space:**
  - ¼ reserved for temp shuffle space, which leaves 9TB/node
  - 3 way replication leads to 3TB effective HDFS space/node
  - But assuming 7x compression that becomes ~ 20TB/node

Effective Cost per user TB: $250/TB

Other solutions cost in the range of $5K to $100K per user TB

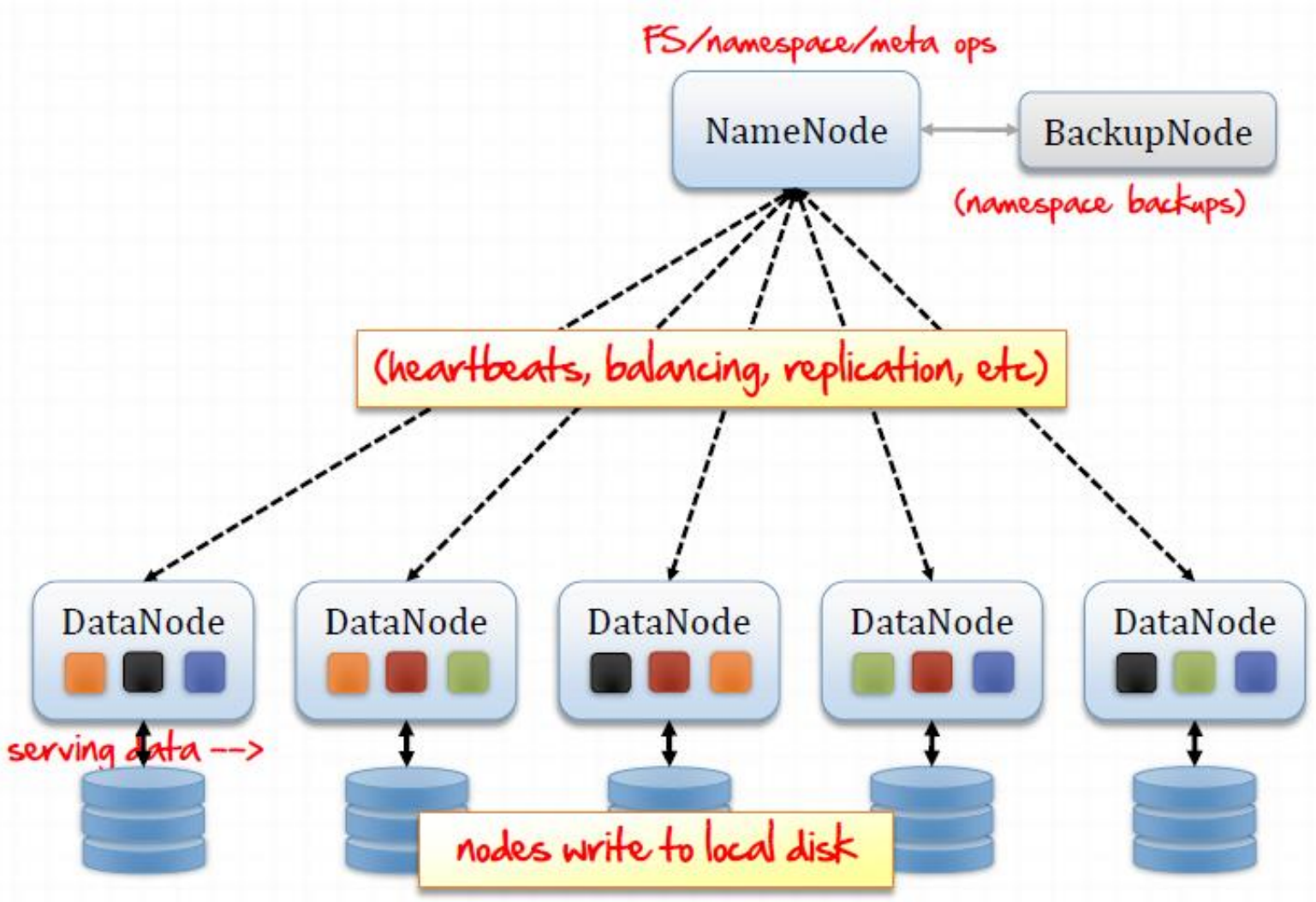
Typical Hadoop Architecture



HDFS Introduction

- Written in Java
- Optimized for larger files
  - Focus on streaming data
    (high-throughput > low-latency)
- Rack-aware
- Only *nix for production env.
- Web consoles for stats

HDFS Architecture



Lets us see what MapReduce is

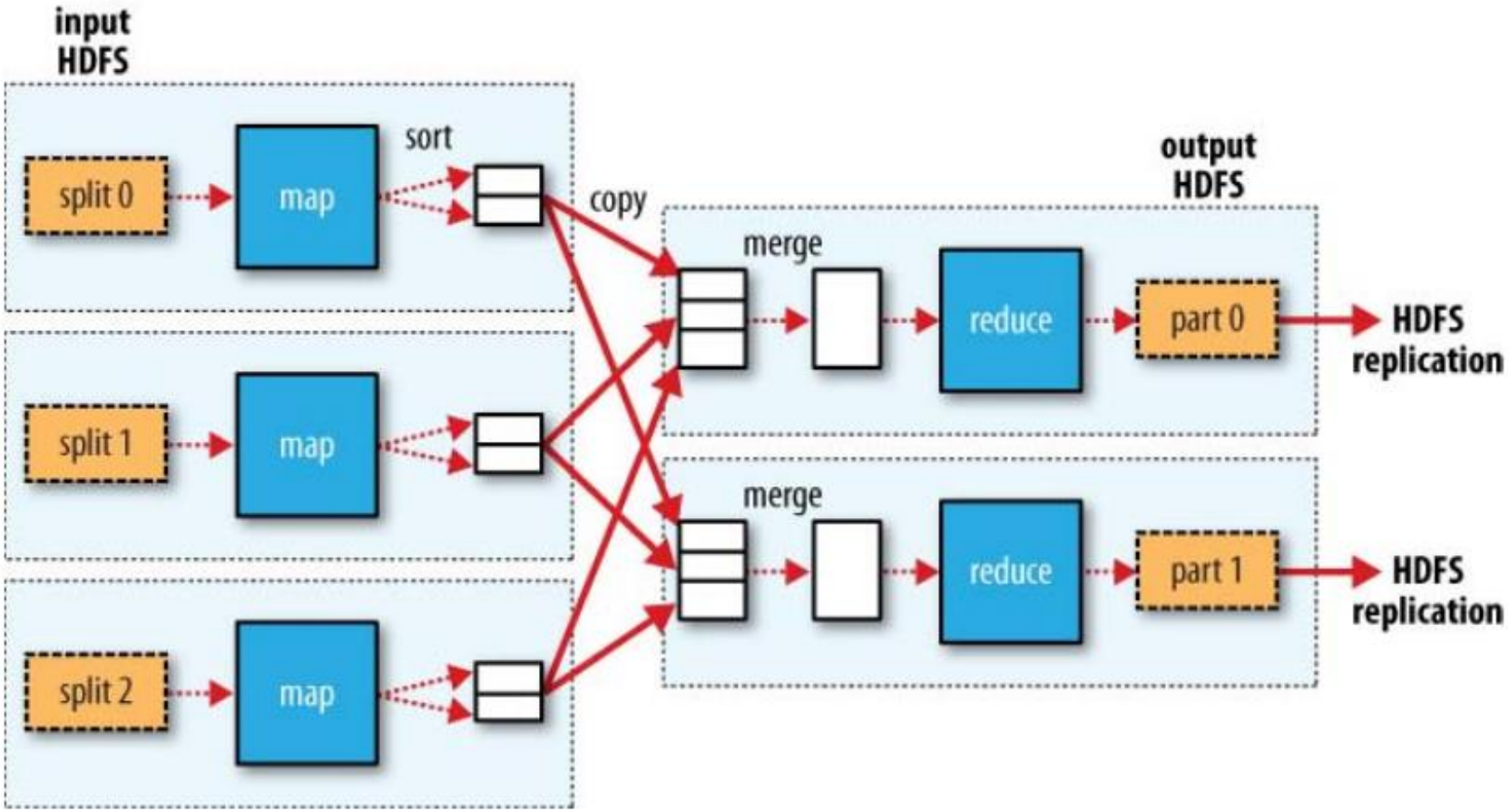
MapReduce basics

- Take a large problem and divide it into sub-problems
- Perform the same function on all sub-problems
- Combine the output from all sub-problems

MapReduce(M/R) facts

- M/R is excellent for problems where the “sub-problems” are **not** interdependent
- For example, the output of one “mapper” should *not* depend on the output or communication with another “mapper”
- The reduce phase does not begin execution until all mappers have finished
- Failed map and reduce tasks get auto-restarted
- Rack/HDFS-aware

What is MapReduce Model



What is the MapReduce Model

- MapReduce is a computation model that supports **parallel processing on distributed data** using clusters of computers.

- The **MapReduce** model expects the input data to be split and distributed to the machines on the clusters so that each splits can be processed **independently and in parallel**.

- There are two stages of processing in MapReduce model to achieve the final result: **Map** and **Reduce**. Every machine in the cluster can run independent map and reduce processes.

What is MapReduce Model

- **Map phase** processes the input splits. The output of the Map phase is distributed again to reduce processes to combine the map output to give final expected result.

- The model treats data at every stage as a **key and value pair**, transforming one set of key/value pairs into different set of key/value pairs to arrive at the end result.

- **Map process** tranforms input key/values pairs to a set of **intermediate key/vaue pairs**.

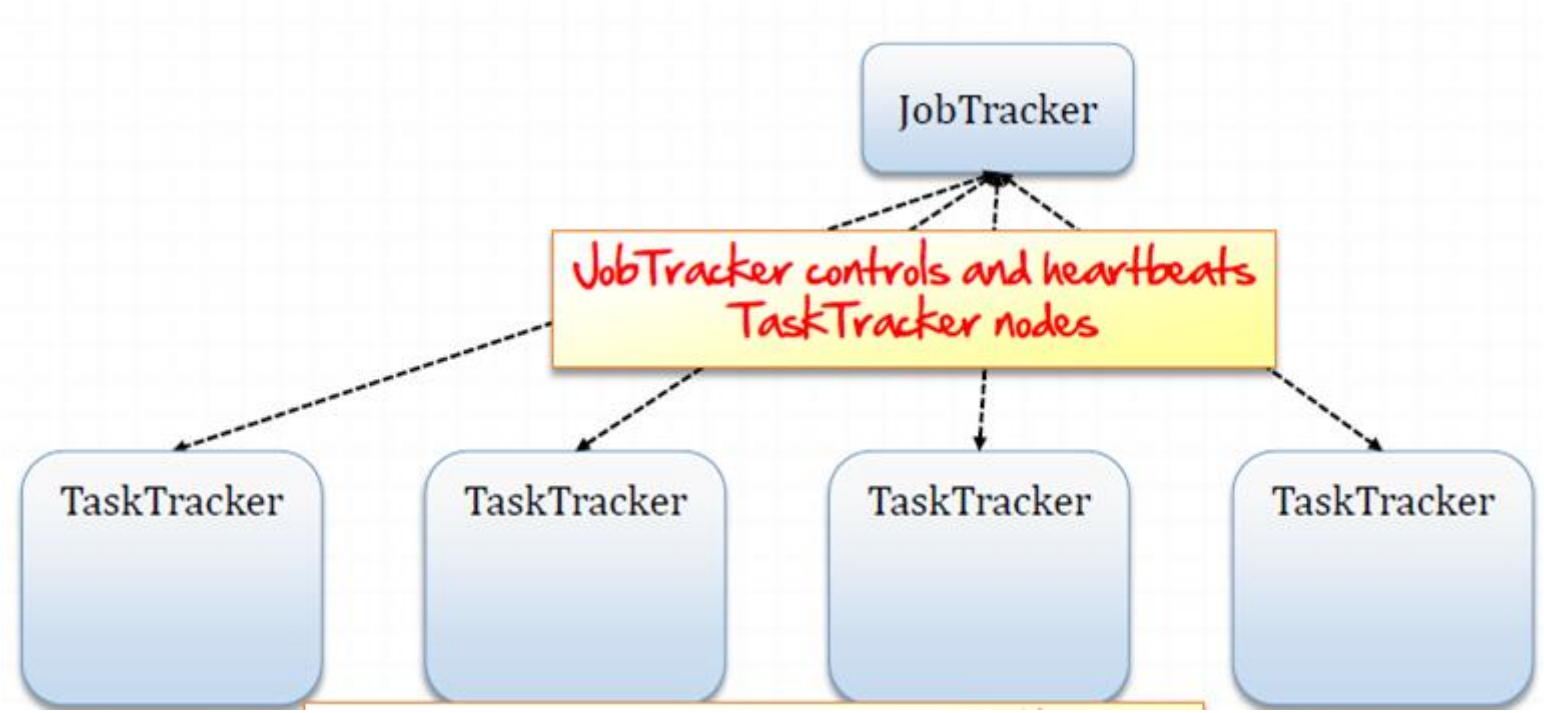
- **MapReduce framework** passes this output to reduce processes which will transform this to get final result which again will be in the form of key/value pairs.

Design of MapReduce -Daemons

The MapReduce system is managed by two daemons

- JobTracker & TaskTracker
- JobTracker TaskTracker function in master / slave fashion
  - JobTracker coordinates the entire job execution
  - TaskTracker runs the individual tasks of map and reduce
  - JobTracker does the bookkeeping of all the tasks run on the cluster
  - One map task is created for each input split
  - Number of reduce tasks is configurable (mapred.reduce.tasks)

Conceptual Overview



Design of HDFS

- HDFS is a Hadoop's Distributed File System
- Designed for storing very large files(Petabytes)
- Single file can be stored across several disks
- Not suitable for low latency data access
- Designed to be highly fault tolerant hence can run on commodity hardware

HDFS Concepts

- Like any file system HDFS stores files by breaking it into smallest units called Blocks.
- The default HDFS block size is 64 MB
- The large block size helps in maintaining high throughput
- Each block is replicated across multiple machines in the cluster for redundancy.

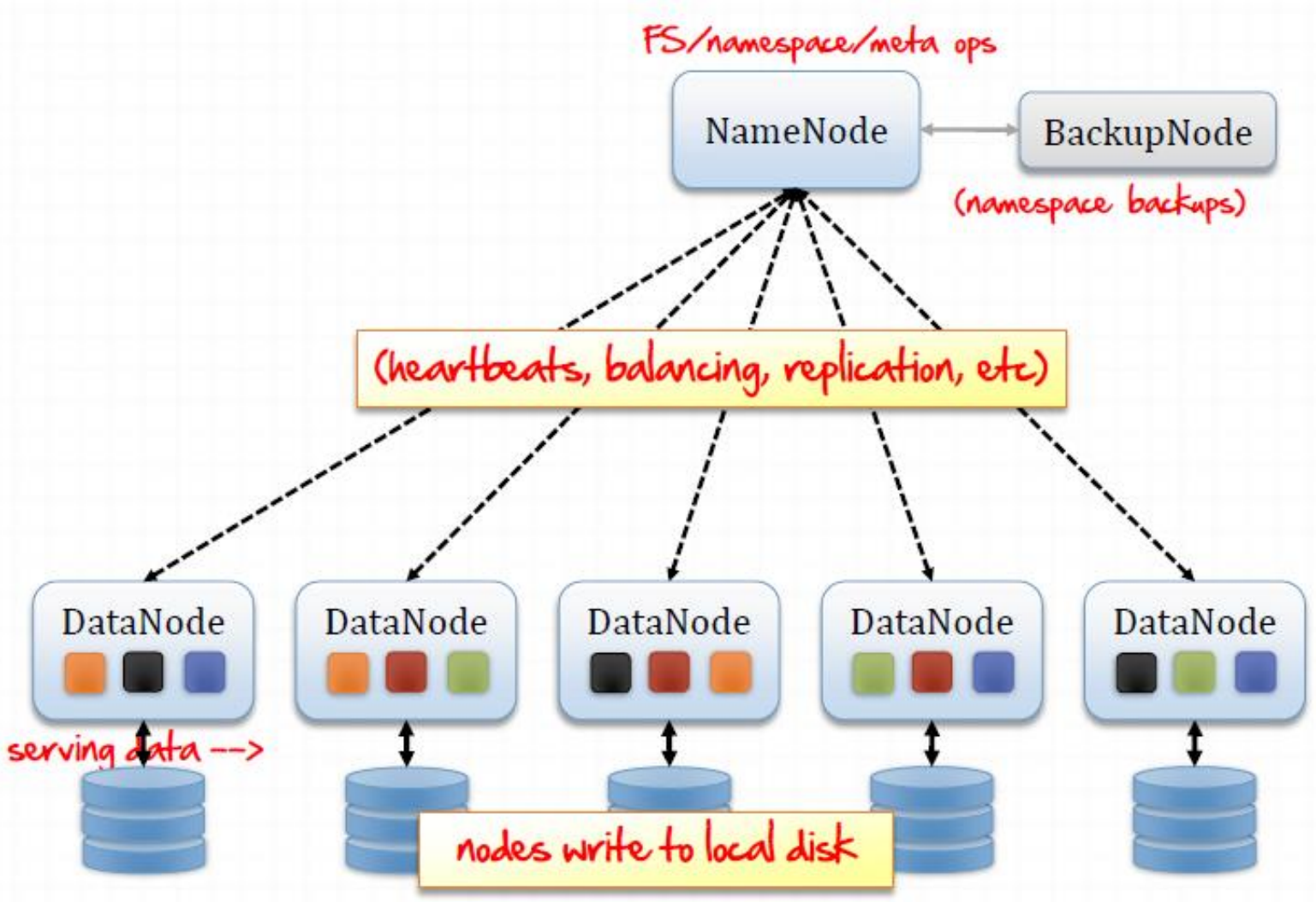
Design of HDFS -Daemons

- The HDFS file system is managed by two daemons
- NameNode and DataNode
- NameNode and DataNode function in master/slave fashion
  - NameNode Manages File system namespace
  - Maintains file system and metadata of all the files and directories
    - Namespace image
    - Edit log

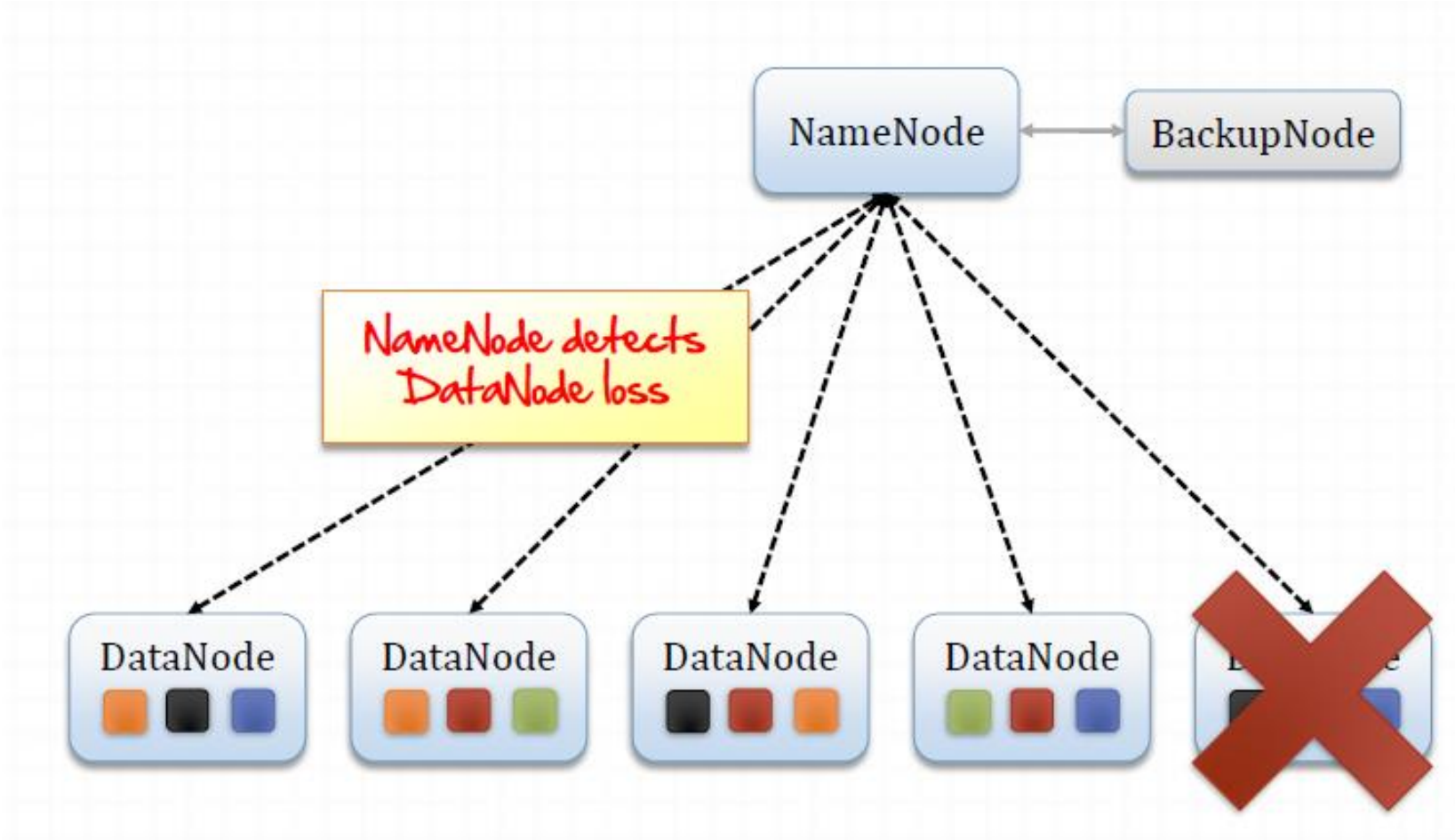
Design of HDFS –Daemons (cont.)

- Datanodes store and retrieve the blocks for the files when they are told by NameNode.
- NameNode maintains the information on which DataNodes all the blocks for a given files are located.
- DataNodes report to NameNode periodically with the list of blocks they are storing.
- With NameNode off ,the HDFS is inaccessible.
- Secondary NameNode
  - Not a backup for NameNode
  - Just helps in merging namespace image with edit log to avoid edit log becoming too large

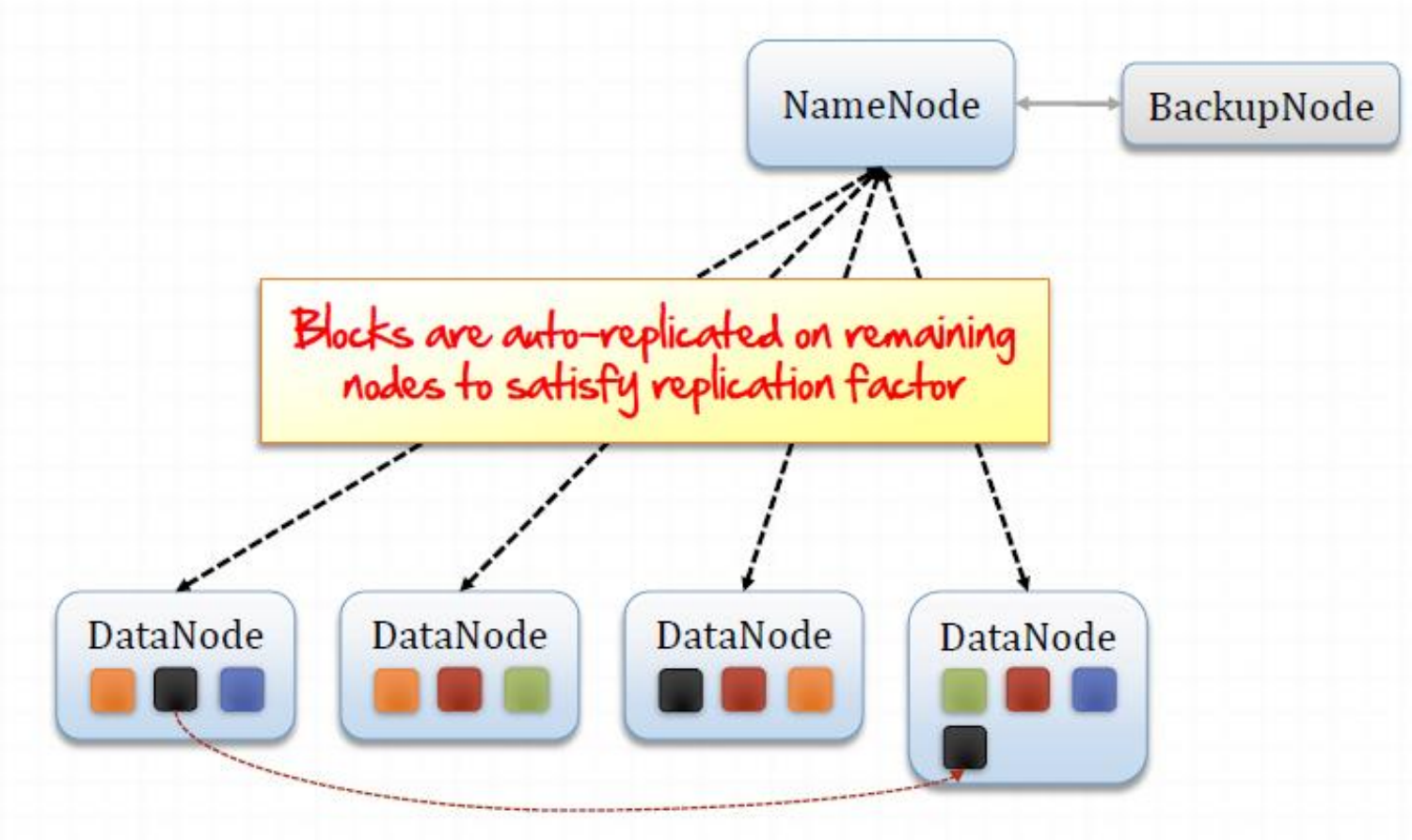
HDFS Architecture



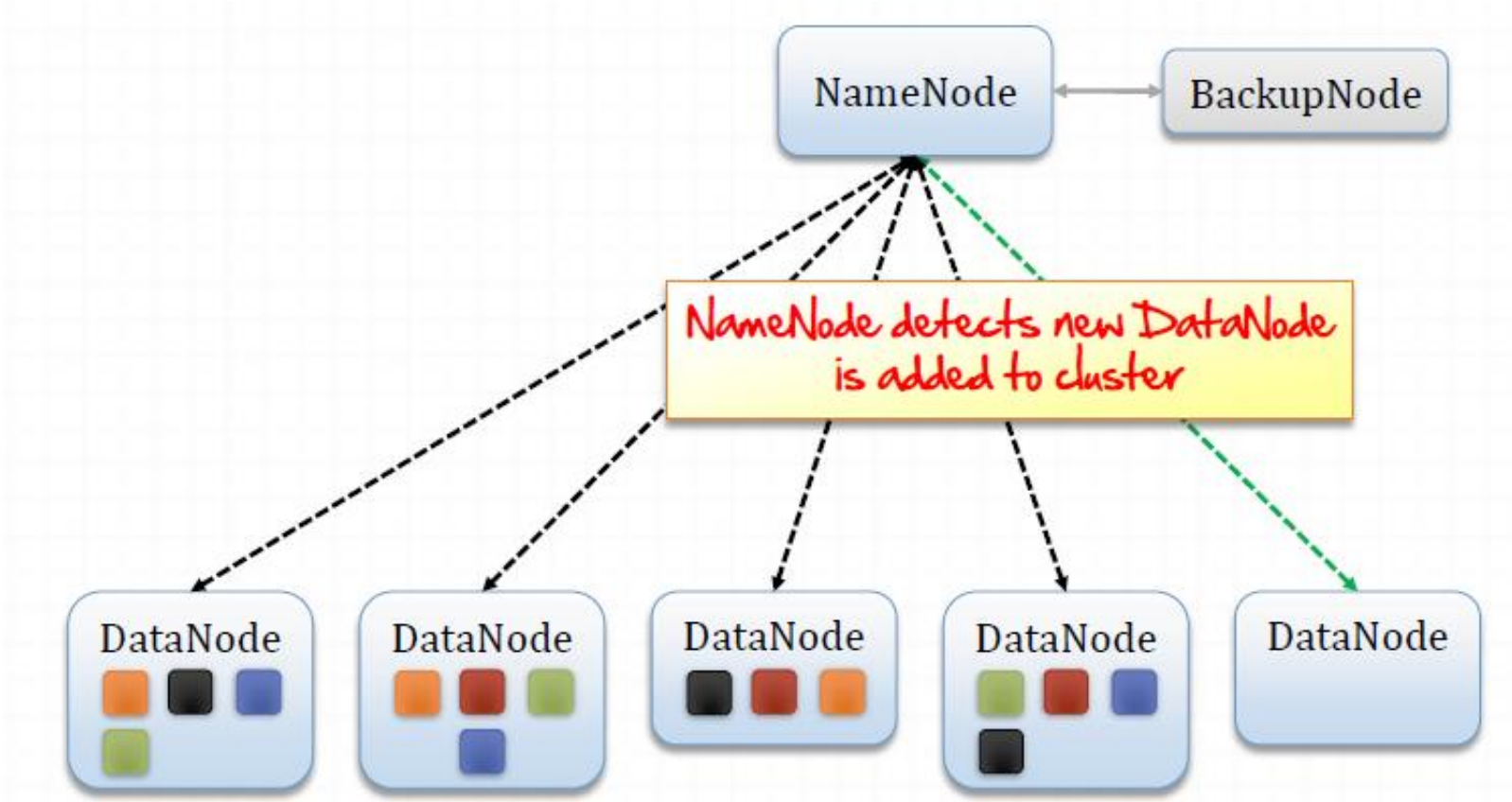
Fault Tolerance



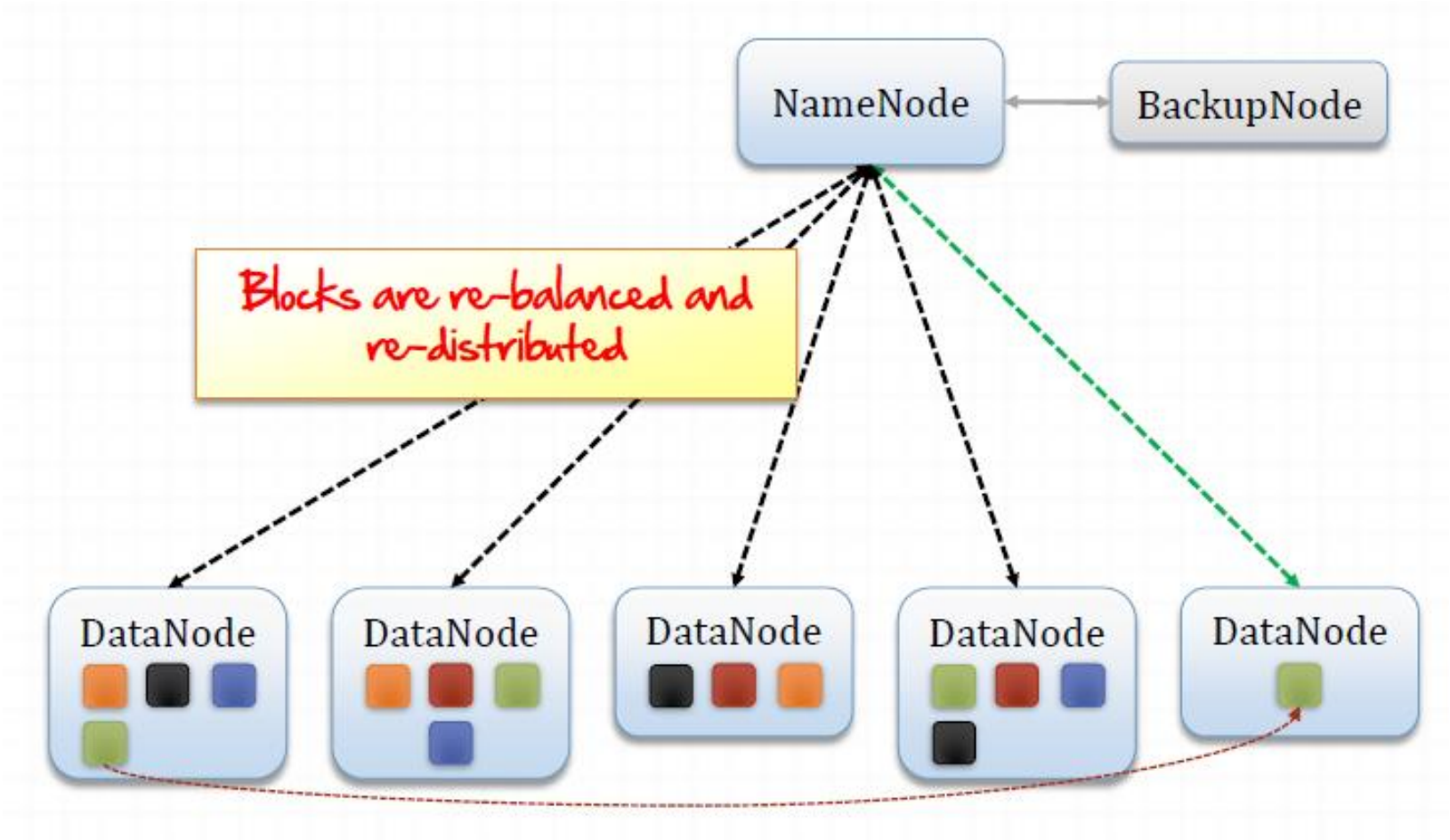
Fault Tolerance



Live Horizontal Scaling + Rebalancing



Live Horizontal Scaling + Rebalancing



Who Loves it

- Yahoo !! Runs 20,000 servers running Hadoop
- Largest Hadoop clusters is 4000 servers,16PB raw storage
- Facebook runs 2000 Hadoop servers
- 24 PB raw storage and 100 TB raw log/day
- eBay and LinkedIn has production use of Hadoop.
- Sears retails uses Hadoop.

Hadoop Requirements

- Supported Platforms
  - GNU/Linux is supported as development and production
- Required Software
  - java 1.6.x +
  - ssh to be installed ,sshd must be running (for machined in the cluster to interact with master machines)
- Development Environment
  - Eclipse 3.6 or above

Lab Requirements

- Window 7-64 bit OS , Min 4 GB Ram
- VMWare Player5.0.0
- Linux VM-Ubuntu 12.04 LTS
  - **User: hadoop, password :hadoop123**
- Java 6 installed on Linux VM
- Open SSH installed on Linux VM
- Putty-For opening Telnet sessions to the Linux VM
- WinSCP- For transferring files between windows and Linux VM
- Other Linux machines will do as well
- Eclipse 3.6

Extract VM

- Folder:

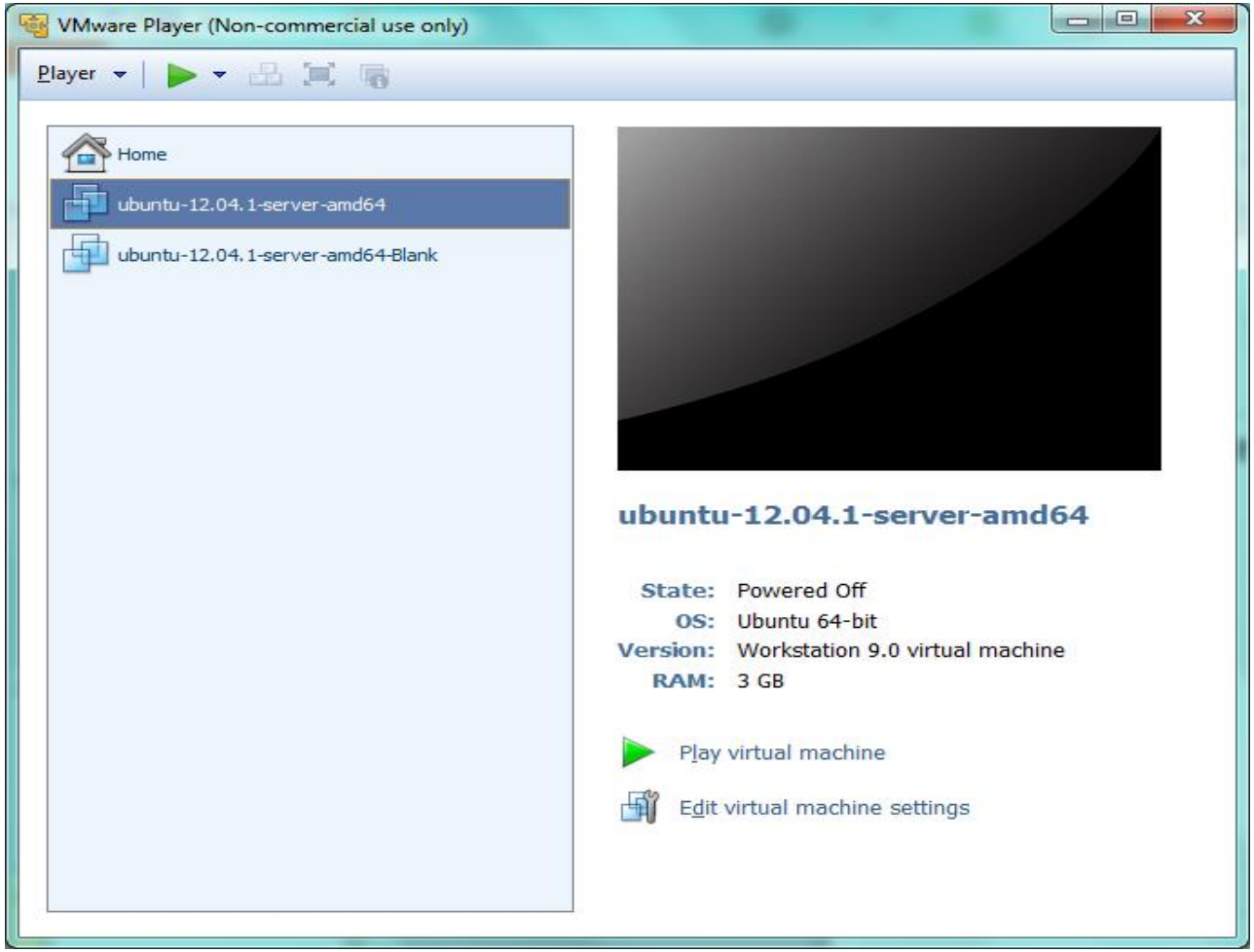
  Hadoop_MasterClass\Ubuntu_VM

- Copy:

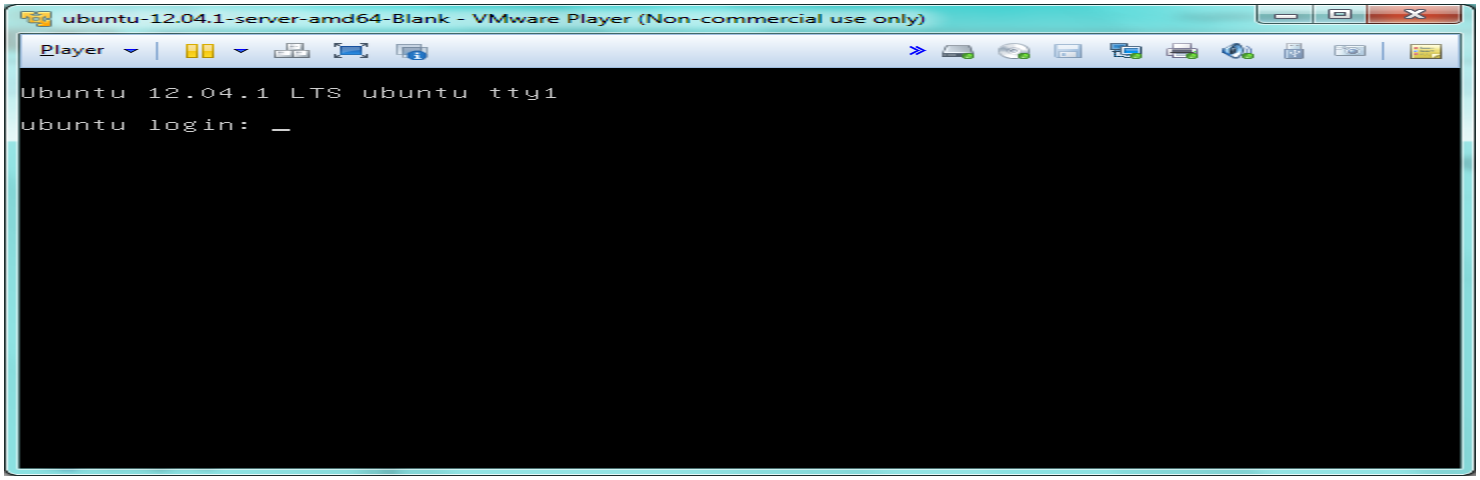
  ubuntu-12.04.1-server-amd64_with_hadoop.zip

- Extract to local using 7-zip

Starting VM


VMware Player (Non-commercial use only)
Player
Home
ubuntu-12.04.1-server-amd64
ubuntu-12.04.1-server-amd64-Blank
ubuntu-12.04.1-server-amd64
State: Powered Off
OS: Ubuntu 64-bit
Version: Workstation 9.0 virtual machine
RAM: 3 GB
Play virtual machine
Edit virtual machine settings

Starting VM



- Enter userID/password
- Type ifconfig
  - Note down the ip address
  - Connect to the VM using Putty

Install and Configure ssh(non-VM users)

- Install ssh

  sudo apt –get install ssh

  – Check ssh installation: **which ssh**

  which sshd

  which ssh-keygen

  – Generate ssh Key

  ssh-keygen –t rsa –P ” –f ~/.ssh/id_rsa

  – Copy public key as an authorized key(equivalent to slave node)

  cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

  chmod 700 ~/.ssh

  chmod 600 ~/.ssh/authorized_keys

Install and Configure ssh

- Verify SSHby logging into target(localhost here)
  - Command:

    ssh localhost

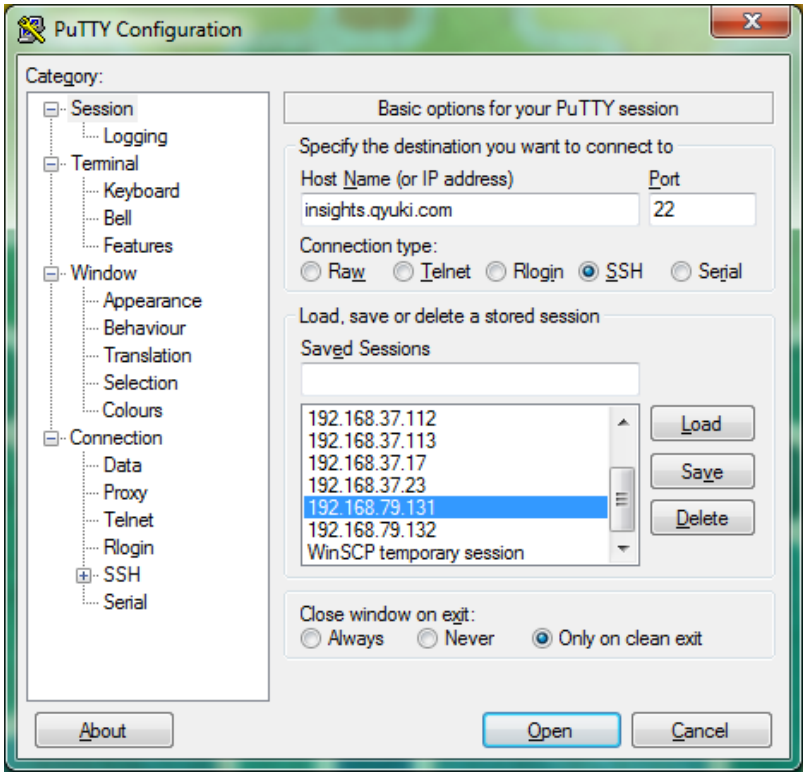
  - (this command will log you into the machine localhost)

Accessing VM Putty and WinSCP

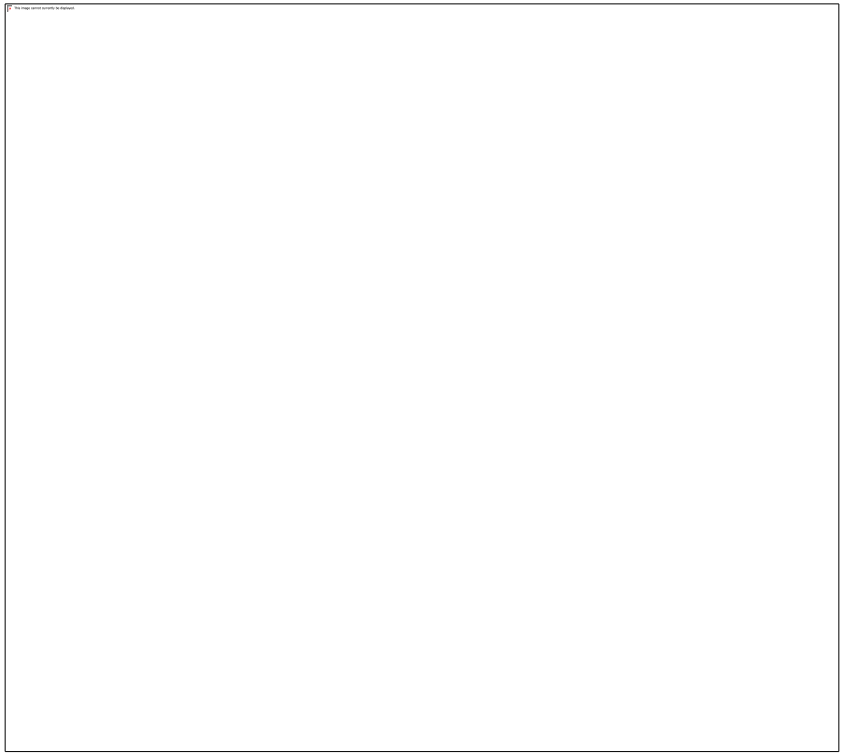
- Get IP address of the VM by using command in Linus VM
- Use Putty to telnet to VM
- Use WinSCP to FTP files to VM

Accesing VM using Putty & WinSCP

Putty



WinSCP



Install and Configure Java(non-VM users)

- Install Open JDK
  - Command :
  - **sudo apt-get install openjdk-6-jdk**
  - Check installation :java -version
- Configure java home in environment
  - Add a line to .bash_profile to set java Home
    - **export JAVA_HOME =/usr/lib/jvm/java-6-openjdk-amd64**
  - Hadoop will use during runtime

Install Hadoop(non-VM users)

- Download hadoop jar with **wget**
  - **http://archive.apache.org/dist/hadoop/core/hadoop-1.0.3/hadoop-1.0.3.tar.gz**

- Untar
  - **cd ~/lab/install**
  - **tar xzf ~/lab/downloads/hadoop-1.03.tar.gz**
  - Check extracted directory “hadoop-1.0.3”

- Configure environment in .bash_profile (or .bashrc)
  - Add below two lines and execute bash profile
    - **export HADOOP_INSTALL=~/lab/install/hadoop-1.0.3**
    - **export PATH=$PATH:$HADOOP_INSTALL/bin**
    - **. .bash_profile** (Execute bashrc)

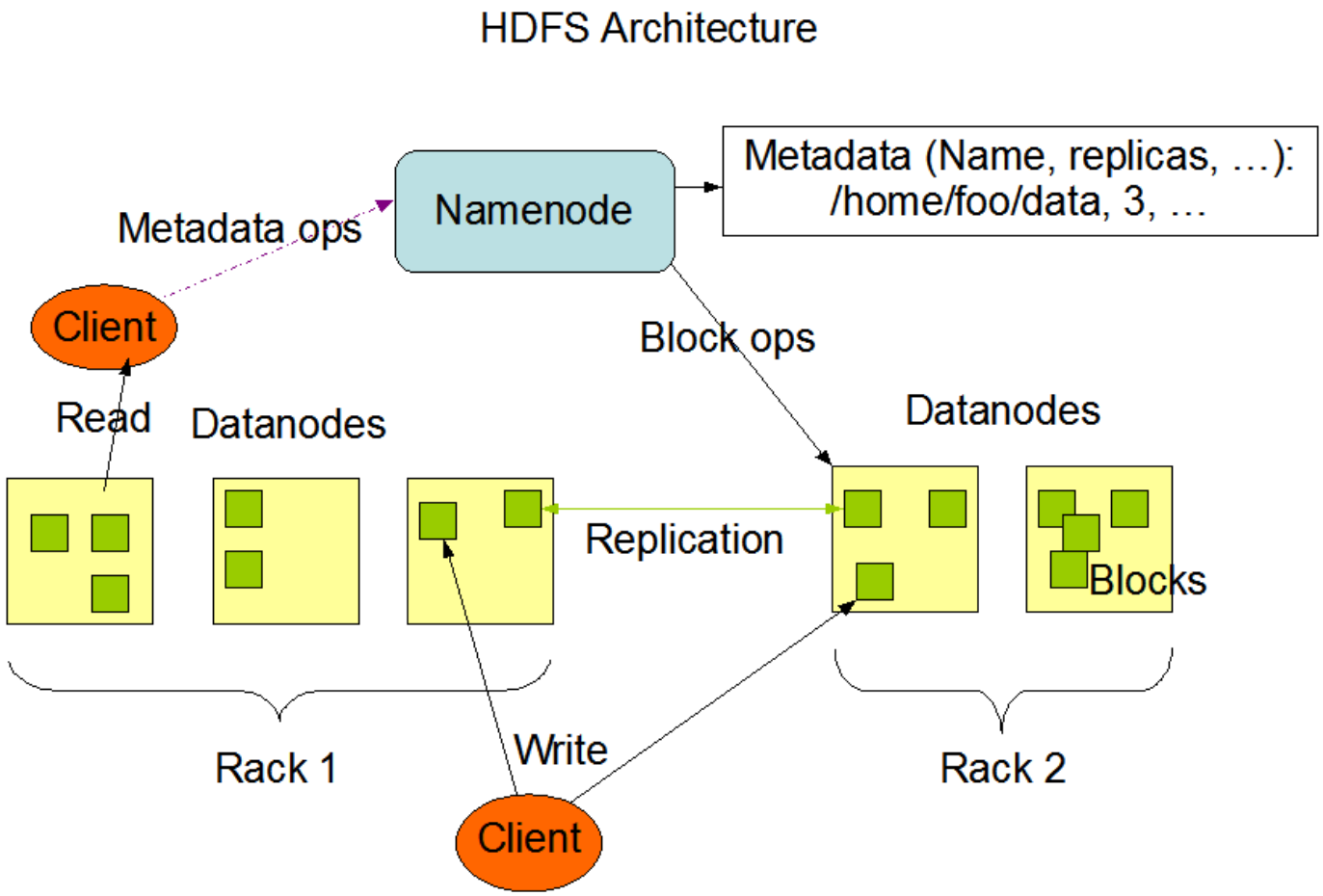
- Check Hadoop installation
  - **hadoop version**

Setting up Hadoop

- Open $HADOOP_HOME/conf/hadoop-env.sh
- set the JAVA_HOME environment variable to the $JAVA_HOME directory.

export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64

Component of core Hadoop



Component of core Hadoop

- At a high-level Hadoop architecture components can be classified into two categories
  - Distributed File management system-HDFS
- This has central and distributed sub components
    - NameNode: Centrally Monitors and controls the whole file system
    - DataNode: Take care of the local file segments and constantly communicates with NameNode
    - Secondary NameNode:Do not confuse. This is not a NameNode Backup. This just backs up the file system status from the NameNode periodically.
- Distributed computing system-MapReduce Framework
- This again has central and distributed sub components
    - Job tracker: Centrally monitors the submitted Job and controls all processes running on the nodes(computers) of the clusters.This communicated with Name Node for file system access
    - Task Tracker: Take care of the local job execution on the local file segments. This talks to DataNode for file information. This constantly communicates with job Tracker daemon to report the task progress.
- When the Hadoop system is running in a distributed mode all the daemons would be running in the respective computer.

Hadoop Operational Modes

Hadoop can be run in one of the three modes

- Standalone(Local) Mode
  - No daemons launched
  - Everything runs in single JVM
  - Suitable for development

- Pseudo Distributed Mode
  - All daemons are launched
  - Runs on a single machine thus simulating a cluster environment
  - Suitable for testing and debugging

- Fully Distributed Mode
  - The Hadoop daemons runs in a cluster environment
  - Each daemons run on machine respectively assigned to them
  - Suitable for integration Testing/Production

Hadoop Configuration Files

Filename	Format	Description
hadoop-env.sh	Bash script	Environment variables that are used in the scripts to run Hadoop
core-site.xml	Hadoop configuration.XML	Configuration settings for Hadoop Core, such as I/O settings that are common to HDFS and MapReduce
hdfs-site.xml	Hadoop configuration.XML	Configuration settings for HDFS daemons: the namenode, the secondary namenode, and the datanodes
mapred-site.xml	Hadoop configuration.XML	Configuration settings for MapReduce daemons: the jobtracker, and the tasktrackers
masters	Plain text	A list of machines (one per line) that each run a secondary namenode
slaves	Plain text	A list of machines (one per line) that each run a datanode and a tasktracker
hadoop-metrics.properties	Java Properties	Properties for controlling how metrics are published in Hadoop
log4j.properties	Java Properties	Properties for system logfiles, the namenode audit log, and the task log for the tasktracker child process

Configuring conf/*-site.xml files

- Need to set the hadoop.tmp.dir parameter to a directory of your choice.
- We will use /app/hadoop/tmp
- **sudo mkdir -p /app/hadoop/tmp**
- sudo chmod 777 /app/hadoop/tmp

Key Configuration Properties

Property Name	Conf File	Standalone	Pseudo Distributed	Fully Distributed
Fs.default.name	core -site.xml	File:///(default)	hdfs://localhost/	hdfs://namenode/
dfs.replication	hdfs -site.xml	NA	1	3(default)
mapred.job.tracker	mapred-site.xml	local(default)	localhost:8021	jobtracker:8021

Configuring HDFS: core-site.xml(Pseudo Distributed Mode)

```
<?xml version ="1.0"?>
<!--core-site.xml-->
<configuration>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/app/hadoop/tmp</value>
        </property>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://localhost/</value>
        </property>
</configuration>
```

Note: Add "fs.default.name" property under configuration tag to specify NameNode location.

"localhost" for pseudo distributed mode. Name node runs at port 8020 by default if no port specified.

Configuring mapred-site.xml(Pseudo Distributed Mode)

```
<?xml version="1.0"?>
<!— mapred-site.xml —>
<configuration>
        <property>
                <name>mapred.job.tracker</name>
                <value>localhost:8021</value>
        </property>
</configuration>
```

Note:Add "mapred.job.tracker" property under configuration tag to specify JobTracker location.

"localhost:8021" for Pseudo distributed mode.

Lastly set JAVA_HOME in conf/hadoop-env.sh export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64

Configuring HDFS: hdfs-site.xml(Pseudo Distributed Mode)

- <?xml version ="1.0"?>
- <!--hdfs-site.xml-->

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

Starting HDFS

- Format NameNode

  hadoop namenode -format

  - creates empty file system with storage directories and persistent data structure
  - Data nodes are not involved
- Start dfs service
  - **start-dfs.sh**
  - Verify daemons :
    jps

    If you get the namespace exception, copy the namespace id of the namenode
    Paste it in the : **/app/hadoop/tmp/dfs/data/current/VERSION** file
  - Stop : **stop-dfs.sh**
- List/check HDFS

  hadoop fsck / -files -blocks

  hadoop fs -ls

  hadoop fs -mkdir testdir

  hadoop fs -ls

  hadoop fsck / -files -blocks

Verify HDFS

- Stop dfs services

  stop-dfs.sh

  Verify Daemons :jps

    No java processes should be running

Configuration HDFS -hdfs-site .xml(Pseudo Distributed Mode)

Property Name	Description	Default Value
dfs.name.dir	Directories for NameNode to store it's persistent data(Comma seperated directory name).A copy of metadata is stored in each of listed directory	${hadoop.tmp.dir}/dfs/name
dfs.data.dir	Directories where DataNode stores blocks.Each block is stored in only one of these directories	${hadoop.tmp.dir}/dfs/data
fs.checkpoint.dir	Directories where secondary namenode stores checkpoints.A copy of the checkpoint is stored in each of the listed directory	${hadoop.tmp.dir}/dfs/namesecondary

Basic HDFS Commands

- Creating Directory

  hadoop fs -mkdir <dirname>

- Removing Directory

  hadoop fs -rm <dirname>

- Copying files to HDFS from local file system

  hadoop fs -put <local dir>/<filename> <hdfs dir Name>/

- Copying files from HDFS to local file system

  hadoop fs -get <hdfs dir Name>/<hdfs file Name> <local dir>/

- List files and directories

  hadoop fs -ls <dir name>

- List the blocks that make up each files in HDFS

  hadoop fsck / -files -blocks

HDFS Web UI

- Hadoop provides a web UI for viewing HDFS
  - Available at **http://namenode-host-ip:50070/**
  - Browse file system
  - Log files

MapReduce

- A distributed parallel processing engine of Hadoop
- Processes the data in sequential parallel steps called
    - Map
    - Reduce
- Best run with a DFS supported by Hadoop to exploit it's parallel processing abilities
- Has the ability to run on a cluster of computers
    - Each computer called as a node
- Input and output data at every stage is handled in terms of key / value pairs
    - Key / Value can be choose by programmer
- Mapper output is always sorted by key
- Mapper output with the same key are sent to the same reducer
- Number of mappers and reducers per node can be configured

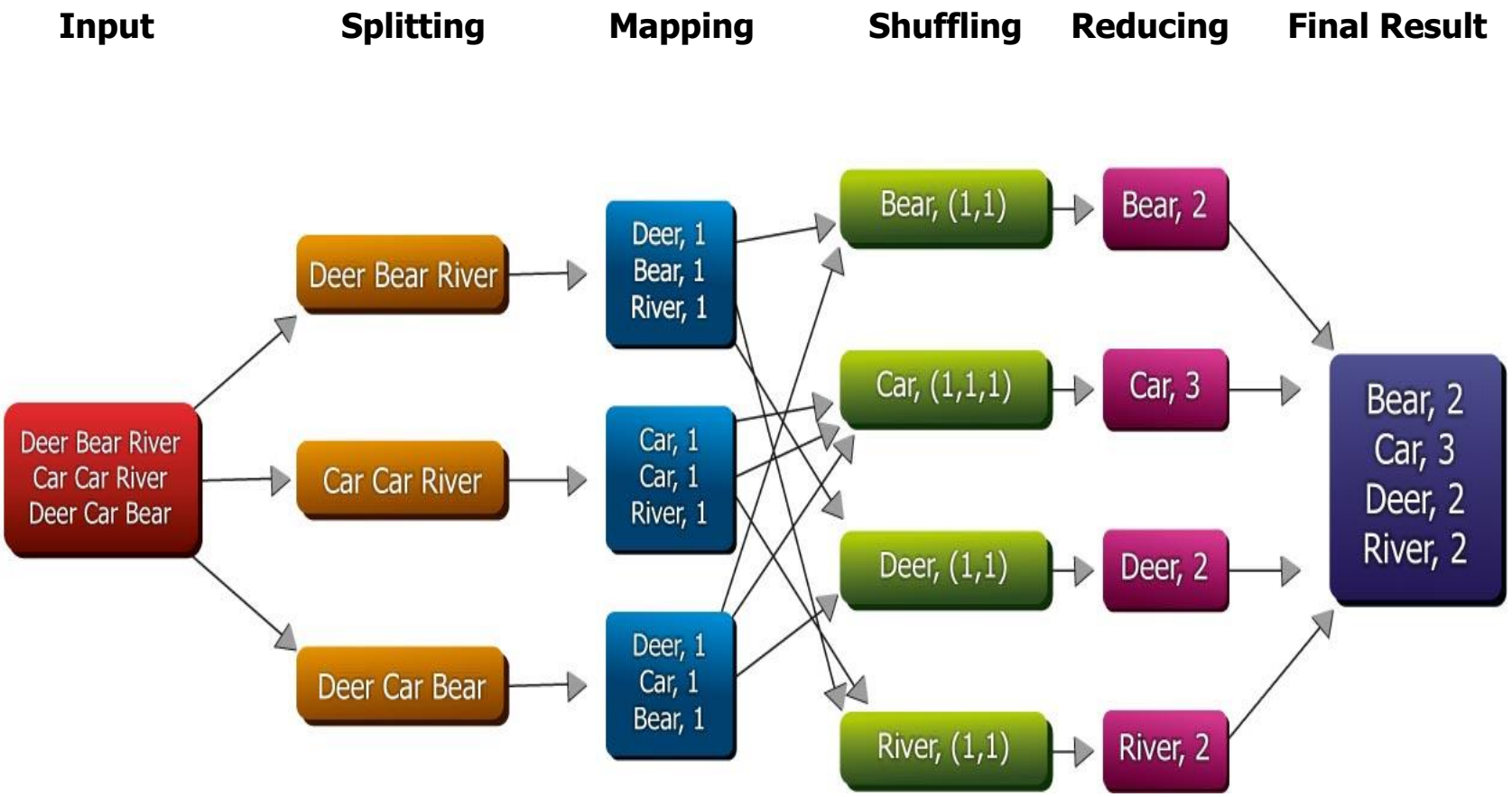
Start the Map reduce daemons

- **start-mapred.sh**

The MapReduce Web UI

- Hadoop provides a web UI for viewing job information
    - Available at http://jobtracker-host:50030/
    - follow job's progress while it is running
    - find job statistics
    - View job logs
    - Task Details

The Overall MapReduce Word count Process

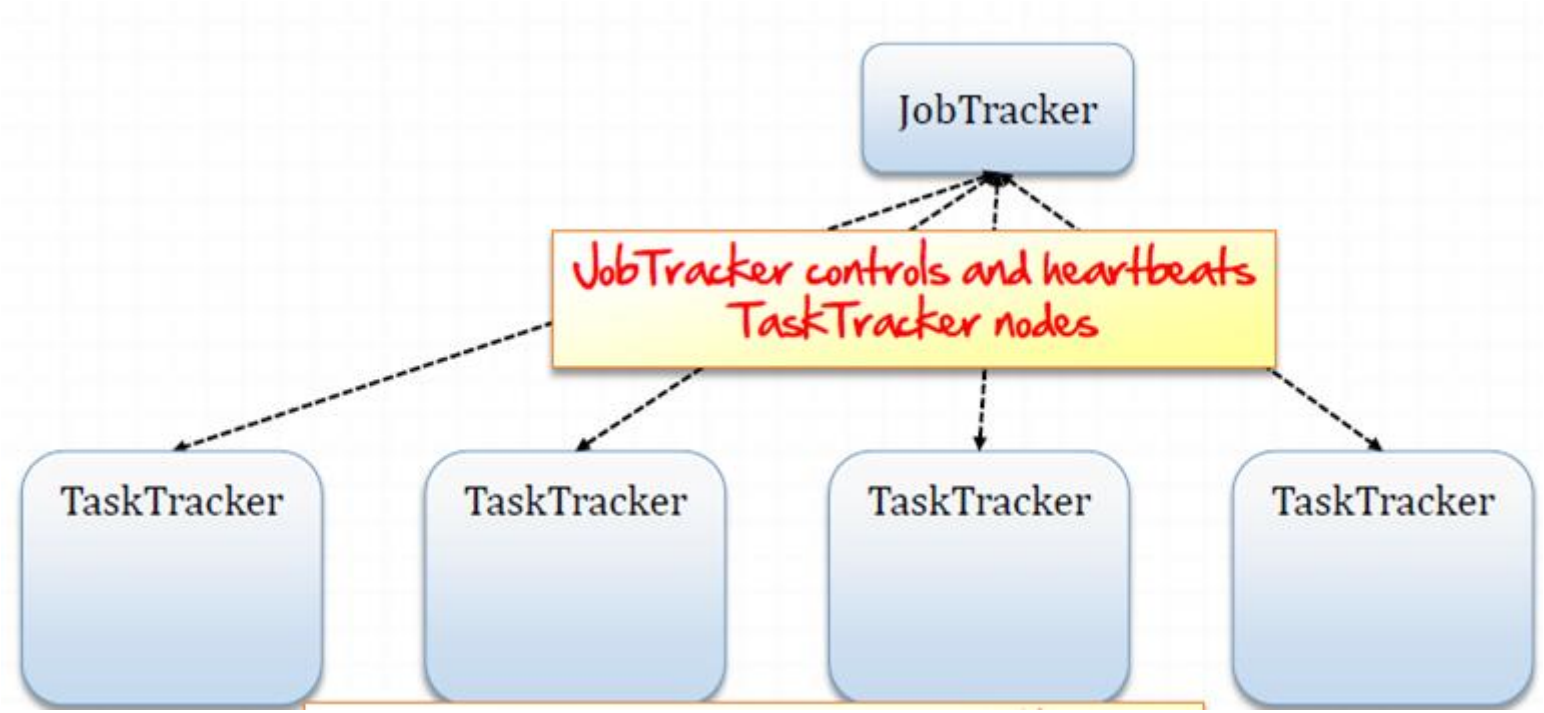


Design of MapReduce -Daemons

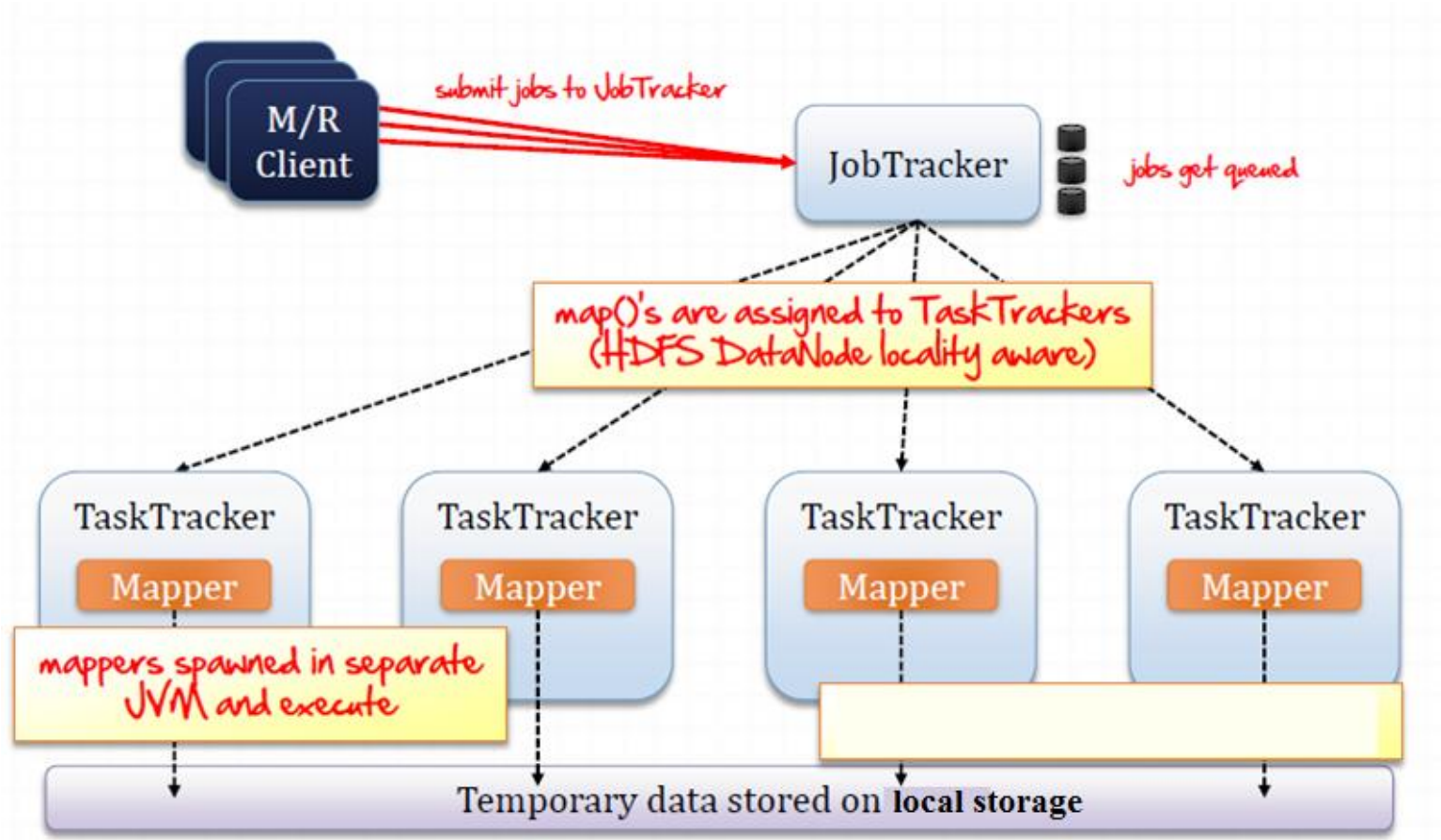
The MapReduce system is managed by two daemons

- JobTracker & TaskTracker
- JobTracker TaskTracker function in master / slave fashion
  - JobTracker coordinates the entire job execution
  - TaskTracker runs the individual tasks of map and reduce
  - JobTracker does the bookkeeping of all the tasks run on the cluster
  - **One map task is created for each input split**
  - Number of reduce tasks is configurable (mapred.reduce.tasks)

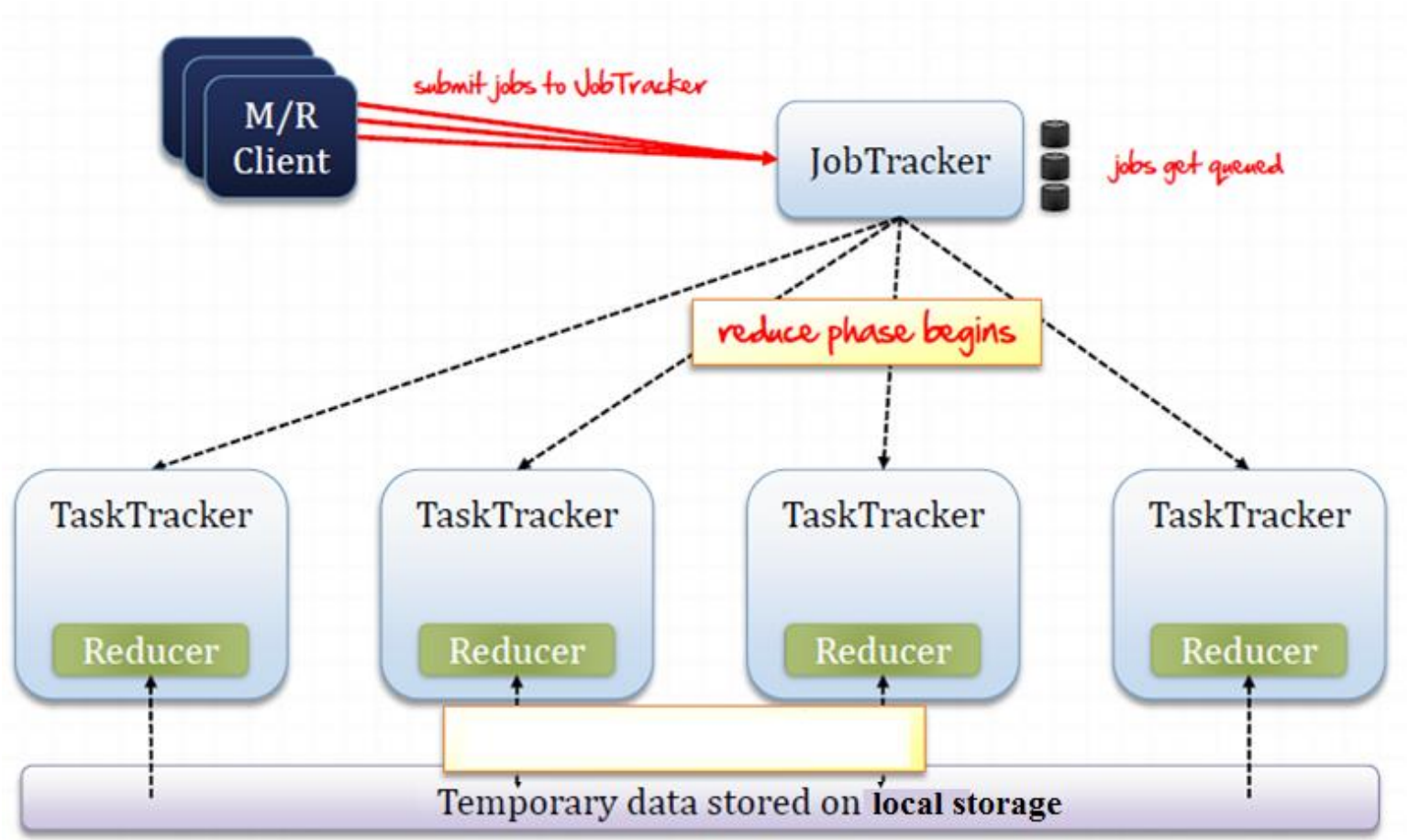
Conceptual Overview



Job Sumission – Map phase



Job Submission – Reduce Phase



mapred-site. xml - Pseudo Distributed Mode

```
<?xml version="1.0"?>
<!— mapred-site.xml —>
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:8021</value>
</property>
</configuration>
```

Note:Add "mapred.job.tracker" property under configuration tag to specify JobTracker location.

"localhost:8021" for Pseudo distributed mode.

Lastly set JAVA_HOME in conf/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64

Starting Hadoop -MapReduce Daemons

- Start MapReduce Services

  start-mapred.sh

  – Verify Daemons:jps

- To start HDFS + MR:

  – **start-all.sh**

Need for High-Level Languages

- Hadoop is great for large-data processing!
  - But writing Java programs for everything is verbose and slow
  - Not everyone wants to (or can) write Java code
- Solution: develop higher-level data processing languages
  - Hive: HQL is like SQL
  - Pig: Pig Latin is a bit like Perl

Hive and Pig

- Hive: data warehousing application in Hadoop
  - Query language is HQL, variant of SQL
  - Tables stored on HDFS as flat files
  - Developed by Facebook, now open source

- Pig: large-scale data processing system
  - Scripts are written in Pig Latin, a dataflow language
  - Developed by Yahoo!, now open source
  - Roughly 1/3 of all Yahoo! internal jobs
- Common idea:
  - Provide higher-level language to facilitate large-data processing
  - Higher-level language “compiles down” to Hadoop jobs

HIVE

- A datawarehousing framework built on top of Hadoop
- Started at Facebook in 2006
- Targets users are data analysts comfortable with SQL
- Allows to query the data using a SQL-like language called HiveQL
- Queries compiled into MR jobs that are executed on Hadoop
- Meant for structured data

Hive: Background

- Started at Facebook
- Data was collected by nightly cron jobs into Oracle DB
- “ETL” via hand-coded python
- Grew from 10s of GBs (2006) to 1 TB/day new data (2007), now 10x that

Hive Components

- Shell: allows interactive queries
- Driver: session handles, fetch, execute
- Compiler: parse, plan, optimize
- Execution engine: DAG of stages (MR, HDFS, metadata)
- Metastore: schema, location in HDFS, SerDe

Data Model

- Tables
  - Typed columns (int, float, string, boolean)
  - Also, list: map (for JSON-like data)
- Partitions
  - For example, range-partition tables by date
- Buckets
  - Hash partitions within ranges (useful for sampling, join optimization)

Metastore

- Database: namespace containing a set of tables
- Holds table definitions (column types, physical layout)
- Holds partitioning information
- Can be stored in Derby, MySQL, and many other relational databases

Physical Layout

- Warehouse directory in HDFS
  - E.g., /user/hive/warehouse
- Tables stored in subdirectories of warehouse
  - Partitions form subdirectories of tables
- Actual data stored in flat files
  - Control char-delimited text, or SequenceFiles
  - With custom SerDe, can use arbitrary format

Hive: Example

- Hive looks similar to an SQL database
- Relational join on two tables:
  - Table of word counts from Shakespeare collection
  - Table of word counts from the bible

```
SELECT s.word, s.freq, k.freq FROM shakespeare s
  JOIN bible k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
  ORDER BY s.freq DESC   LIMIT  10;
```

the	25848	62394
I	23031	8854
and	19671	38985
to	18038	13526
of	16700	34654
a	14170	8057
you	12702	2720
my	11297	4135
in	10797	12445
is	8882	6884

Hive: Behind the Scenes

```
SELECT s.word, s.freq, k.freq FROM shakespeare s
  JOIN bible k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
  ORDER BY s.freq DESC LIMIT 10;
```

(Abstract Syntax Tree)

```
(TOK_QUERY (TOK_FROM (TOK_JOIN (TOK_TABREF shakespeare s) (TOK_TABREF bible k) (= (. (TOK_TABLE_OR_COL s) word) (.
(TOK_TABLE_OR_COL k) word)))) (TOK_INSERT (TOK_DESTINATION (TOK_DIR TOK_TMP_FILE)) (TOK_SELECT (TOK_SELEXPR (.
(TOK_TABLE_OR_COL s) word)) (TOK_SELEXPR (. (TOK_TABLE_OR_COL s) freq)) (TOK_SELEXPR (. (TOK_TABLE_OR_COL k) freq))) (TOK_WHERE
(AND (>= (. (TOK_TABLE_OR_COL s) freq) 1) (>= (. (TOK_TABLE_OR_COL k) freq) 1))) (TOK_ORDERBY (TOK_TABSORTCOLNAMEDESC (.
(TOK_TABLE_OR_COL s) freq))) (TOK_LIMIT 10)))
```

(one or more of MapReduce jobs)

Hive: Behind the Scenes

```
STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-2 depends on stages: Stage-1
  Stage-0 is a root stage

STAGE PLANS:
  Stage: Stage-1
    Map Reduce
      Alias -> Map Operator Tree:
        s
          TableScan
            alias: s
            Filter Operator
              predicate:
                  expr: (freq >= 1)
                  type: boolean
              Reduce Output Operator
                key expressions:
                      expr: word
                      type: string
                sort order: +
                Map-reduce partition columns:
                      expr: word
                      type: string
                tag: 0
                value expressions:
                      expr: freq
                      type: int
                      expr: word
                      type: string
        k
          TableScan
            alias: k
            Filter Operator
              predicate:
                  expr: (freq >= 1)
                  type: boolean
              Reduce Output Operator
                key expressions:
                      expr: word
                      type: string
                sort order: +
                Map-reduce partition columns:
                      expr: word
                      type: string
                tag: 1
                value expressions:
                      expr: freq
                      type: int
      Reduce Operator Tree:
        Join Operator
          condition map:
               Inner Join 0 to 1
          condition expressions:
            0 {VALUE._col0} {VALUE._col1}
            1 {VALUE._col0}
          outputColumnNames: _col0, _col1, _col2
          Filter Operator
            predicate:
                expr: ((_col0 >= 1) and (_col2 >= 1))
                type: boolean
            Select Operator
              expressions:
                    expr: _col1
                    type: string
                    expr: _col0
                    type: int
                    expr: _col2
                    type: int
              outputColumnNames: _col0, _col1, _col2
              File Output Operator
                compressed: false
                GlobalTableId: 0
                table:
                    input format: org.apache.hadoop.mapred.SequenceFileInputFormat
                    output format: org.apache.hadoop.hive.ql.io.HiveSequenceFileOutputFormat

  Stage: Stage-2
    Map Reduce
      Alias -> Map Operator Tree:
        hdfs://localhost:8022/tmp/hive-training/364214370/10002
            Reduce Output Operator
              key expressions:
                    expr: _col1
                    type: int
              sort order: -
              tag: -1
              value expressions:
                    expr: _col0
                    type: string
                    expr: _col1
                    type: int
                    expr: _col2
                    type: int
      Reduce Operator Tree:
        Extract
          Limit
            File Output Operator
              compressed: false
              GlobalTableId: 0
              table:
                  input format: org.apache.hadoop.mapred.TextInputFormat
                  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat


  Stage: Stage-0
    Fetch Operator
      limit: 10
```

Example Data Analysis Task

Find users who tend to visit “good” pages.

Visits

user	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00

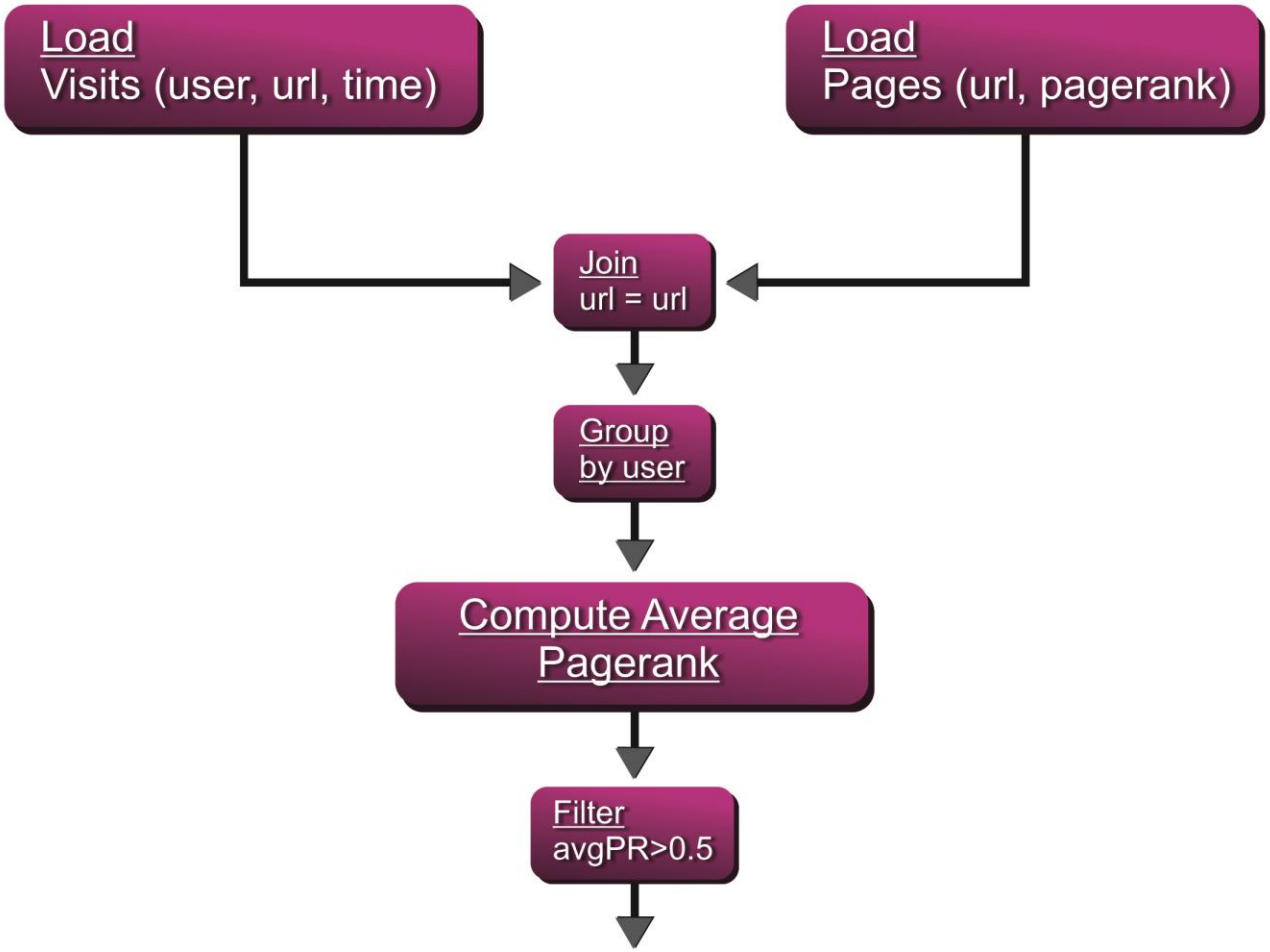
⋮

Pages

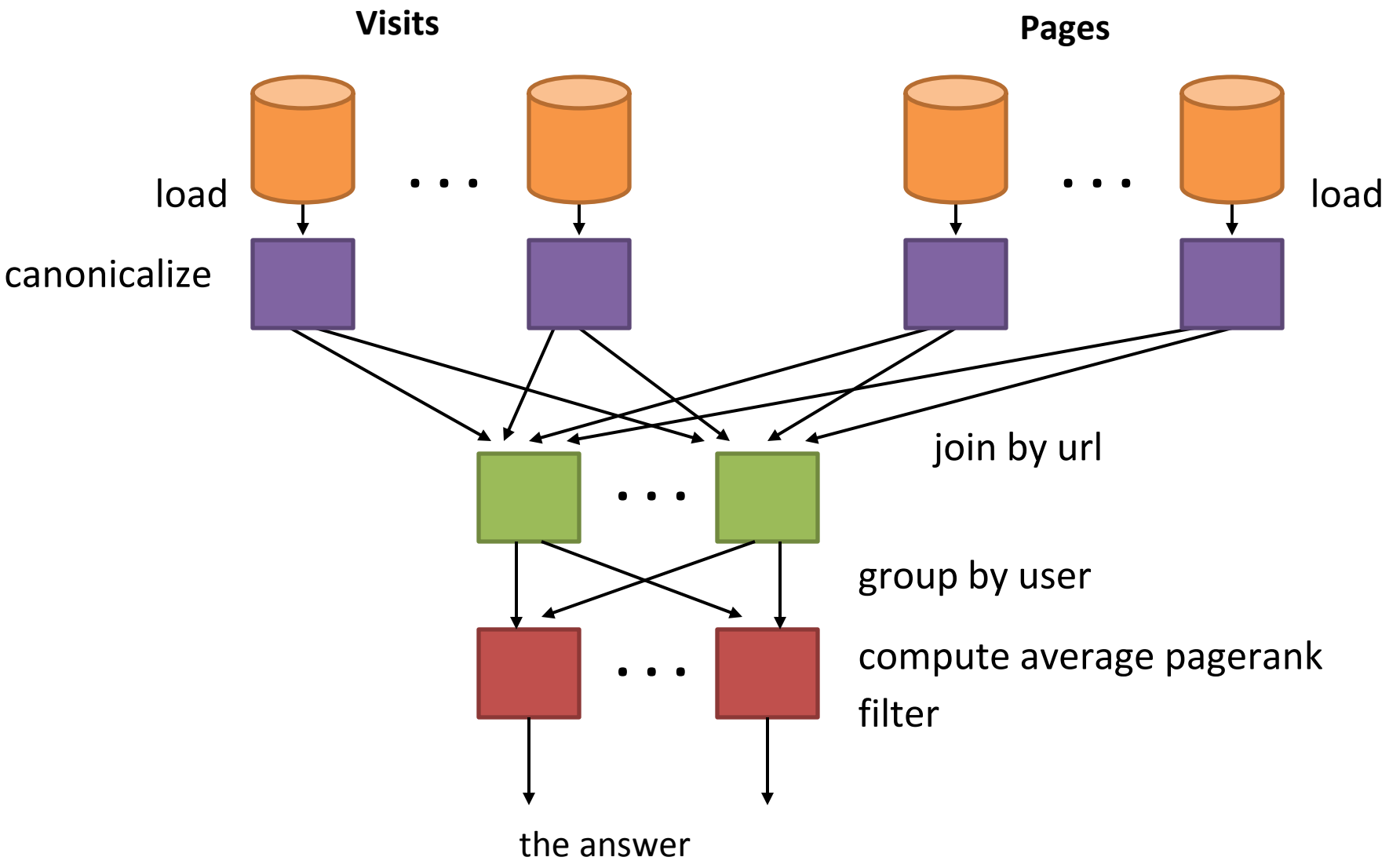
url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2

⋮

Conceptual Workflow

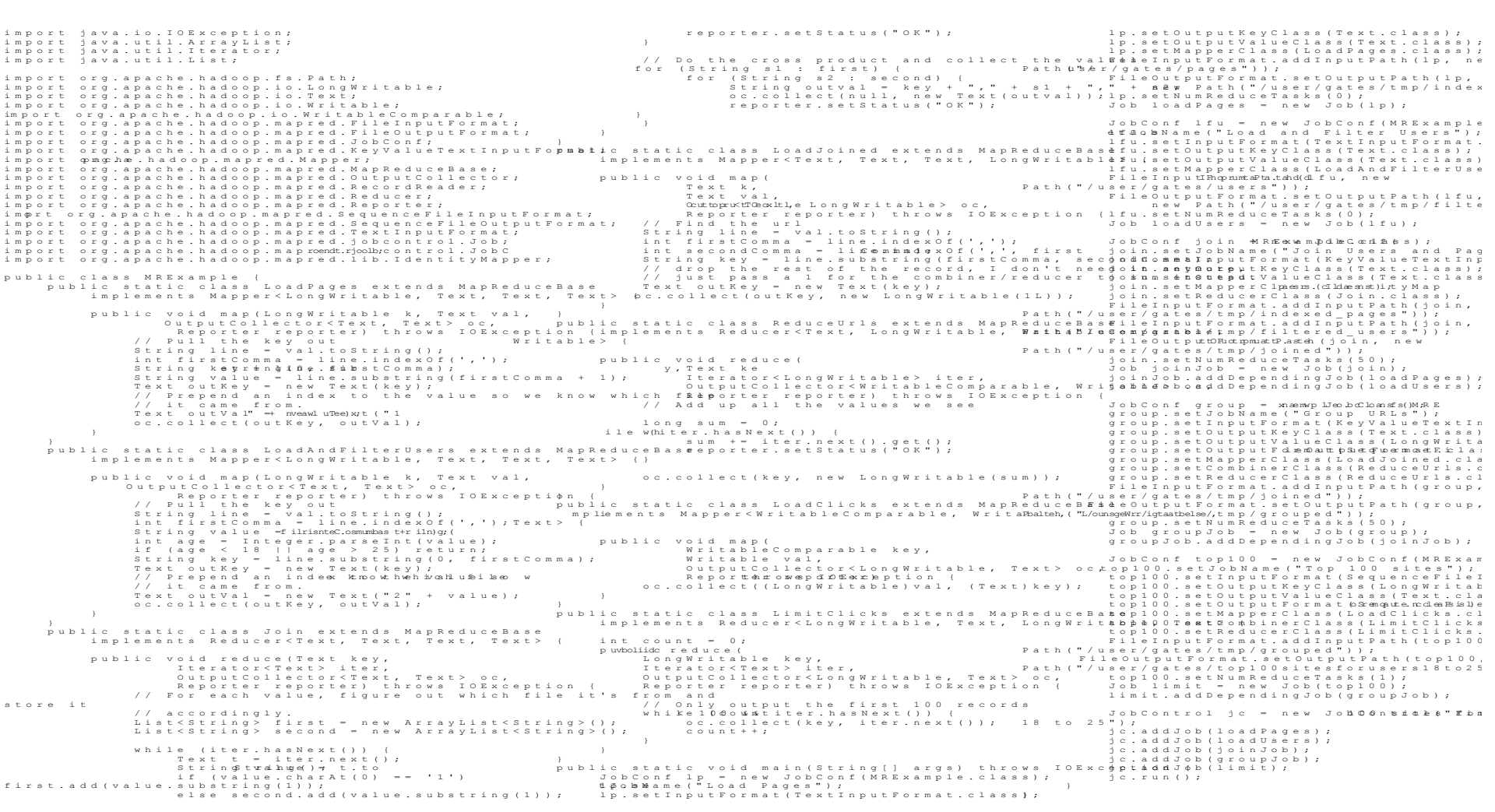


System-Level Dataflow



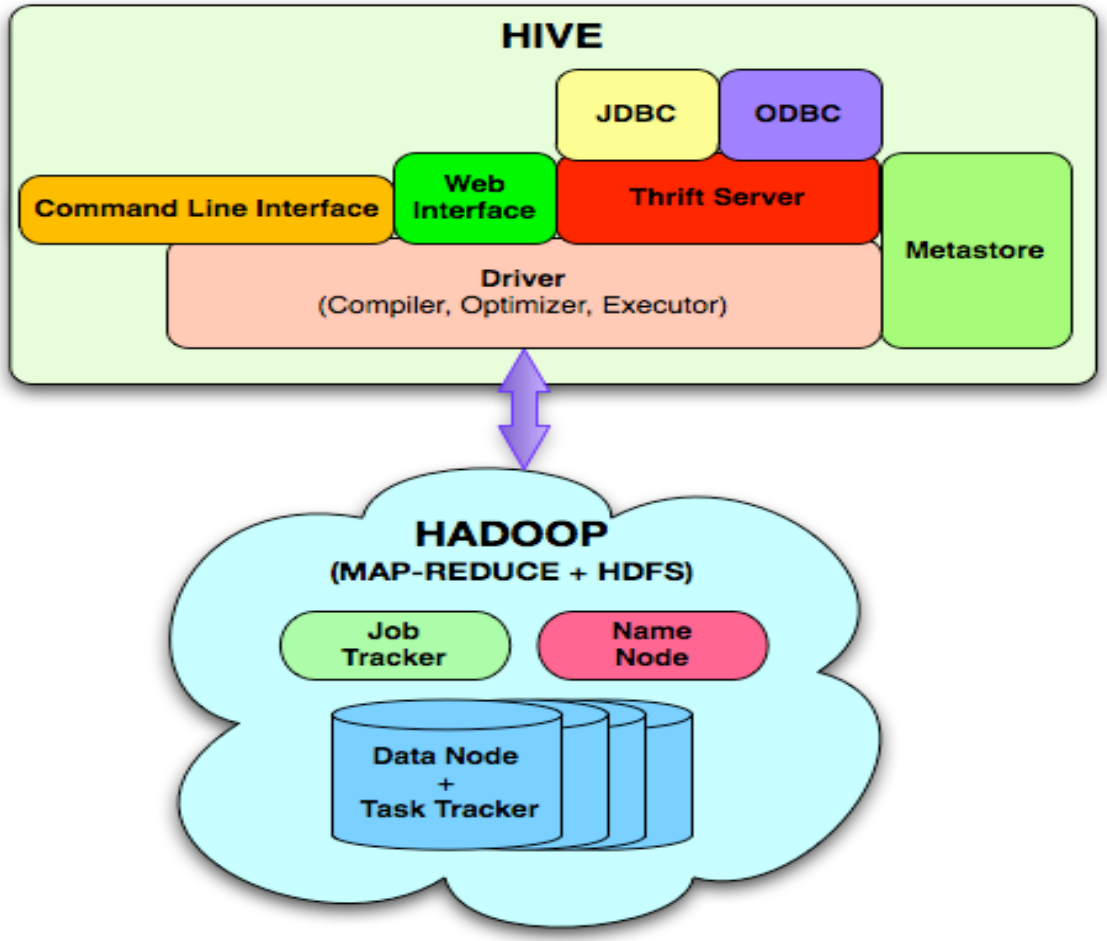
Pig Slides adapted from Olston et al.

MapReduce Code



Pig Slides adapted from Olston et al.

Hive Architecture



Hive Architecture cont.

- Can interact with Hive using :
  - CLI(Command Line Interface)
  - JDBC
  - Web GUI
- Metastore – Stores the system catalog and metadata about tables, columns, partitions etc.
- Driver – Manages the lifecycle of a HiveQL statement as it moves through Hive.
  Query Compiler – Compiles HiveQL into a directed acyclic graph of map/reduce tasks.
- Execution Engine – Executes the tasks produced by the compiler interacting with the underlying Hadoop instance.
- HiveServer – Provides a thrift interface and a JDBC/ODBC server.

Hive Architecture cont.

- **Physical Layout**

- Warehouse directory in HDFS
  - e.g., **/user/hive/warehouse**

- Table row data stored in subdirectories of warehouse

- Partitions form subdirectories of table directories

- Actual data stored in flat files
  - Control char-delimited text, or SequenceFiles

Hive Vs RDBMS

- Latency for Hive queries is generally very high (minutes) even when data sets involved are very small

- On RDBMSs, the analyses proceed much more iteratively with the response times between iterations being less than a few minutes.

- Hive aims to provide acceptable (but not optimal) latency for interactive data browsing, queries over small data sets or test queries.

- Hive is not designed for online transaction processing and does not offer real-time queries and row level updates.

It is best used for batch jobs over large sets of immutable data (like web logs).

Supported Data Types

- Integers
  - **BIGINT(8 bytes), INT(4 bytes), SMALLINT(2 bytes), TINYINT(1 byte).**
  - All integer types are signed.

- Floating point numbers
  - FLOAT(single precision), DOUBLE(double precision)

- STRING : sequence of characters
- BOOLEAN : True/False

- Hive also natively supports the following complex types:
  - Associative arrays – map<key-type, value-type>
  - Lists – list<element-type>
  - Structs – struct<field-name: field-type, ... >

Hive : Install & Configure

- Download a HIVE release compatible with your Hadoop installation from :
  - http://hive.apache.org/releases.html
- Untar into a directory. This is the HIVE's home directory
  - **tar xvzf hive.x.y.z.tar.gz**
- Configure
  - Environment variables – add in .bash_profile
    - **export HIVE_INSTALL=/<parent_dir_path>/hive-x.y.z**
    - **export PATH=$PATH:$HIVE_INSTALL/bin**
- Verify Installation
  - Type : ***hive –help*** (Displays commands usage)
  - Type : ***hive*** (Enter the hive shell)

    hive>

Hive : Install & Configure cont.

- Start Hadoop daemons (Hadoop needs to be running)
- Configure to Hadoop
- Create hive-site.xml in $HIVE_INSTALL/conf directory
- Specify the filesystem and jobtracker using the properties fs.default.name & mapred.job.tracker
- If not set, these default to the local filesystem and the local(in-process) job-runner
- Create following directories under HDFS
- /tmp (execute: **hadoop fs –mkdir /tmp**)
- user/hive/warehouse (execute: **hadoop fs –mkdir /user/hive/warehouse**)
- chmod g+w for both (execute : **hadoop fs –chmod g+w <dir_path>**)

Hive : Install & Configure cont.

- Data Store
  - Hive stores data under /user/hive/warehouse by default
- Metastore
  - Hive by default comes with a lightweight SQL database Derby to store the metastore metadata.
  - But this can be configured to other databases like MySQL as well.
- Logging
  - Hive uses Log4j
  - You can find Hive's error log on the local file system at */tmp/$USER/hive.log*

Hive Data Models

- **Databases**
- **Tables**
- **Partitions**
  - Each Table can have one or more partition Keys which determines how the data is stored.
  - Partitions - apart from being storage units - also allow the user to efficiently identify the rows that satisfy a certain criteria.
  - For example, a date_partition of type STRING and country_partition of type STRING.
  - Each unique value of the partition keys defines a partition of the Table. For example all “India" data from "2013-05-21" is a partition of the page_views table.
  - Therefore, if you run analysis on only the “India" data for 2013-05-21, you can run that query only on the relevant partition of the table thereby speeding up the analysis significantly.

Partition example

```
CREATE TABLE logs (ts BIGINT, line STRING)
PARTITIONED BY (dt STRING, country STRING);
```

- When we load data into a partitioned table, the partition values are specified explicitly:

```
LOAD DATA LOCAL INPATH 'input/hive/partitions/file1'
INTO TABLE logs
PARTITION (dt='2001-01-01', country='GB');
```

Hive Data Model cont.

- Buckets
  - Data in each table may in turn be divided into buckets based on the value of a hash function of some column of the Table.
  - For example the page_views table may be bucketed by userid, which is one of the columns, other than the partitions columns, of the page_views table. These can be used to efficiently sample the data.

A Practical session

Starting the Hive CLI

- Start a terminal and run :
  - $hive

- Will take you to the hive shell/prompt

  hive>

A Practical Session

Hive CLI Commands

- List tables:
  - **hive> show tables;**
- Describe a table:
  - **hive> describe <tablename>;**
- More information:
  - **hive> describe extended <tablename>;**

A Practical Session

Hive CLI Commands

- **Create tables:**
  – hive> **CREATE TABLE cite (citing INT, cited INT)**

>ROW FORMAT DELIMITED

>FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

- The 2nd and the 3rd lines tell Hive how the data is stored (as a text file)and how it should be parsed (fields are separated by commas).

- **Loading data into tables**
  - Let's load the patent data into table cite

  hive> **LOAD DATA LOCAL INPATH '<path_to_file>/cite75_99.txt'**
  > **OVERWRITE INTO TABLE cite;**

- **Browse data**
  hive> **SELECT * FROM cite LIMIT 10;**

A Practical Session

Hive CLI Commands

- **Count**
  hive>**SELECT COUNT(*) FROM cite;**

Some more playing around

- Create table to store citation frequency of each patent
  hive> **CREATE TABLE cite_count (cited INT, count INT);**

- Execute the query on the previous table and store the results :
  hive> **INSERT OVERWRITE TABLE cite_count**
- > **SELECT cited, COUNT(citing) FROM cite GROUP BY cited;**

- Query the count table
  hive> **SELECT * FROM cite_count WHERE count > 10 LIMIT 10;**

- Drop Table
  hive> **DROP TABLE cite_count;**

Managing Hive Tables

- **Managed table**
  - Default table created (without EXTERNAL keyword)
  - Hive manages the data

CREATE TABLE managed_table (dummy STRING);

LOAD DATA INPATH '/user/tom/data.txt' INTO table managed_table;

- Moves the data into the warehouse directory for the table.

DROP TABLE managed_table;

- Deletes table data & metadata.

Managing Hive Tables

- **External table**
  - You control the creation and deletion of the data.
  - The location of the external data is specified at table creation time:

    - **CREATE *EXTERNAL* TABLE external_table (dummy STRING)**
    - **LOCATION '/user/tom/external_table';**

      LOAD DATA INPATH '/user/tom/data.txt' INTO TABLE external_table;

    - **DROP TABLE external_table;**
      Hive will leave the data untouched and only delete the metadata.

Conclusion

- Supports rapid iteration of adhoc queries

- High-level Interface (HiveQL) to low-level infrastructure (Hadoop).

- Scales to handle much more data than many similar systems

Hive Resources/References

Documentation

- cwiki.apache.org/Hive/home.html

Mailing Lists

- hive-user@hadoop.apache.org

Books

- Hadoop, The Definitive Guide, 3rd edition by Tom White, (O'Reilly)

Hive Resources/References

Documentation

- cwiki.apache.org/Hive/home.html

Mailing Lists

- hive-user@hadoop.apache.org

Books

- Hadoop, The Definitive Guide, 3rd edition by Tom White, (O'Reilly)

PIG

PIG

- PIG is an abstraction layer on top of MapReduce that frees analysts from the complexity of MapReduce programming

- Architected towards handling unstructured and semi structured data

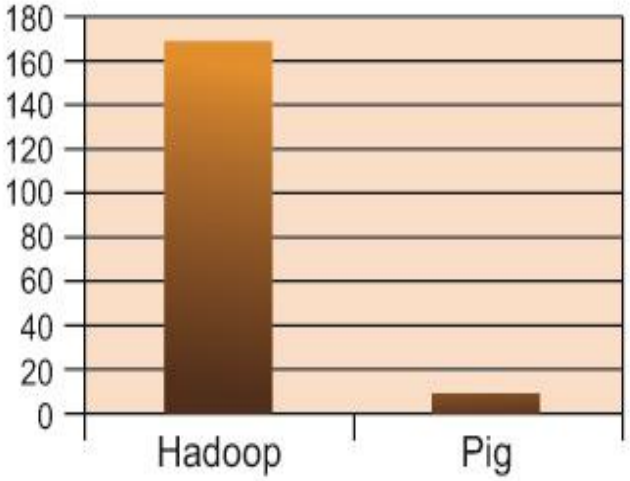
- It's a dataflow language, which means the data is processed in a sequence of steps transforming the data
- The transformations support relational-style operations such as filter, union, group, and join.
- Designed to be extensible and reusable
    - Programmers can develop own functions and use (UDFs)
- Programmer friendly
    - Allows to introspect data structures
    - Can do sample run on a representative subset of your input
- PIG internally converts each transformation into a MapReduce job and submits to hadoop cluster
- 40 percent of Yahoo's Hadoop jobs are run with PIG

Pig : What for ?

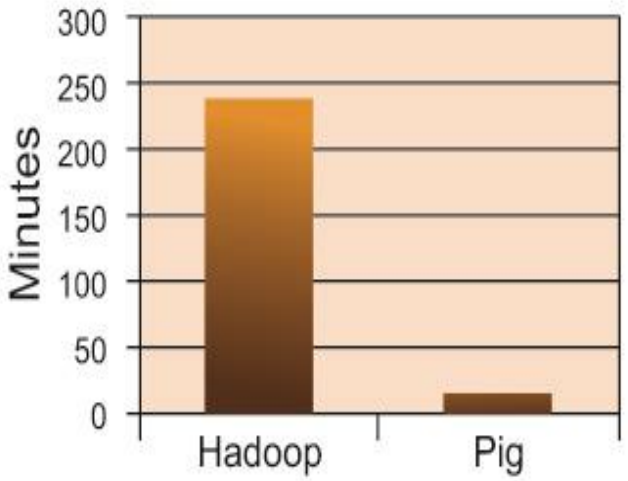
- An ad-hoc way of creating and executing map-reduce jobs on very large data sets
- Rapid development
- No Java is required
- Developed by Yahoo

PIG Vs MapReduce(MR)

1/20 the lines of code



1/16 the development time



Performance on par with raw Hadoop

Pig Use cases

- ✓ Processing of **Web Logs**.
- ✓ **Data processing** for search platforms.
- ✓ Support for **Ad Hoc queries** across large datasets.
- ✓ **Quick Prototyping** of algorithms for processing large datasets.

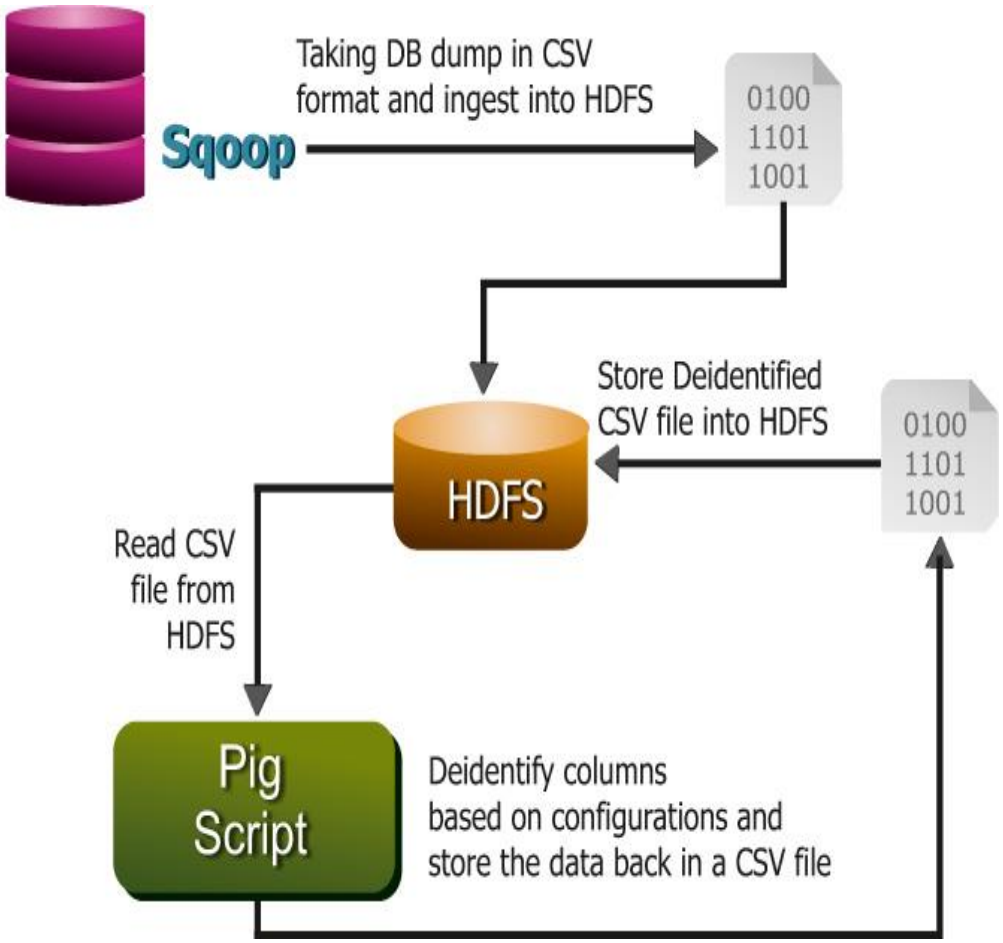
Use Case in Healthcare

Problem Statement:
De-identify personal health information.

Challenges:
- ✓ Huge amount of data flows into the systems daily and there are multiple data sources that we need to aggregate data from.

- ✓ Crunching this huge data and deidentifying it in a traditional way had problems.

Use Case in Healthcare



When to Not Use PIG

- Really nasty data formats or **completely unstructured data** (video, audio, raw human-readable text).
- Pig is definitely **slow** compared to Map Reduce jobs.
- When you would like **more power** to optimize your code.

PIG Architecture

- Pig runs as a client side application ,there is no need to install anything on the cluster.

Install and Configure PIG

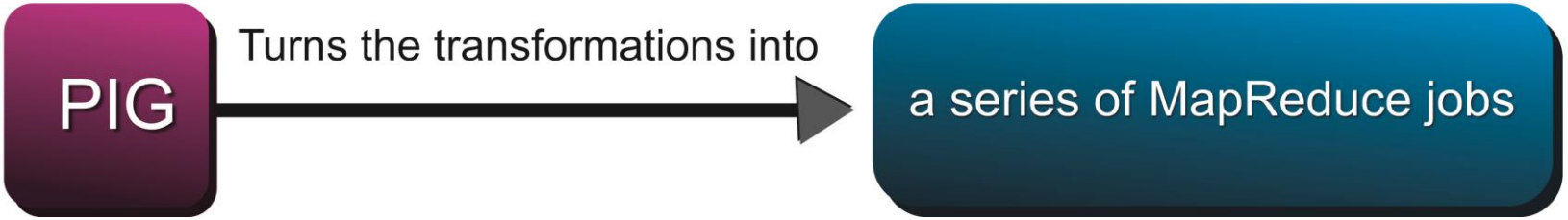
- Download a version of PIG compatible with your hadoop installation
    - http://pig.apache.org/releases.html
- Untar into a designated folder. This will be Pig's home directory
    - tar xzf pig-x.y.z.tar.gz
- Configure
    - Environment Variables add in .bash_profile
        - export PIG_INSTALL=/<parent directory path>/pig-x.y.z
        - export PATH=$PATH:$PIG_INSTALL/bin
- Verify Installation
    - Try pig -help
        - Displays command usage
    - Try **pig**
        - Takes you into Grunt shell grunt>

PIG Execution Modes

- Local Mode
    - Runs in a single JVM
    - Operates on local file system
    - Suitable for small datasets and for development
    - To run PIG in local mode
        - pig -x local
- MapReduce Mode
    - In this mode the queries are translated into MapReduce jobs and run on hadoop cluster
    - PIG version must be compatible with hadoop version
    - Set HADOOP HOME environment variable to indicate pig which hadoop client to use
        - export HADOOP_HOME=$HADOOP_INSTALL
        - If not set it will use a bundled version of hadoop

Pig Latin

It is made up of a series of operations or transformations that are applied to the input data to produce output.



Example Data Analysis Task

Find users who tend to visit “good” pages.

Visits

user	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00

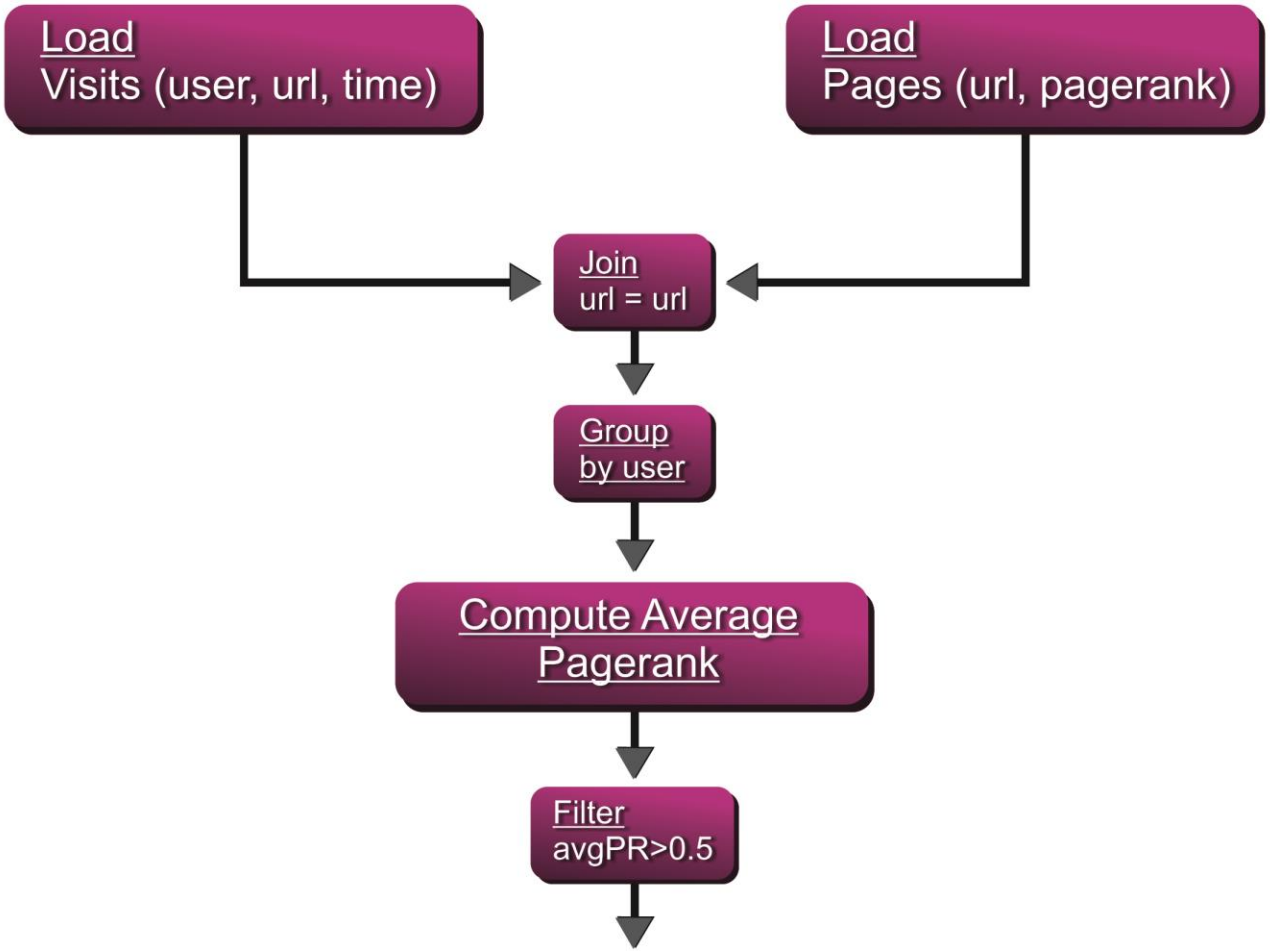
⋮

Pages

url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2

⋮

Conceptual Workflow



Pig Latin Relational Operators

Category	Operator	Description
Loading and Storing	LOAD STORE DUMP	Loads data from the file system or other storage into a relation . Saves a relation to the file system or other storage. Prints a relation to the console.
Filtering	FILTER DISTINCT FOREACH…GENERATE STREAM	Removes unwanted rows from a relation. Removes duplicate rows from a relation. Adds or removes fields from a relation. Transforms a relation using an external program.
Grouping and Joining	JOIN COGROUP GROUP CROSS	Joins two or more relations. Groups the data in two or more relations. Groups the data in a single relation. Creates the cross product of two or more relations.
Sorting	ORDER LIMIT	Sorts a relation by one or more fields. Limits the size of a relation to a maximum number of tuples.
Combining and Splitting	UNION SPLIT	Combines two or more relations into one. Splits a relation into two or more relations.

Ways of Executing PIG programs

Grunt

- An interactive shell for running Pig commands
- Grunt is started when the pig command is run without any options

- Script
  - Pig commands can be executed directly from a script file

    pig pigscript.pig
  - It is also possible to run Pig scripts from Grunt shell using run and exec.
- Embedded
  - You can run Pig programs from Java using the PigServer class, much like you can use JDBC
  - For programmatic access to Grunt, use PigRunner

An Example

Create a file sample.txt with(tab delimited) :

1932 23 2
1905 12 1

And so on.

grunt> records = LOAD ‘<your_input_dir>/sample.txt'
AS (year:chararray, temperature:int, quality:int);

DUMP records;

grunt>filtered_records = FILTER records BY temperature >=22;

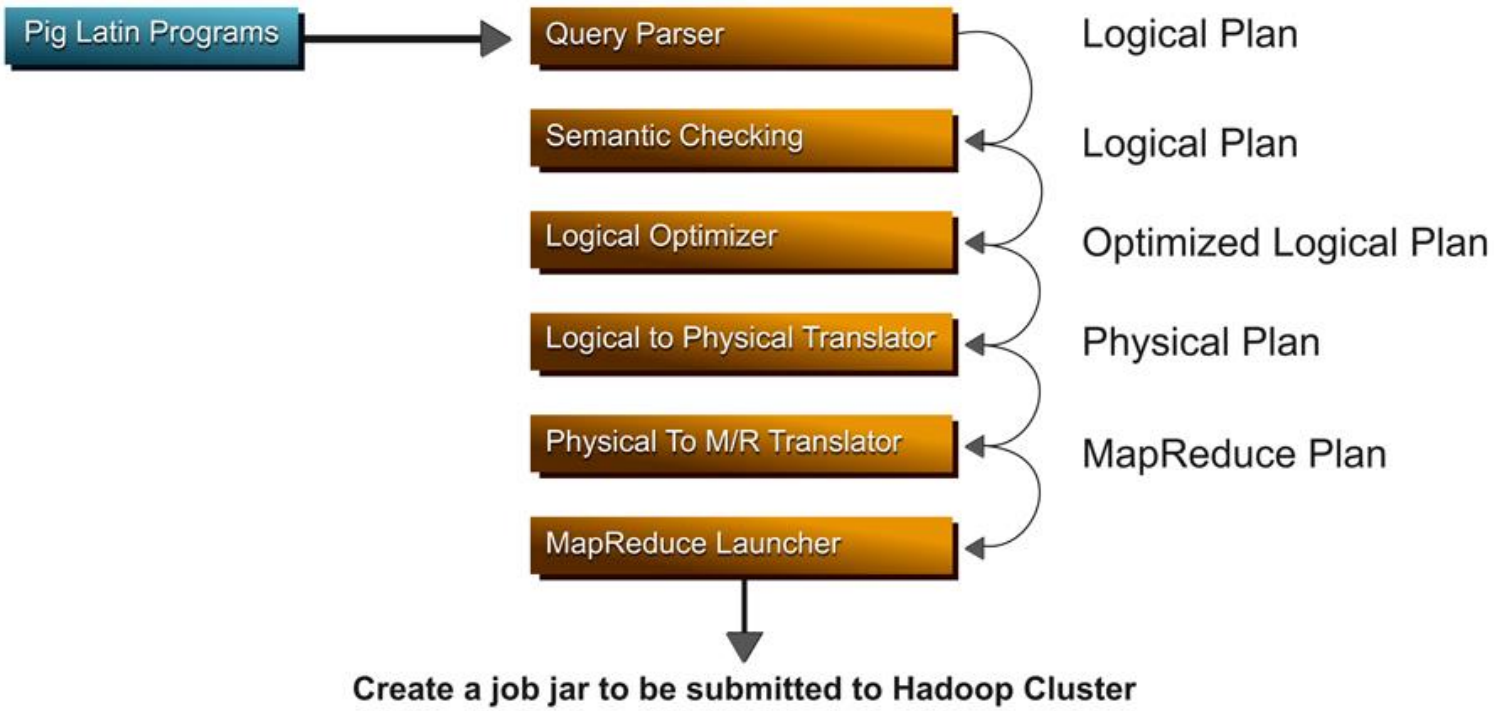
Example cont.

```
grunt>grouped_records = GROUP filtered_records
BY year;


grunt>
max_temp  = FOREACH  grouped_records
GENERATE group,
MAX (filtered_records.temperature);


grunt>DUMP max_temp;
```

Compilation



Data Types

- Simple Type

Category	Type	Description
Numeric	int	32 - bit signed integer
	long	64 - bit signed integer
	float	32-bit floating-point number
	double	64-bit floating-point number
Text	chararray	Character array in UTF-16 format
Binary	bytearray	Byte array

Data Types

- Complex Types

Type	Description	Example
Tuple	Sequence of fields of any type	(1 ,'pomegranate')
Bag	An unordered collection of tuples, possibly with duplicates	{(1 ,'pomegranate') ,(2)}
map	A set of key-value pairs; keys must be character arrays but values may be any type	['a'#'pomegranate']

LOAD Operator

<relation name> = LOAD '<input file with path>' [USING UDF()]

[AS (<field name>:dataType, <field name2>:dataType<field name3>:dataType)]

- Loads data from a file into a relation
- Uses the PigStorage load function as default unless specified otherwise with the USING option
- The data can be given a schema using the AS option.
- The default data type is bytearray if not specified

records=LOAD 'sales.txt';

records=LOAD 'sales.txt' AS (fl:chararray, f2:int, f3:f1oat.);

records=LOAD isales.txt' USING PigStorage('\t');

records= LOAD (sales.txt' USING PigStorage('\t') AS (fl:chararray, f2:int,f3:float);

Diagnostic Operators

- DESCRIBE
    - Describes the schema of a relation
- EXPLAIN
    - Display the execution plan used to compute a relation
- ILLUSTRATE
    - Illustrate step-by-step how data is transformed
    - Uses sample of the input data to simulate the execution.

Data Write Operators

- LIMIT
    - Limits the number of tuples from a relation
- DUMP
    - Display the tuples from a relation
- STORE
    - Store the data from a relation into an HDFS directory.
    - The directory must not exist

Relational Operators

- FILTER
- Selects tuples based on Boolean expression

  teenagers = FILTER cust BY age <20;
- ORDER
  - Sort a relation based on one or more fields
  - Further processing (FILTER, DISTINCT, etc.) may destroy the ordering

    ordered list = ORDER cust BY name DESC;
- DISTINCT
  - Removes duplicate tuples

    unique_custlist = DISTINCT cust;

Relational Operators

- GROUP BY
    - Within a relation, group tuples with the same group key
    - GROUP ALL will group all tuples into one group

        groupByProfession=GROUP cust BY profession

        groupEverything=GROUP cust ALL
- FOR EACH
    - Loop through each tuple in nested_alias and generate new tuple(s)

    . countByProfession=FOREACH groupByProfession GENERATE group, count(cust);
    - Built in aggregate functions AVG, COUNT, MAX, MIN, SUM

Relational Operators

- GROUP BY
    - Within a relation, group tuples with the same group key
    - GROUP ALL will group all tuples into one group

      groupByProfession=GROUP cust BY profession

      groupEverything=GROUP cust ALL
- FOR EACH
    - Loop through each tuple in nested alias and generate new tuple(s).
    - At least one of the fields of nested alias should be a bag
    - DISTINCT, FILTER, LIMIT, ORDER, and SAMPLE are allowed operations in nested op to operate on the inner bag(s).
    - countByProfession=FOREACH groupByProfession GENERATE group, count(cust);
    - Built in aggregate functions AVG, COUNT, MAX, MIN, SUM

Operating on Multiple datasets

- **Join**

Compute inner join of two or more relations based on common fields values

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)

DUMP B;
(2,4)
(8,9)
(1,3)
(2,7)
(7,9)

X=JOIN A BY a1,B BY b1
DUMP X;
(1,2,3,1,3)
(8,3,4,8,9)
(7,2,5,7,9)

Operating on Multiple datasets

- **COGROUP**

  Group tuples from two or more relations,based on common group values

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)

DUMP B;
(2,4)
(8,9)
(1,3)
(2,7)
(7,9)

X=COGROUP A BY a1,B BY b1
DUMP X;
(1,{(1,2,3)},{(1,3)})
(8,{(8,3,4)},{(8,9)})
(7,{(7,2,5)},{(7,9)})
(2,{},{(2,4),(2,7)})
(4,{(4,2,1),(4,3,3)},{})

Joins & Cogroups

- **JOIN** and **COGROUP** operators perform similar functions**.**
- **JOIN** creates a flat set of output records while **COGROUP** creates a nested set of output records

Data

File – student

Name	Age	GPA
Joe	18	2.5
Sam		3.0
Angel	21	7.9
John	17	9.0
Joe	19	2.9

File – studentRoll

Name	RollNo
Joe	45
Sam	24
Angel	1
John	12
Joe	19

Pig Latin - GROUP Operator

Example of GROUP Operator:

```
A = load 'student' as (name:chararray, age:int, gpa:float);
dump A;
(joe,18,2.5)
(sam,,3.0)
(angel,21,7.9)
(john,17,9.0)
(joe,19,2.9)

X = group A by name;
dump X;
(joe,{(joe,18,2.5),(joe,19,2.9)})
(sam,{(sam,,3.0)})
(john,{(john,17,9.0)})
(angel,{(angel,21,7.9)})
```

Pig Latin – COGROUP Operator

Example of COGROUP Operator:

```
A = load 'student' as (name:chararray, age:int,gpa:float);
B = load 'studentRoll' as (name:chararray, rollno:int);

X = cogroup A by name, B by name;
dump X;
```

```
(joe,{(joe,18,2.5),(joe,19,2.9)},{(joe,45),(joe,19)})
(sam,{(sam,,3.0)},{(sam,24)})
(john,{(john,17,9.0)},{(john,12)})
(angel,{(angel,21,7.9)},{(angel,1)})
```

Operating on Multiple datasets

UNION
Creates the union of two or more relations

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)

DUMP B;
(2,4)
(8,9)

X=UNION A , B;
DUMP X;

(1,2,3)
(4,2,1)
(8,3,4)
(2,4)
(8,9)

Operating on Multiple datasets

SPLITS

Splits a relation into two or more relations, based on a Boolean expression

Y=SPLIT X INTO C IF a1<5 , D IF a1>5;

DUMP C;	DUMP D;
(1,2,3)	
(4,2,1)	(8,3,4)
(2,4)	(8,9)

User Defined Functions (UDFs)

- PIG lets users define their own functions and lets them be used in the statements
- The UDFs can be developed in Java, Python or Javascript
  - Filter UDF
    - To be subclassof FilterFunc which is a subclass ofEvalFunc
  - Eval UDF

    To be subclassed of EvalFunc

```
public abstract class EvalFunc<T> {
public abstract T exec(Tuple input) throws 1OException;
}
```

  - Load / Store UDF
    - To be subclassed of LoadFunc /StoreFunc

Creating UDF : Eval function example

```
public class UPPER extends EvalFunc<String> {
      @Override
       public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0)
        return null;
        try {
               String str = (String) input.get(0);
               return str.toUpperCase(); // java upper case
function
        } catch (Exception e) {
        throw new IOException("Caught exception processing
input...", e);
        }
       }
}
```

Define and use an UDF

- Package the UDF class into a jar
- Define and use a UDF
    - REGISTER yourUDFjar_name.jar;
- cust = LOAD some_cust_data;
    - filtered = FOREACH cust GENERATE com.pkg.UPPER(title) ;
- DUMP filtered;

Piggy Bank

- Piggy Bank is a place for Pig users to share the Java UDFs they have written for use with Pig. The functions are contributed "as-is.“
- Piggy Bank currently supports Java UDFs.
- No binary version to download, download source, build and use.

Pig Vs. Hive

- Hive was invented at Facebook.
- Pig was invented at Yahoo.

- If you know SQL, then Hive will be very familiar to you. Since Hive uses SQL, you will feel at home with all the familiar **select**, **where**, **group by**, and **order by** clauses similar to SQL for relational databases.

Pig Vs. Hive

- Pig needs some mental adjustment for SQL users to learn.
- Pig Latin has many of the usual data processing concepts that SQL has, such as filtering, selecting, grouping, and ordering, but the syntax is a little different from SQL (particularly the **group by** and **flatten** statements!).

Pig Vs. Hive

- Pig gives you more control and optimization over the flow of the data than Hive does.
- If you are a data engineer, then you'll likely feel like you'll have better control over the dataflow (ETL) processes when you use Pig Latin, if you come from a procedural language background.
- If you are a data analyst, however, you will likely find that you can work on Hadoop faster by using Hive, if your previous experience was more with SQL.

Pig Vs. Hive

- Pig Latin allows users to store data at any point in the pipeline without disrupting the pipeline execution.
- Pig is not meant to be an ad-hoc query tool.

Pig Vs. Hive

- Pig needs some mental adjustment for SQL users to learn.
- Pig Latin has many of the usual data processing concepts that SQL has, such as filtering, selecting, grouping, and ordering, but the syntax is a little different from SQL (particularly the **group by** and **flatten** statements!).

Further Reading/References

- Hadoop in Action
- Hadoop, The Definitive Guide
- All respective apache sites

Thank You!

- Email : abhishekroy8@gmail.com
- LinkedIn: **http://www.linkedin.com/in/abhishekroy8**
- Skype: **abhishek.roy08**
- Twitter: **@abhishekroy88**