BIG DATA

Hadoop Ecosystem

K V Subramaniam
Computer Science and Engineering

Hadoop Ecosystem Overview

PES UNIVERSITY ONLINE

What we have learnt so far..

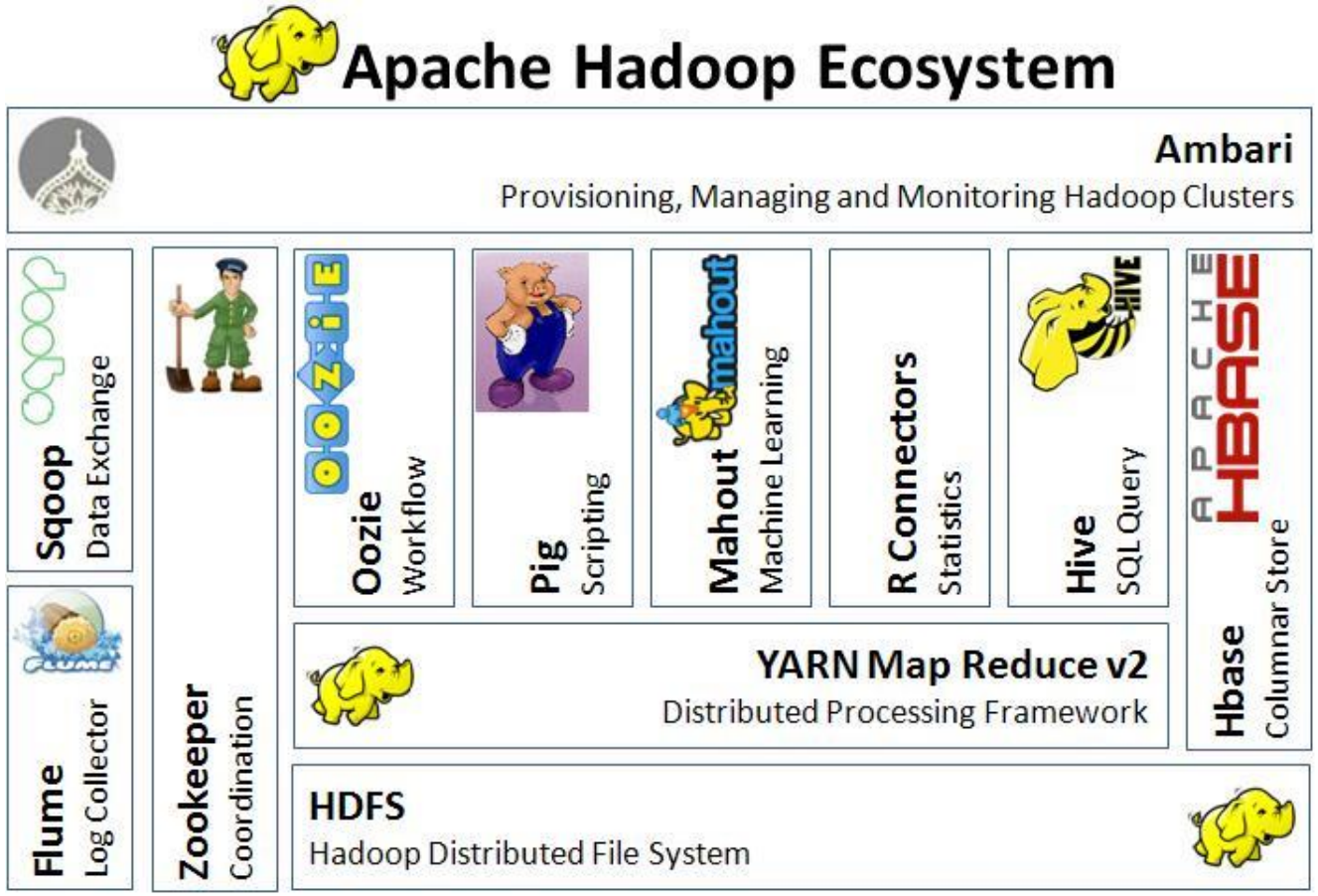
HDFS – for storage

And MapReduce (Hadoop) for computation

So where does it all fit in the bigger scheme of a Big Data architecture

BIG DATA

Hadoop Ecosystem



www.facebook.com/hadoopers

Hadoop Workflows: Oozie

Oozie: Motivational exercise

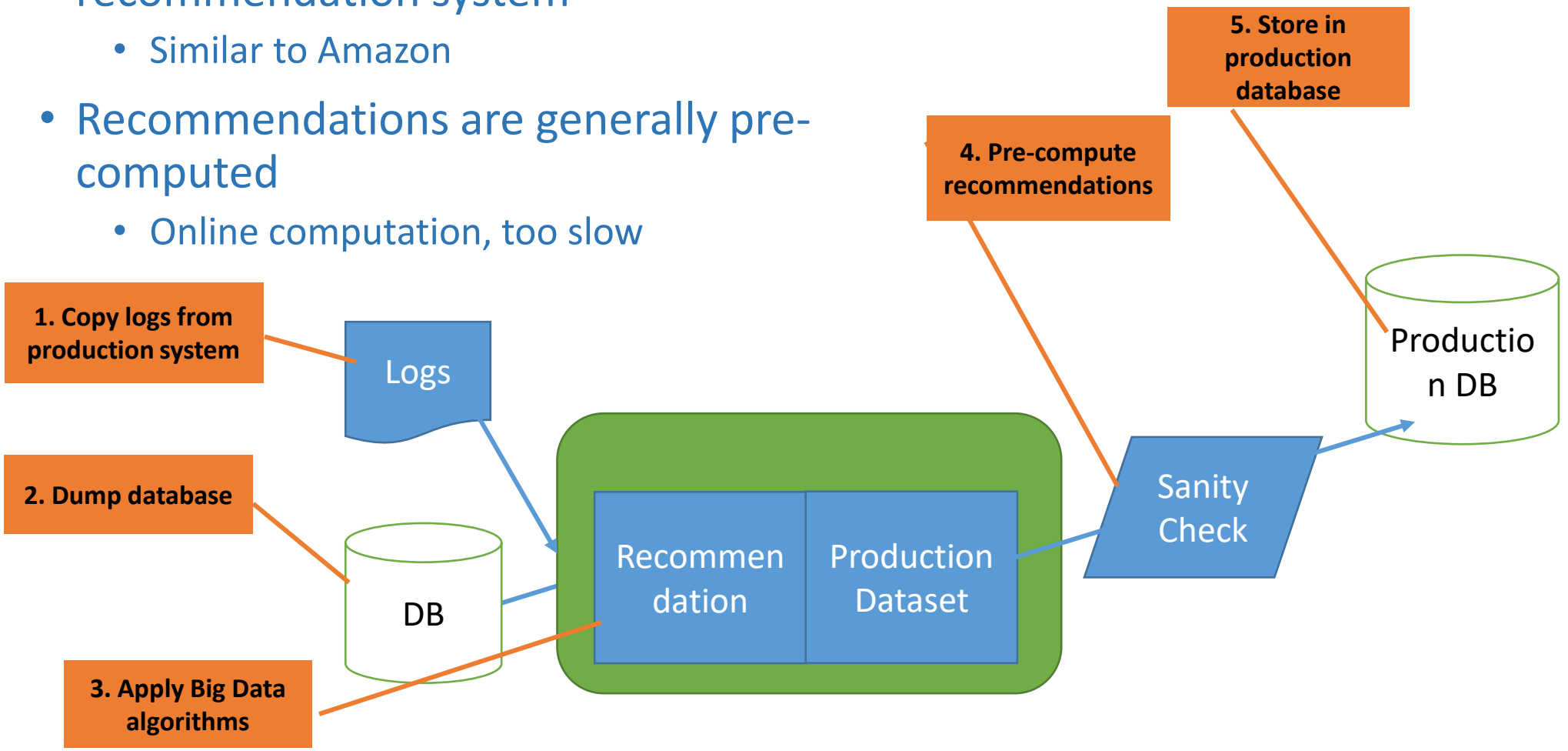
- Suppose we want to build a recommendation system like in AMAZON

  - What inputs would we need?
  - How often would we need to update the recommendations (every hour? Every day? EVERY week?)
  - Is it enough to just build a recommendation algorithm?
    - Let's assume that we have a MAGIC "RECOMMENDATION ALGORITHM" that can work PROVIDED the right inputs are given to it.

BIG DATA

Hadoop Ecosystem

- We want to use the sales to build a recommendation system
  - Similar to Amazon
- Recommendations are generally pre-computed
  - Online computation, too slow



BIG DATA

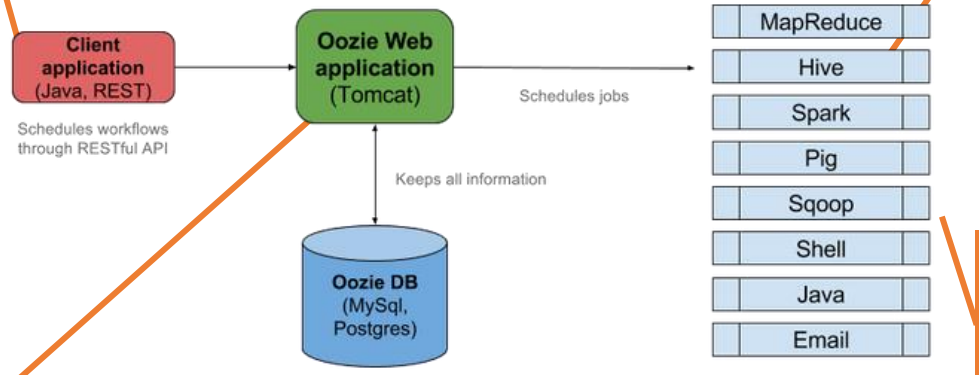
Workflow Definition

- The sequence of steps is called a *Workflow*
- Workflows are common in data centers
- Users would like to
  - Specify the steps
  - Specify when the steps are to be run
    - Maybe periodically
  - Run the Workflow
  - What to do in case of error

BIG DATA

Oozie Architecture

1. User submits Workflow (XML)



3. Some of the applications supported by Oozie

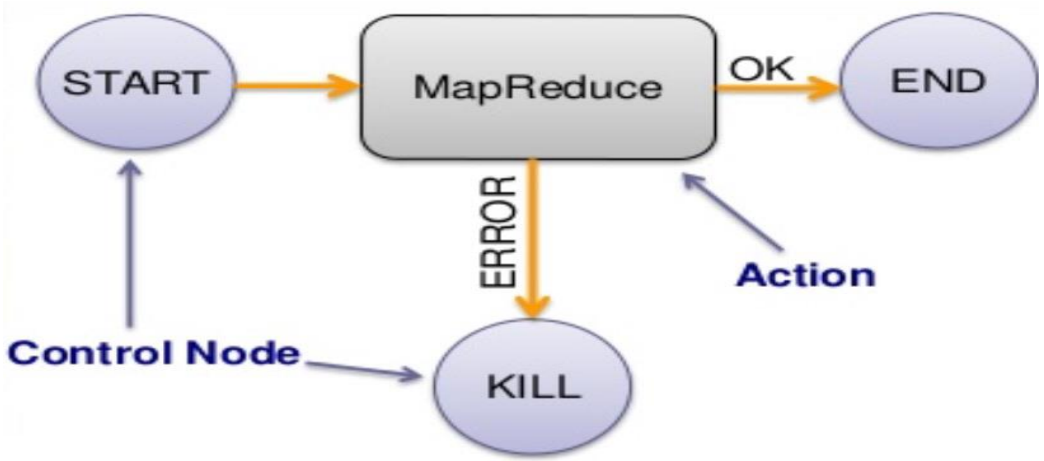
2. Oozie: Tomcat application

4. Workflow can include any of these applications

Image courtesy: https://oyermolenko.blog/2017/10/01/scheduling-jobs-in-hadoop-through-oozie/

Oozie Workflow (pictorial)

- **Action nodes**
  - Does something, e.g., run Mapreduce
  - Every action node has a normal exit and an error exit
- **Control nodes**
  - Start is the beginning of a Workflow
  - End and kill are the end
  - Other control nodes
    - Fork (start parallel tasks) and Join (merge parallel tasks)
    - Decision: like switch



DAG Expressing Workflow

BIG DATA

Oozie References

- T1 – Chapter 2.6.1.2
- T2 – Chapter 7.5

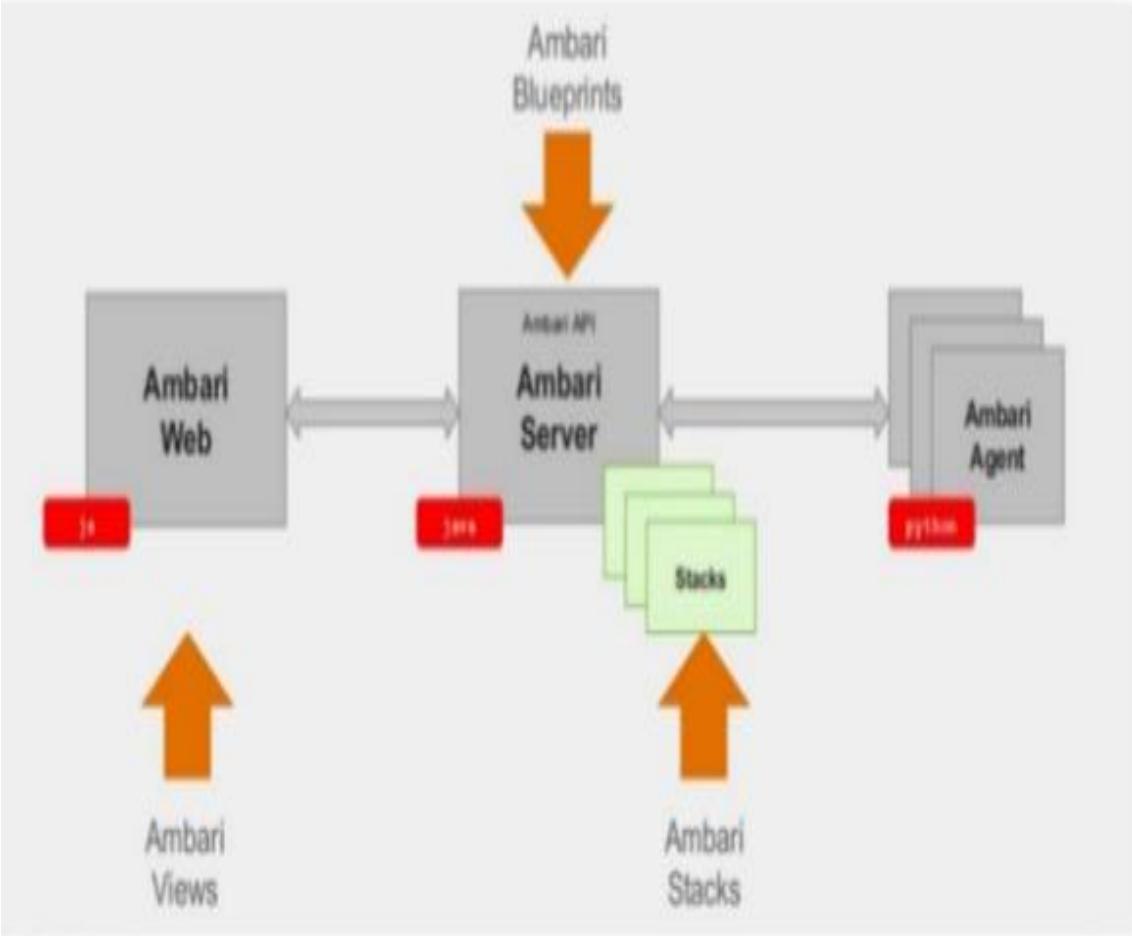
- http://oozie.apache.org/docs/4.3.0/index.html
  - Official Oozie Homepage
- https://oyermolenko.blog/2017/10/01/scheduling-jobs-in-hadoop-through-oozie/
  - A very good introduction to Oozie

Hadoop Workflows: Ambari

Ambari : Deploy and Manage Hadoop Clusters

- Simplifies Installation, Configuration and Management
- Easy, efficient, repeatable creation of clusters
- Manages and Monitors clustering

Ambari Architecture



- **Ambari Stacks**
  Describes the applications to be installed,
  eg, Hadoop, its components and its structure
- **Ambari Blueprints**
  Creation of the cluster
- **Ambari Views**
  User interface

Source: https://www.slideshare.net/hortonworks/managing-enterprise-hadoop-clusters-with-apache-ambari

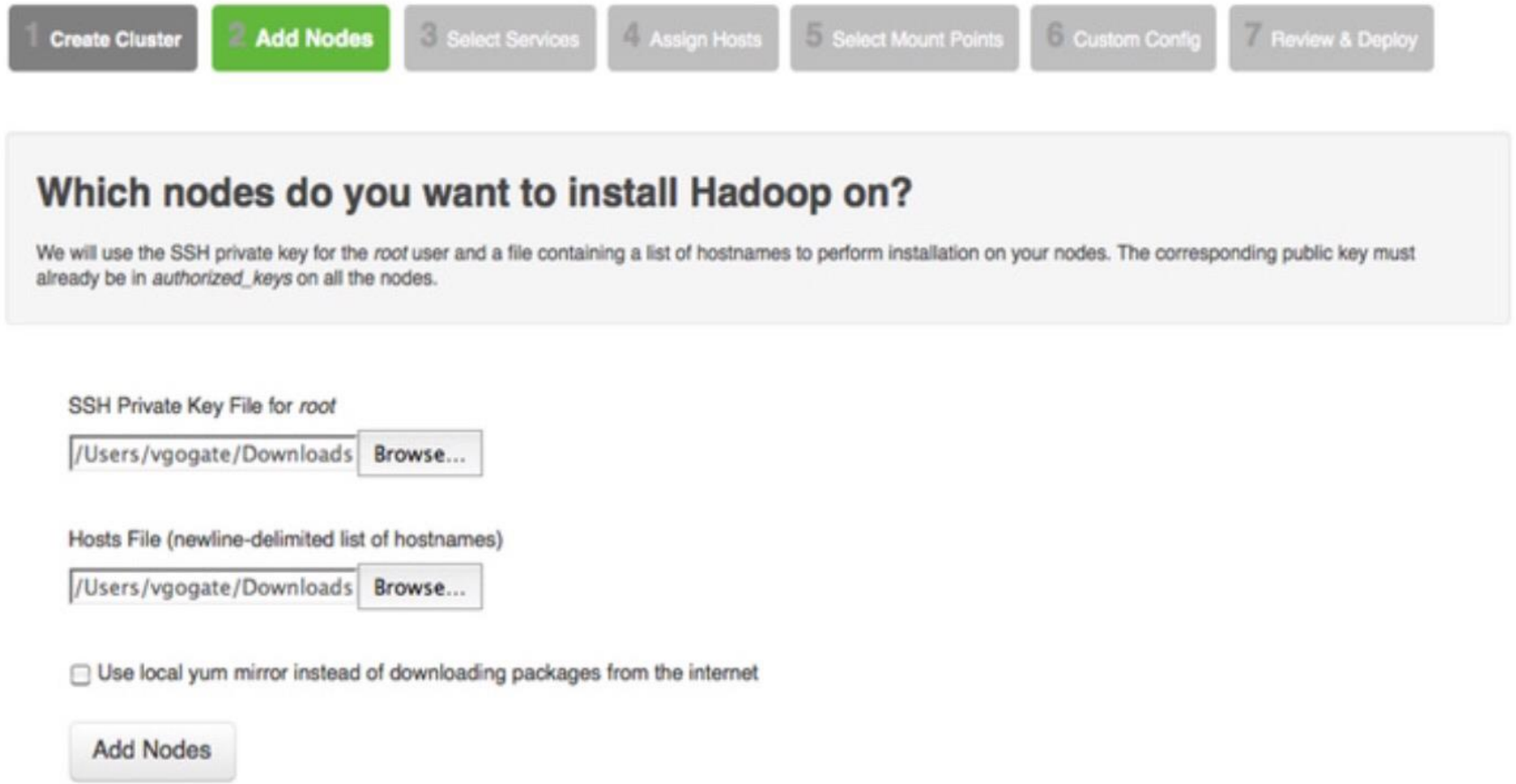
Ambari Stacks

What do we need to define in Ambari to install a cluster ?

Term	Meaning
Stack	Set of services, where to get the software packages, e.g. HDP (Hortonworks Data Platform)
Service	Components that make up the service e.g, HDFS
Component	Building blocks of the service – Namenode, Datanode
Category	Master, slave, client

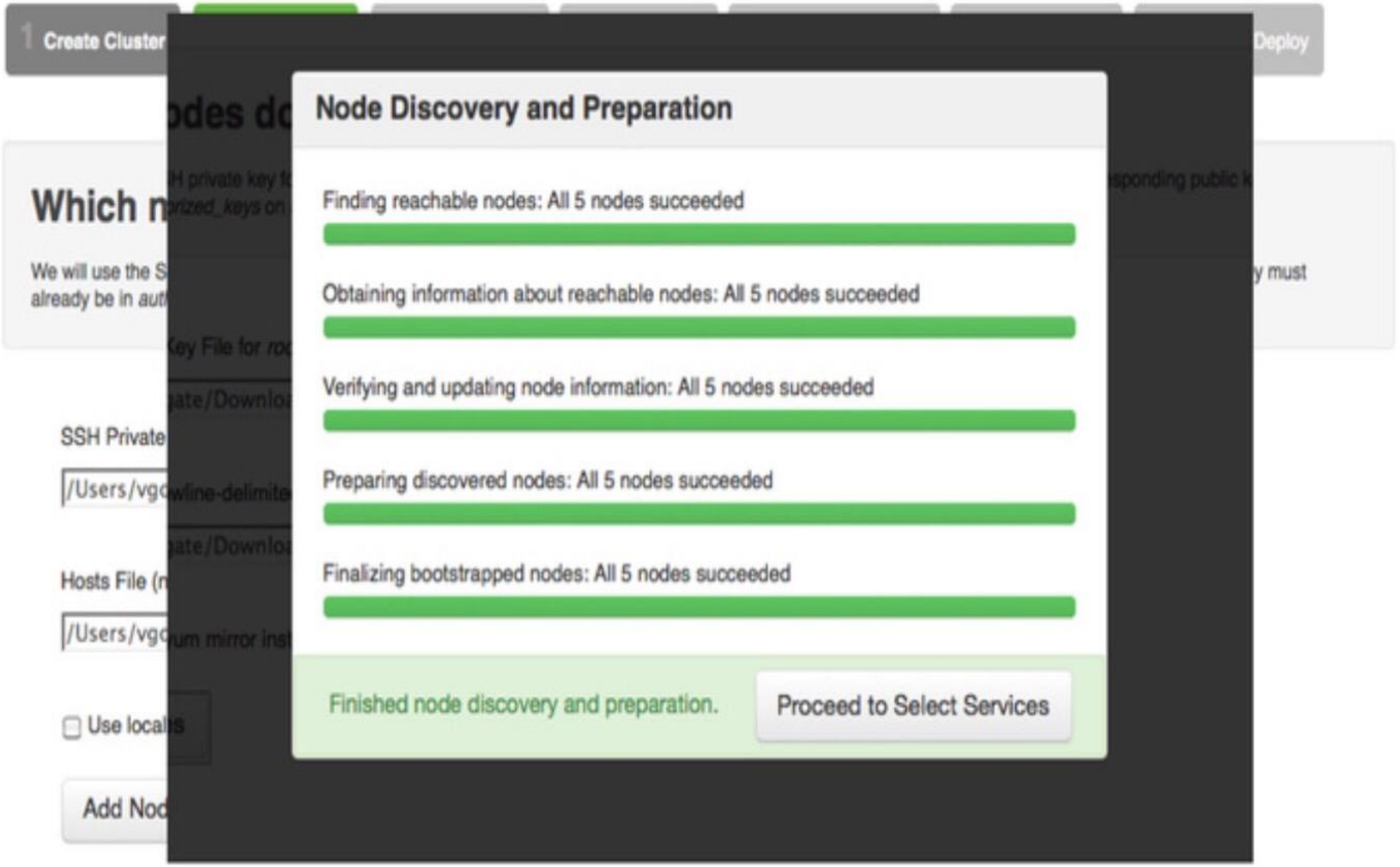
BIG DATA

Hadoop Ecosystem



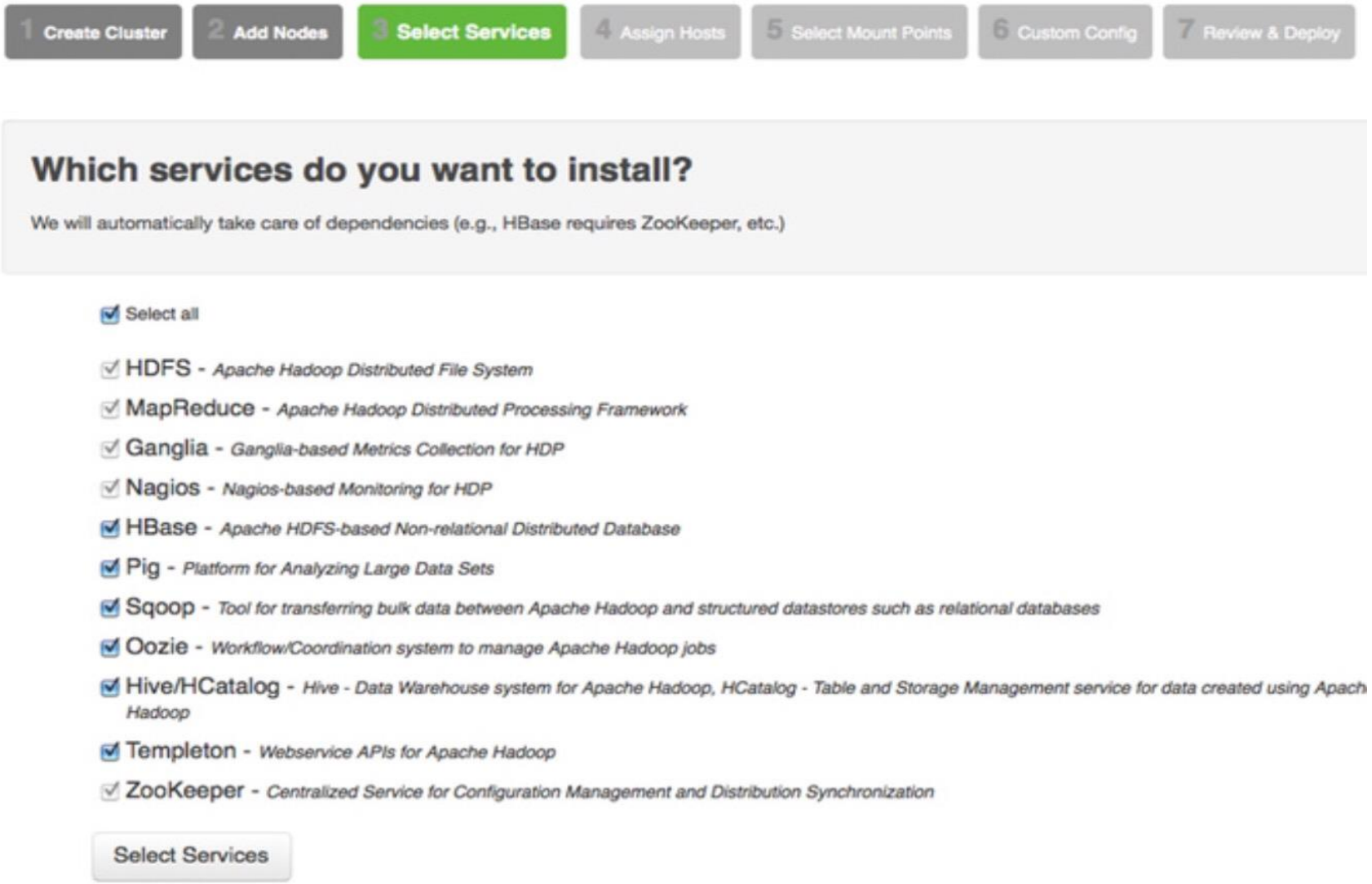
BIG DATA

Hadoop Ecosystem



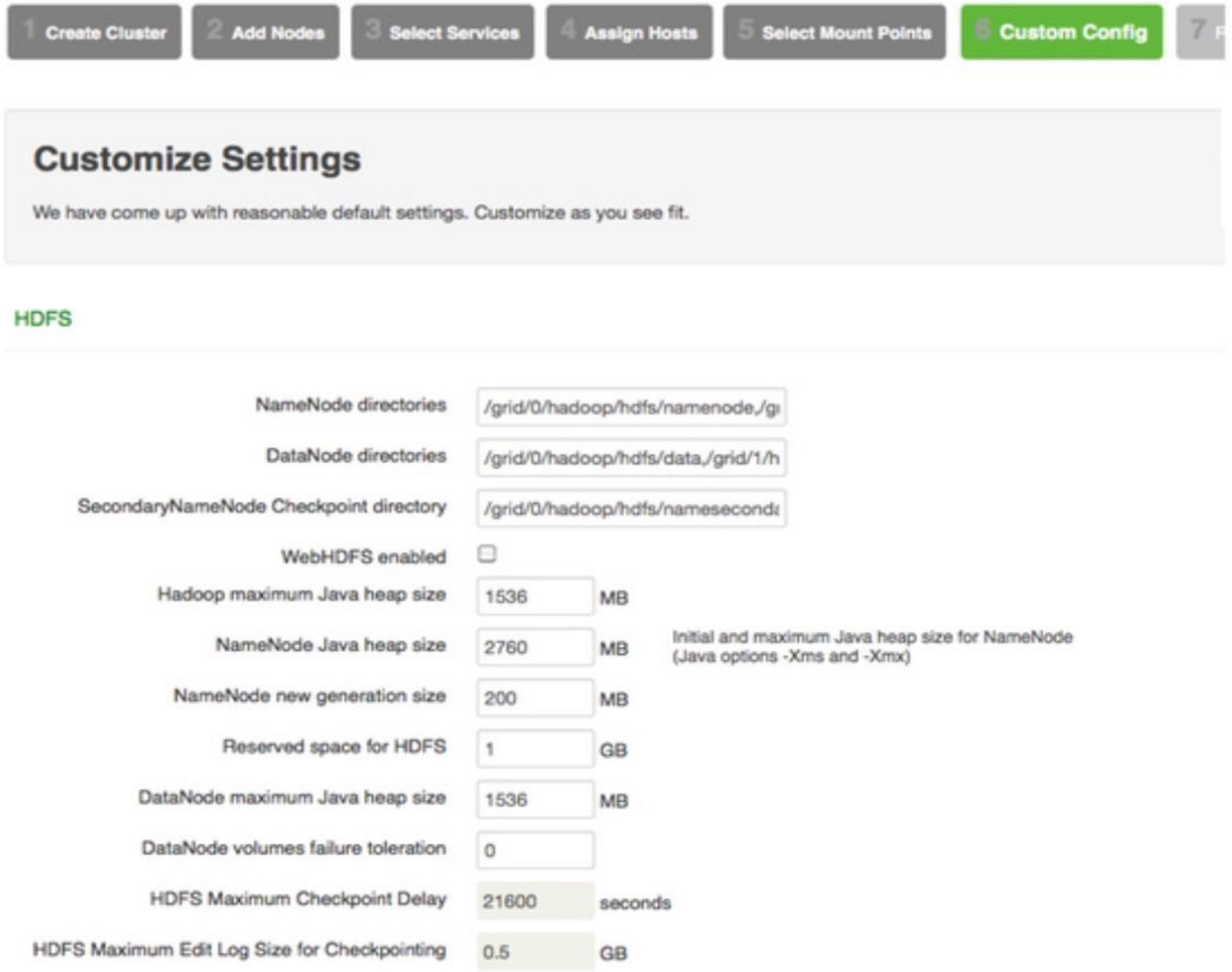
BIG DATA

Hadoop Ecosystem



BIG DATA

Hadoop Ecosystem



BIG DATA

Hadoop Ecosystem

Pig : Building High-Level Dataflows over Map Reduce

Disadvantages of MapReduce

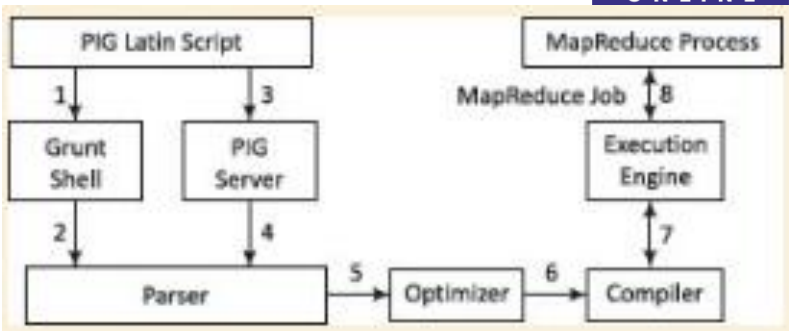
- For Data analysis, Map-Reduce is too low-level
- Writing Map and Reduce code requires retraining.

- Something SQL-Like may be better
  - HIVE is an option (which we will look at later)
  - But how about a scripting language

BIG DATA

PIG Introduction

- Complex data transformations using scripts
  - Builtin operators – join, group, filter, limit…
- Interactive shell → Grunt
- Language → Pig Latin



- Scripts are internally converted to Map Reduce jobs.

- Created @Yahoo

Image courtesy: Big Data Analytics,
Rajkamal, Preeti Saxena

BIG DATA

Example Data Analysis Task

- Find the top 10 most popular IPL matches in each venue.

Visits

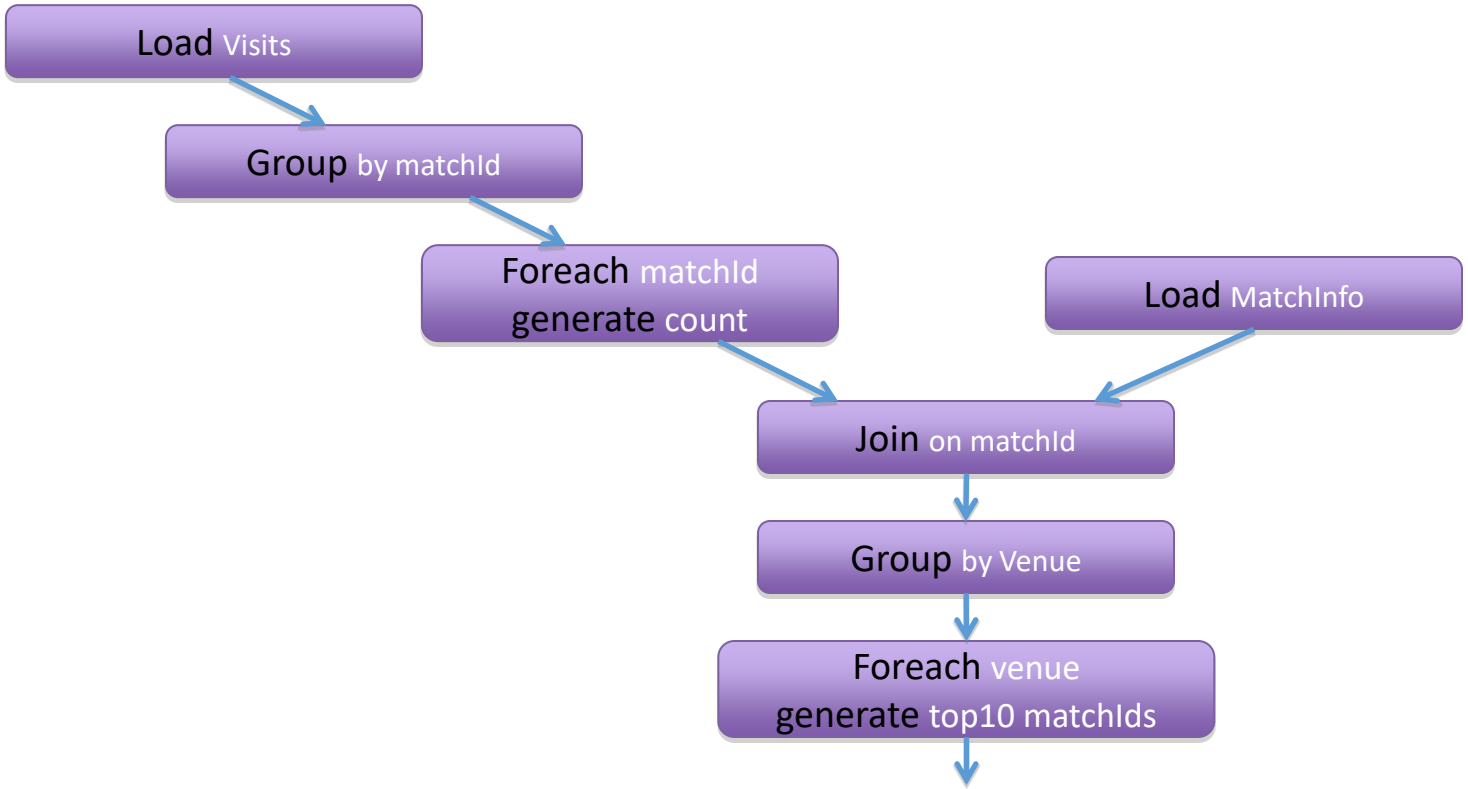
User	matchId	Time
RajniKan	Match 400	2:00
RajniKan	Match 201	5:00
Superman	Match 42	10:05
Spiderman	Match 108	13:03

MatchInfo

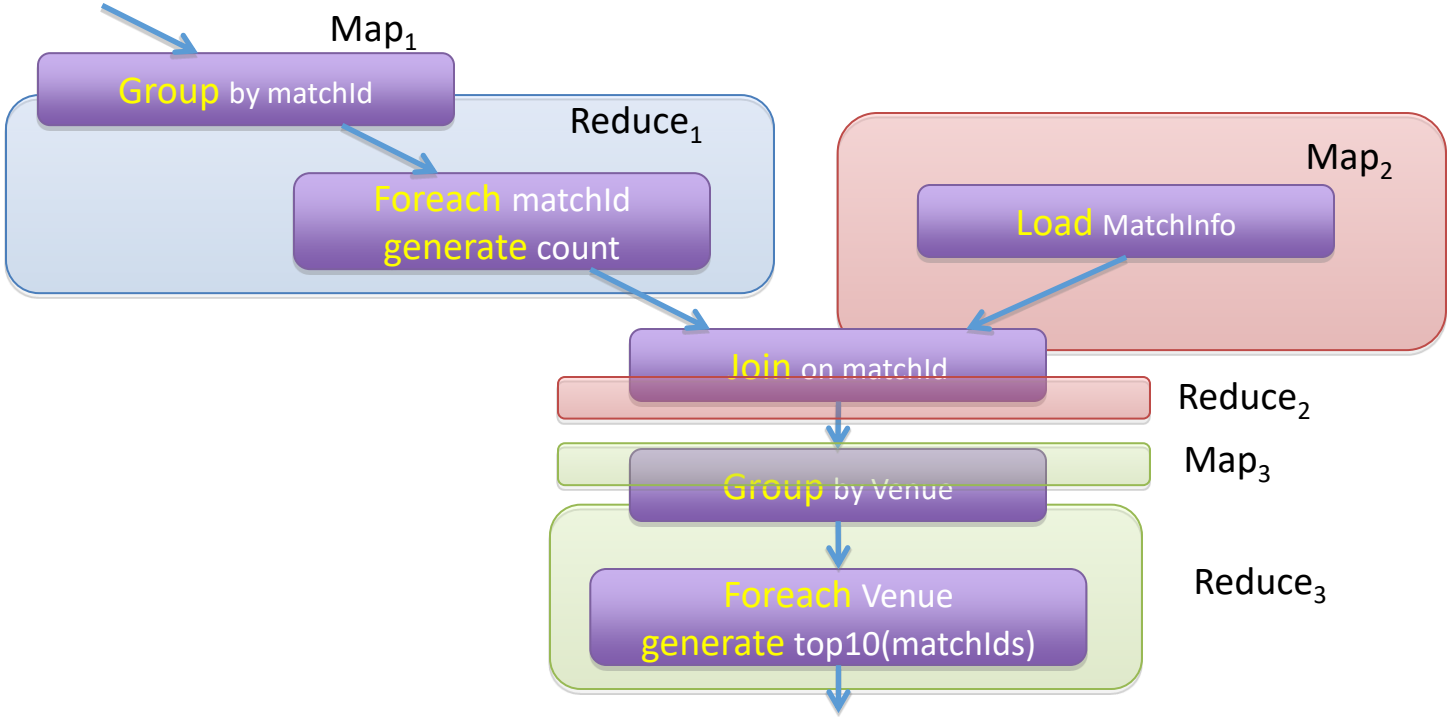
matchId	Venue	Winner
Match 108	Chennai	CSK
Match 201	Bengaluru	RCB
Match 42	Kolkata	DC
Match 400	Mumbai	RCB

BIG DATA

Data Flow



Compilation into Map-Reduce



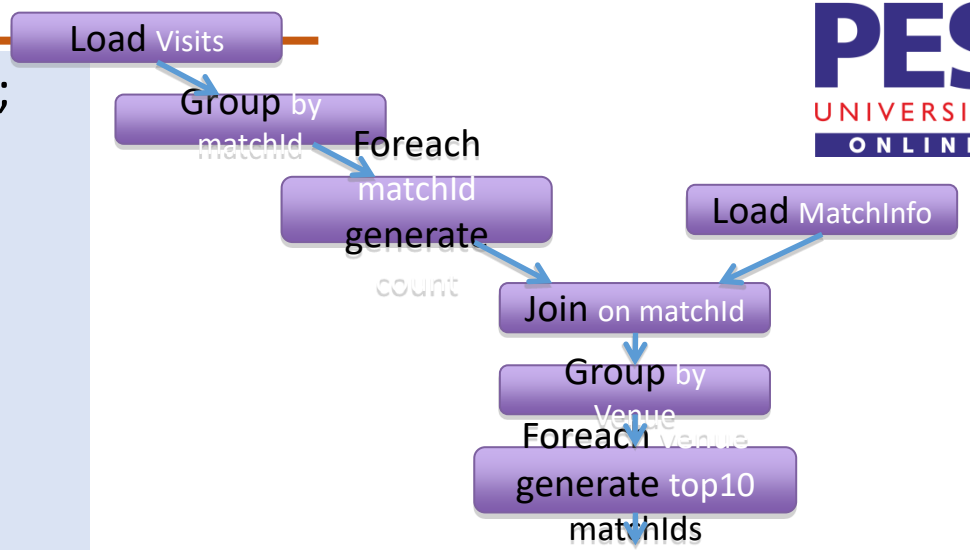
BIG DATA

In Pig Latin

- visits = load '/ipldata/visits' as (user,matchid, time);
- gMatches = group visits by matchId;
- matchPopularity = foreach gMatches generate matchId, count(visits);

- matchInfo = load '/ipldata/matchInfo' as (url, venue, winner);
- venueCounts = join gMatches by matchId, matchInfo by matchId;
- gVenues = group venueCounts by venue;
- topMatches = foreach gVenues generate top(matchPopularity,10);

- store topMatches into '/data/topMatches';



BIG DATA

Pig References

- T1 – Chapter 4.6 – you are not expected to memorize the syntax of Pig. Just the basics and how it is converted to map reduce tasks. Please go through the entire section if you want to learn to code.
- T2 – Chapter 7.5

SQOOP

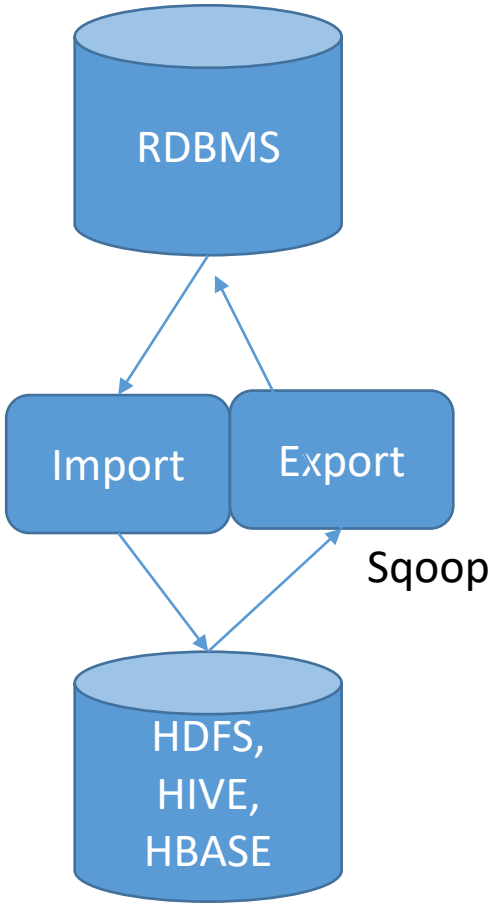
Why Sqoop ?

- Sometimes we need to use data periodically from a
  - Data warehouse
  - SQL database
- For performing analytics
- And store the data back into an SQL database
- SQOOP → SQL to Hadoop provides this functionality

BIG DATA

What is SQOOP?

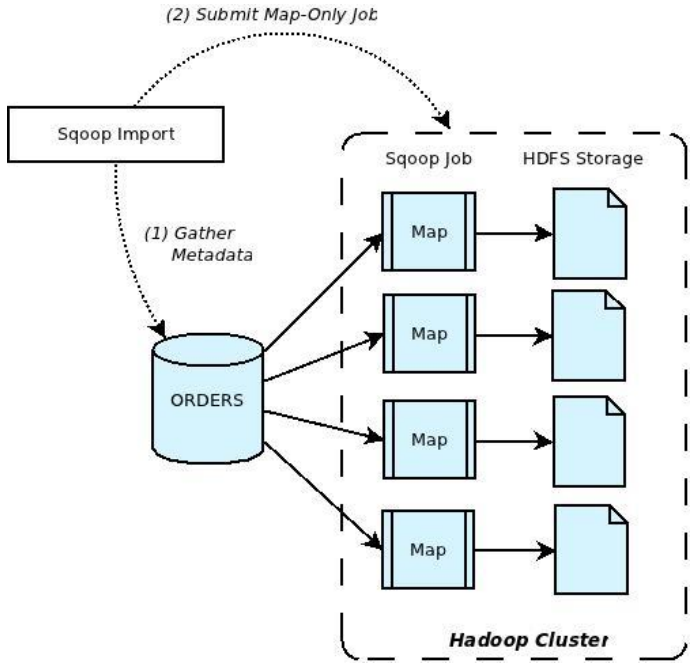
- Bulk Data Transfer Tool – voluminous data
- Import/Export data to/from SQL
  - Defines schema for import
- Integrates with Oozie as an action
- Support plugins for data sources
  - Let's say a newer database that is not supported by default.



BIG DATA

How does sqoop work? (IMPORT)

- Step 1
  - Inspects database to gather necessary metadata on data being imported
- Step 2
  - Transfers the data
    - Map only hadoop job
    - Stores data to hdfs directory
    - Imports into csv file, with newline as record separator



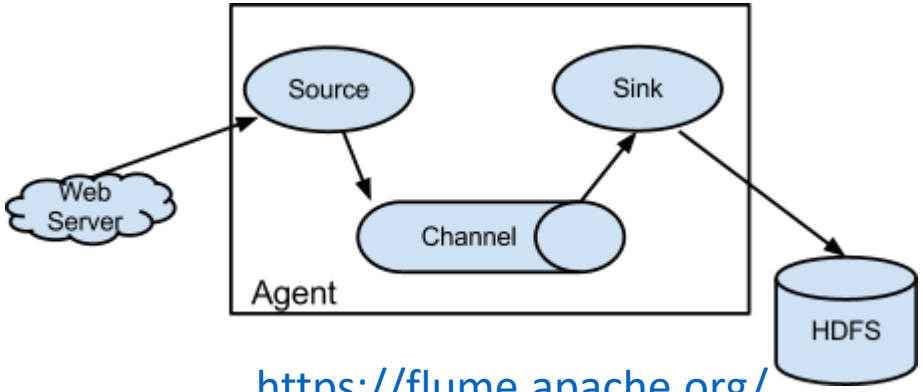
https://blogs.apache.org/sqoop/entry/apache_sqoop_overview

FLUME

BIG DATA

What is Flume ?

- Meant for collecting large amounts of streaming data
  - Events
  - Logs – like web server logs
- Architecture
  - Sources – accept data from an application
  - Sinks – receive data and store into HDFS
  - Channels – connect sources to sinks
  - Agents run the sources and sinks within Flume



https://flume.apache.org/

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu

BIG DATA

MapReduce Algorithms – Matrix Multiplication

K V Subramaniam
Computer Science and Engineering

Matrices and Vectors - introduction

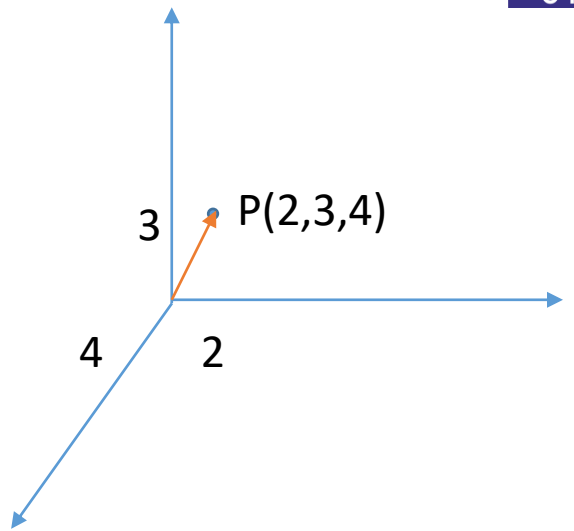
Overview

- Matrix Multiplication algorithms
  - Fundamental to many computations, including Page Rank

- Source
  - Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.
  - http://infolab.stanford.edu/~ullman/mmds/book.pdf
  - 4.3.2 of T1

Background: Vectors and Matrices

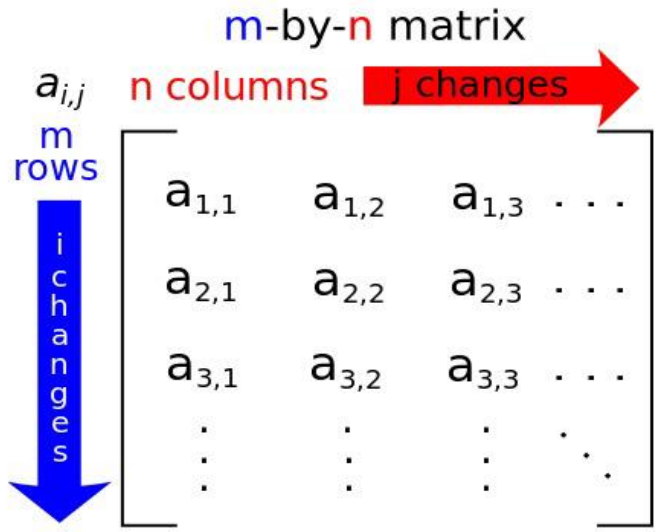
- Vectors
  - Can be defined as an ordered list of numbers
  - Visualization
    - An arrow where the direction of the vector is given by the relative size of the components
- Some Common Operations
  - Addition: $\boldsymbol{v}+w$
    - Add components
  - Scalar multiplication $a\boldsymbol{v}$
    - Multiply each component by constant



BIG DATA

Matrix

- Rectangular array of numbers.
- The numbers are called the elements of the matrix.
- An *mxn* matrix has *m* rows and *n* columns
  - Can be considered as a collection of
    - *m* row vectors
    - *n* column vectors
  - An *nxn* matrix is called a square matrix.
- Vector can be considered as a
  - *1xn* matrix (row matrix)
  - *nx1* matrix (column matrix)



Each element of a matrix is often denoted by a variable with two subscripts. For example, $a_{2,1}$ represents the element at the second row and first column of a matrix **A**.

BIG DATA

Matrix Vector Multiplication – Definition

- Multiply each row vector of ***A*** by the corresponding elements of ***x*** and sum
- Multiplying a *mxn* matrix by an *n* element vector

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 3 & 1 & 2 \end{bmatrix}. \qquad \mathbf{x} = (x, y, z)$$

$$A\mathbf{x} = \begin{bmatrix} 1 & 0 & -1 \\ 3 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x - z \\ 3x + y + 2z \end{bmatrix}$$

$$= (x - z, 3x + y + 2z).$$

Traditional Representation of Matrices

- Typically, matrices are stored as multi-dimensional arrays in programs
- int A[10][10]

  allocates 100 integers and is accessed as a 10x10 matrix

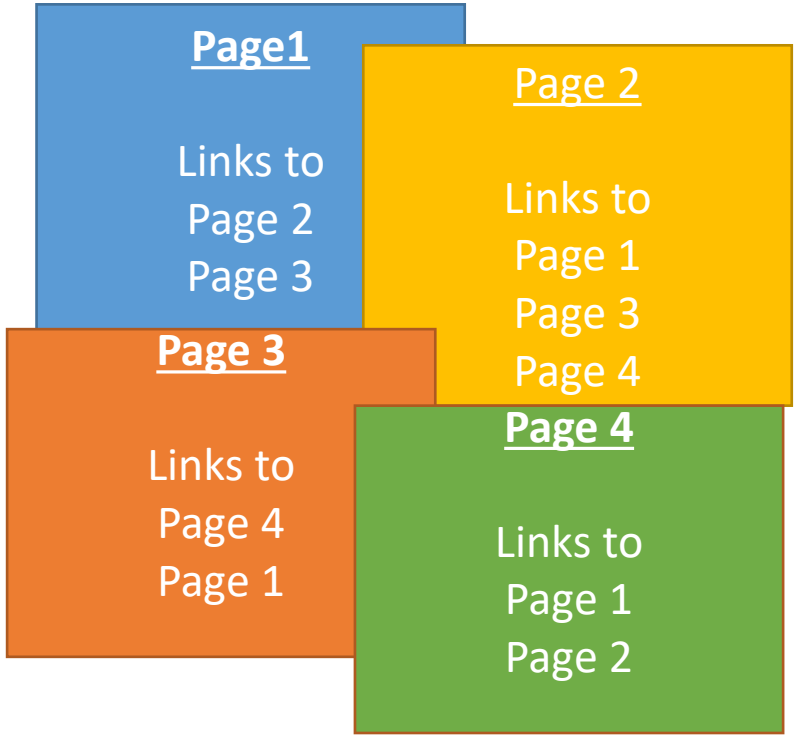
Space required to store the matrix – =
10x10*sizeof(int)= 100*sizeof(int)=100*4 = 400 bytes.

In general, we need n^2 integers to store an ***nxn*** matrix

Matrix Representation of WWW

BIG DATA

How do you represent the pages in WWW?



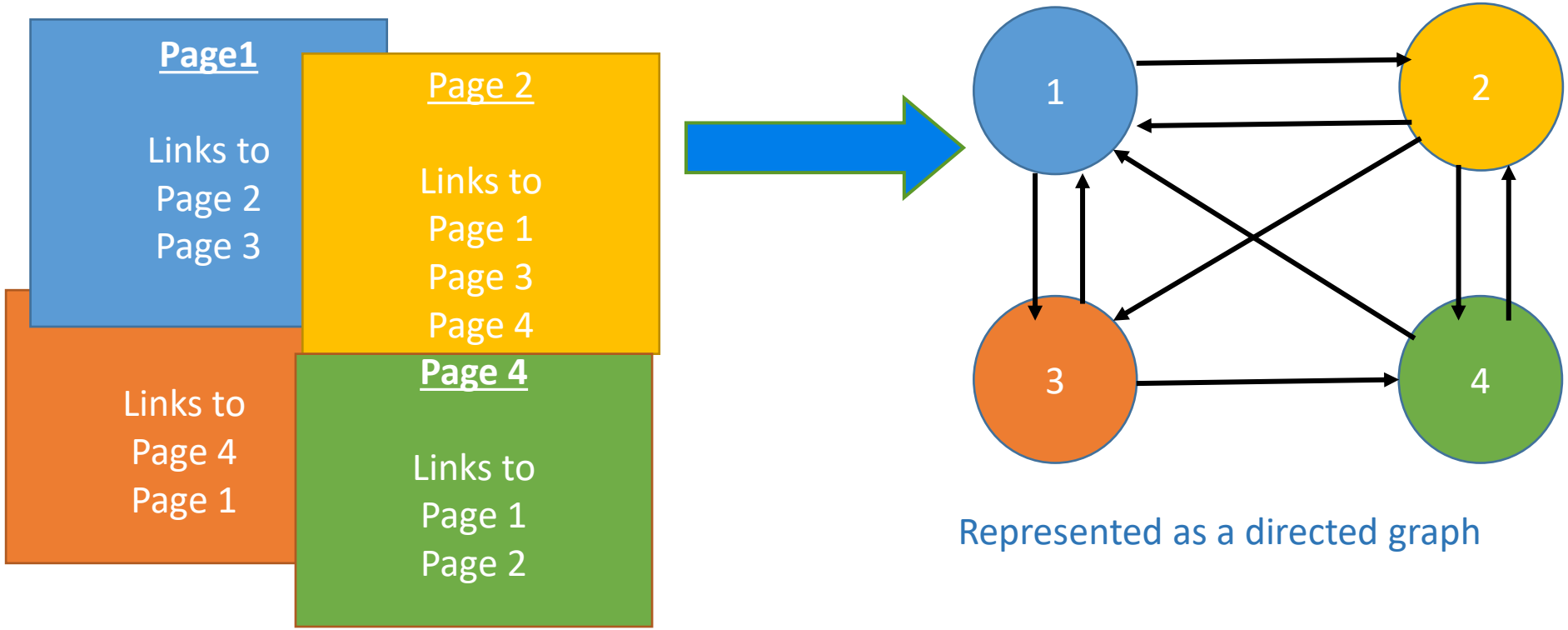
Pages in the WWW

Consider a sample of the internet that contains 4 pages

- How should we represent this?

BIG DATA

Modelling the WWW as a directed graph

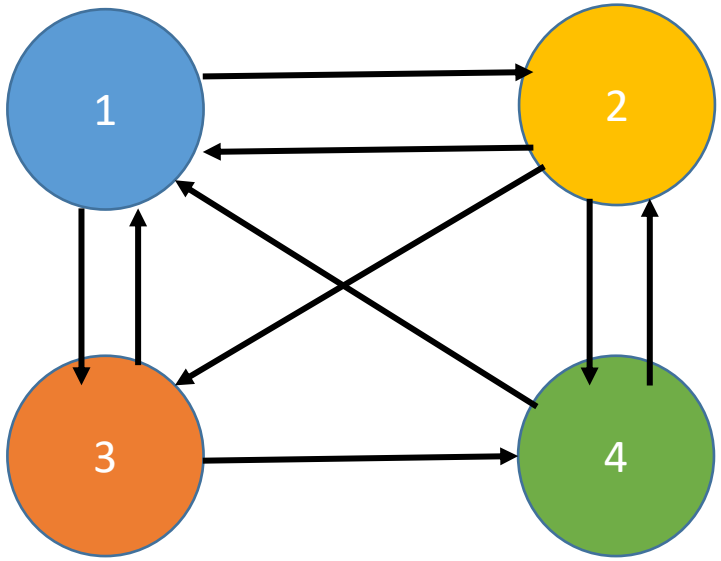


Pages in the WWW

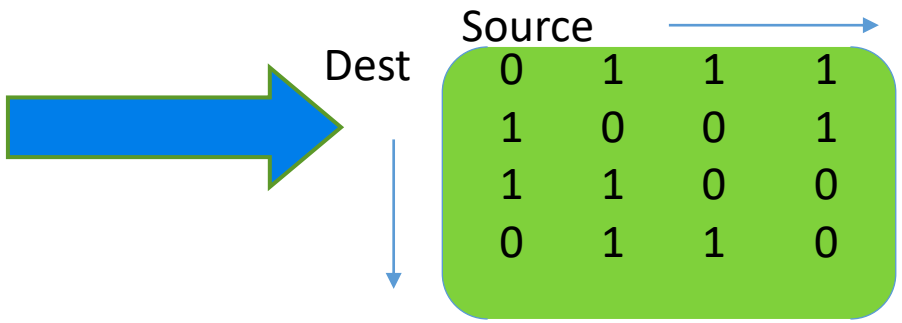
Represented as a directed graph

BIG DATA

Representing the graph as an Adjacency Matrix



Directed Graph

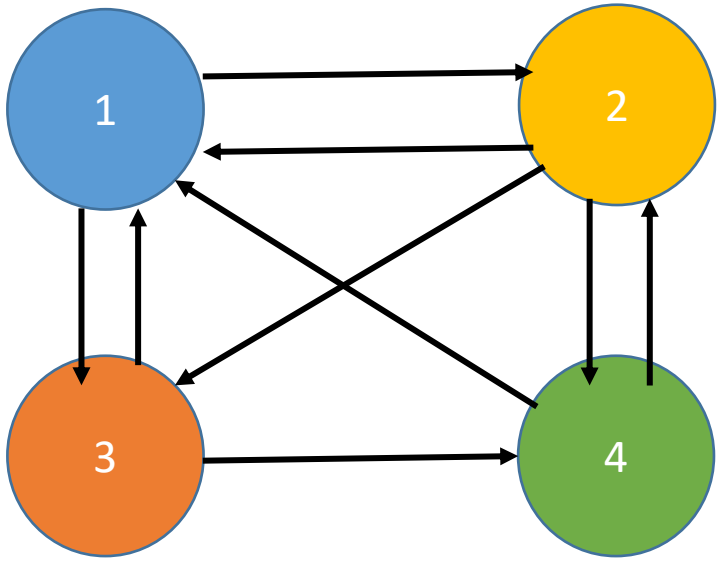


- This is fine for a small graph – 4 pages.
- But internet is large – billions of pages.
- How much storage will we require?

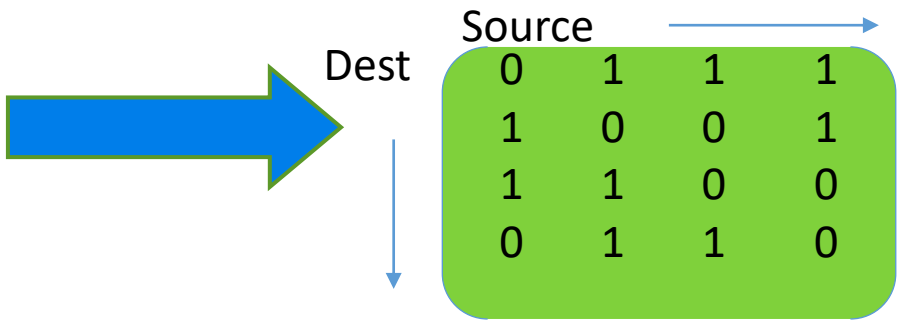
Large scale matrix representations

BIG DATA

Representing the graph as an Adjacency Matrix



Directed Graph



- Internet is large – billions of pages.
- Note – most of the entries will be 0
- Store as a ***sparse*** matrix..

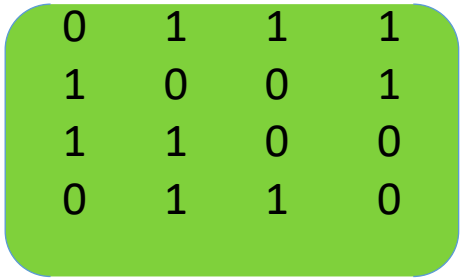
Sparse Matrix representation

- In Big Data, we deal with large matrices
  - e.g, n will be the order of 10^{10} if n is number of web pages

- And it will be a sparse matrix
- Won't fit in the memory (DRAM)
- Have to store it in HDFS

BIG DATA

HDFS Sparse matrix representation



- Store only non-zero elements as a separate record in CSV format
- For each element store
  - *<row_number, column_number, value>*
  - As the format
- As many entries as there are links
- Exercise –Store the graph given on the right into a HDFS CSV file

BIG DATA

Solution

1, 2, 1
1, 3, 1
1, 4, 1
2, 1, 1
2, 4, 1
3, 1, 1
3, 2, 1
4, 2, 1
4, 3, 1

0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

As an exercise, try saving this in a file and loading it onto HDFS that you have installed.

Matrix Vector Multiplication

Matrix Vector multiplication with MapReduce

- To multiply an *nxn* matrix ***M*** with an *n*-element vector ***v***, compute

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix}$$

Matrix Vector Multiplication with MapReduce

- Let us assume that the vector ***v*** fits into memory
- Vector v is shared by all the mappers
- M_{ij} is stored as a CSV file on HDFS and is distributed across multiple nodes

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{bmatrix}$$

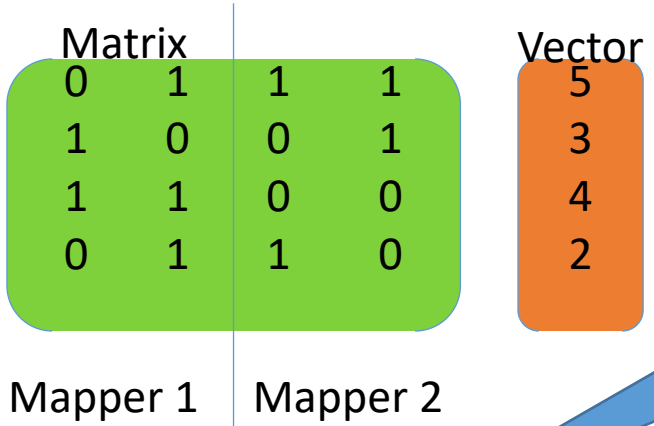
2

Matrix Vector Multiplication with MapReduce

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

- **map**:
  - Computes the partial product
  - Uses the key as i → the index into the target vector
  - output (i, $m_{ij}v_j$)
- **reduce**:
  - Sums all the partial products.

Working of the MR algorithm – Map Stage

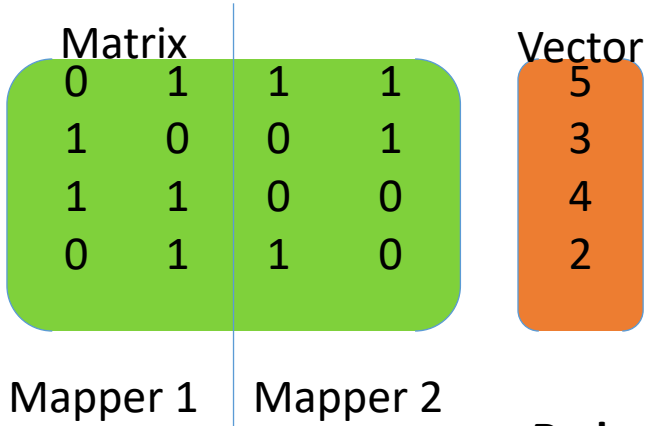


Note that the key is the index into the vector where this value will contribute

Key	Value
1	1*3 = 3
2	1*5 = 5
3	1*5= 5
3	1*3=3
4	1*3=3

Key	Value
1	1*4=4
1	1*2=2
2	1*2=2
4	1*4=4

Working of the MR algorithm – Reduce Stage



Reducer Input

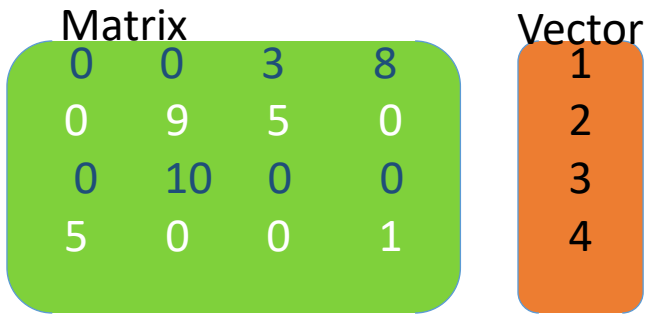
Key	Intermediate Value List
1	2,3,4
2	2, 5
3	3, 5
4	3, 4

Reducer output

Key	Value
1	9
1	7
2	8
4	7

BIG DATA

Review Problem



- Assuming rows 1 and 3 are in datanode1 and 2 and 4 are in datanode 2, perform a matrix multiplication using Map Reduce
- Show inputs/outputs of mappers/reducers as dicussed in previous slides

Matrix Vector Multiplication - extensions

BIG DATA

Matrix-Vector Multiplication using Mapreduce - 2

- Case 2: ***v*** doesn't fit into main memory.
- Partition *M* and ***v*** into stripes.

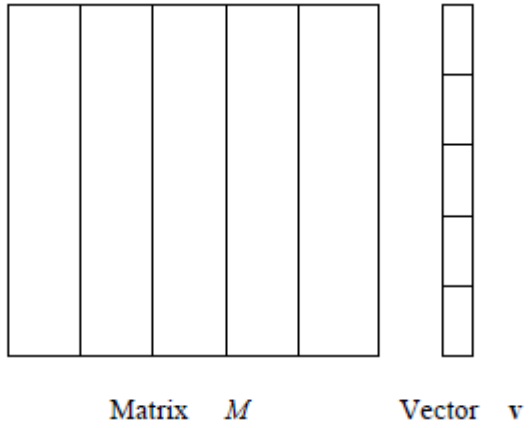


Figure 2.4: Division of a matrix and vector into five stripes

- The same MapReduce algorithm can be used.

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu,
ushadevibg@pes.edu

BIG DATA

Big Data Algorithms

Page Rank Computations

K V Subramaniam
Computer Science and Engineering

Matrix Multiplication

Overview

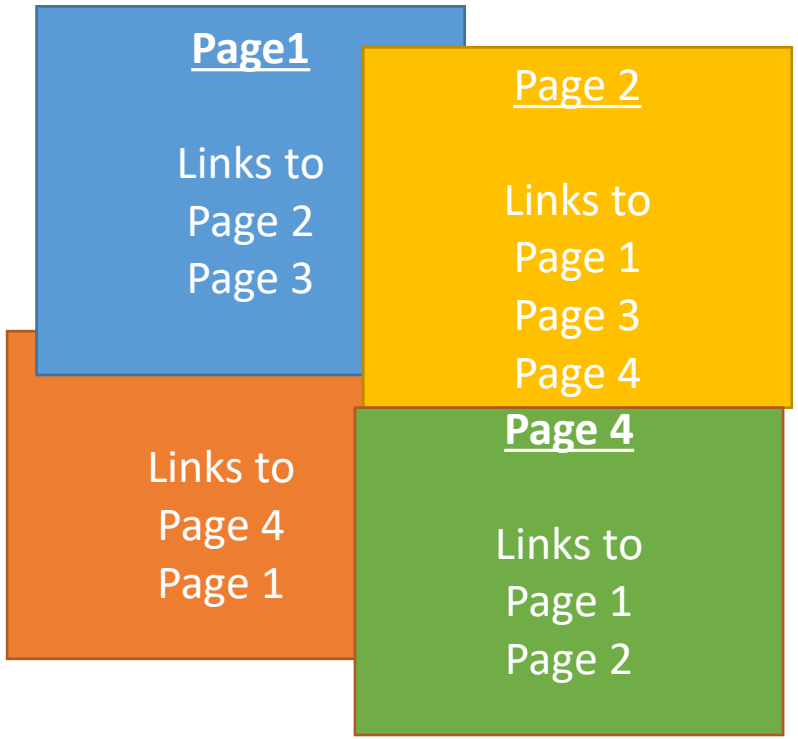
- Page Rank

- Source
  - Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.
  - http://infolab.stanford.edu/~ullman/mmds/book.pdf

Page Rank : Overview

BIG DATA

Search Engine working



Pages in the WWW

Search for a term e.g “Big Data

Lookup an inverted index: maps every word to documents that contain the term

Sort matched pages based on **importance**

Importance computation based on usage of terms within the page.

BIG DATA

Issues with early search engines

- Term Spam
  - Unethical use → add terms multiple times

- Example
  - I could setup a page with PES University occurring a 1000 times in the page.
  - Search Engine would think that my page is an authority for PES Univesity → mark it as ***important***
  - How to fix this?

BIG DATA

One possible solution

- Instead of taking terms in my page
- Consider how many pages are pointing into my page?
- Will this not solve the problem.

- NO!!
- Can add many spam pages that point to my page.

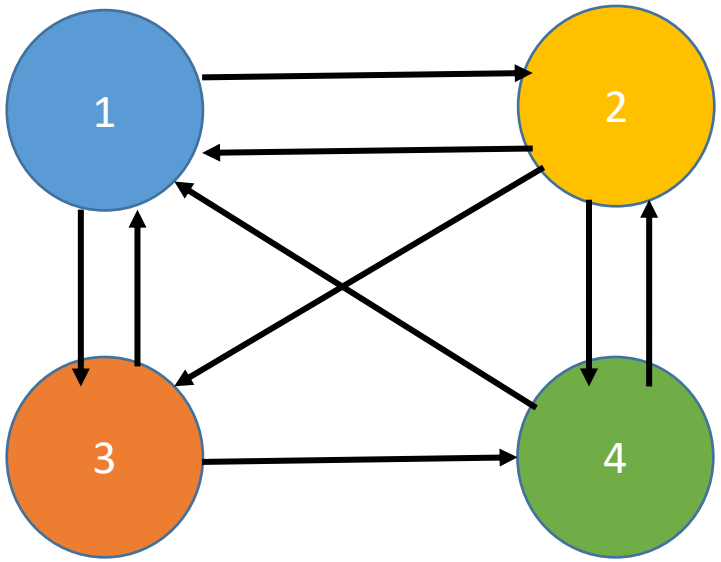
How to solve this

- Two techniques
  - Consider random surfers starting at a random pages
    - What fraction of surfers end up at my page?
    - This gives an indication of the importance of my page.
    - *Page Rank*
  - Take into account the terms near the links present in pages that point to my page
    - Gives an indication of how relevant my page is.

Page Rank : Transition Matrix

BIG DATA

Adjacency Matrix

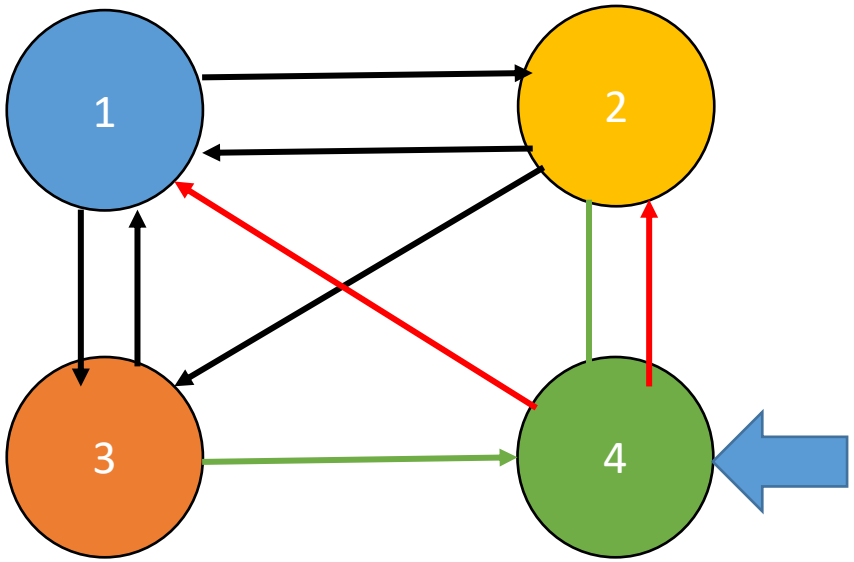


0	1	1	1
1	0	0	1
1	1	0	0
0	1	1	0

Let us start from our graph model of the internet

- Adjacency matrix represents which pages are reachable from a given page

Adjacency Matrix



If we are at node 4 (page 4) and we follow a link out

Then

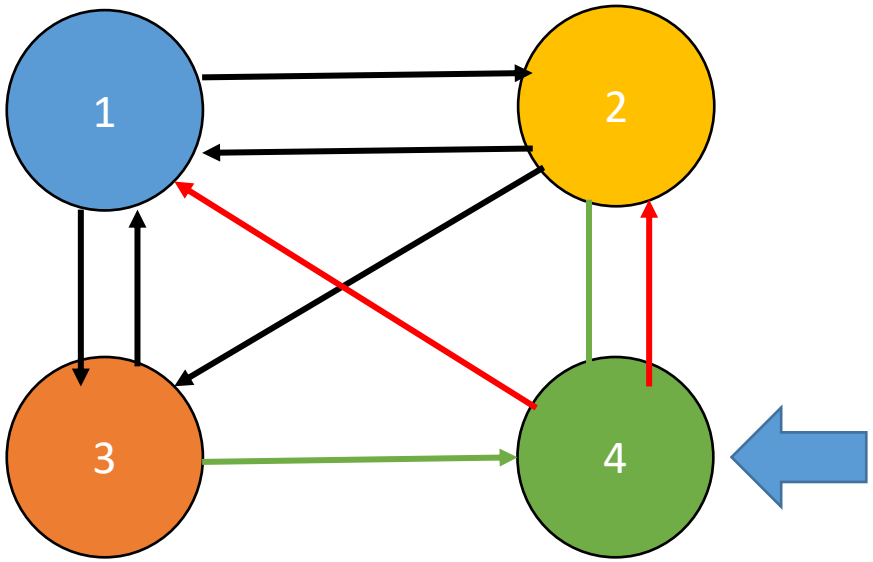
- the pages we can visit are 1 and 2
- 3 is not reachable directly from 4

To summarize for node 4

#in links = 2

#out links = 2

Random surfer



Consider, a random surfer starting from page 4

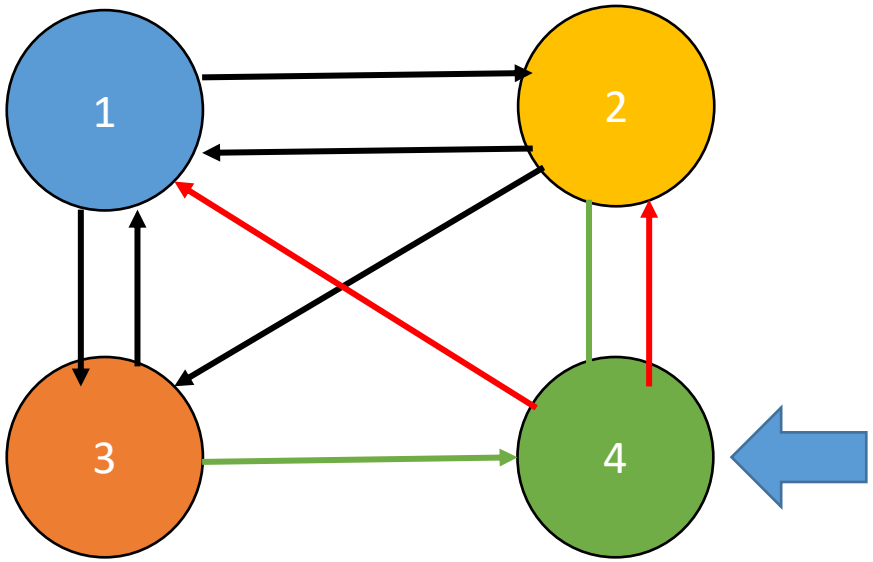
Assume, that we have equal probability of taking either out link

So,

P(transition at node 4) = 1/#outlinks

= ½

Transitioning at a node



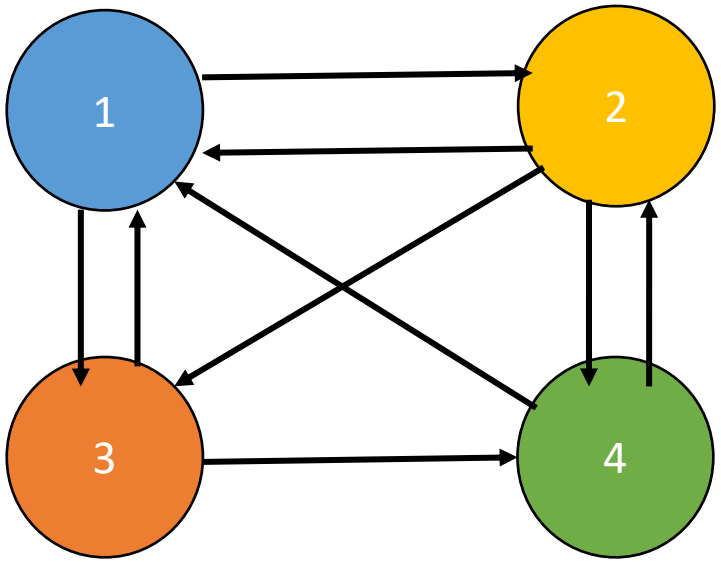
For node for we can represent the probabilities of directly transitioning

- To every other page as a column vector

	Source
Dest	1/2
	1/2
	0
	0

BIG DATA

Transition Matrix



For the entire graph with n nodes, this would be a *nxn* matrix

- Transition matrix **(M)**
- Each entry represents the probability of transition from a source to a destination
- Source → column #
- Destination → row #

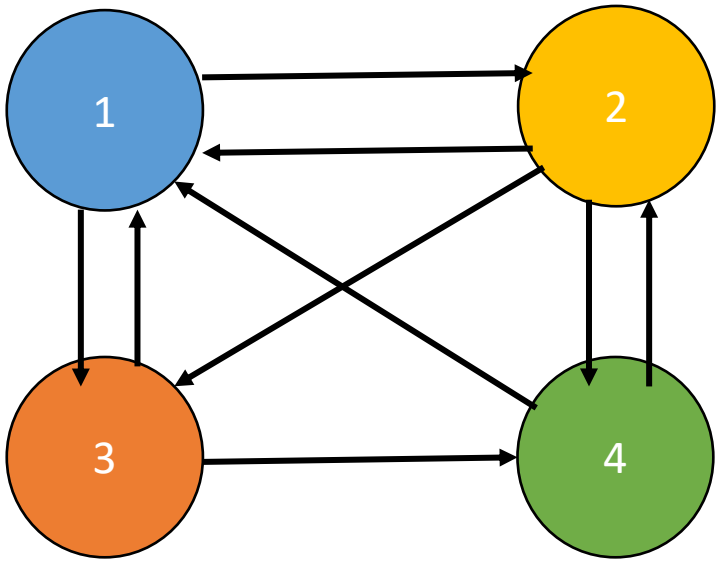
Source →

Dest ↓

0	1/3	1/2	1/2
1/2	0	0	1/2
1/2	1/3	0	0
0	1/3	1/2	0

Page Rank : Following the random surfer

Random surfer - initialization



The random surfer can start off on any of the nodes

So each node has equal chance of being a starting node.

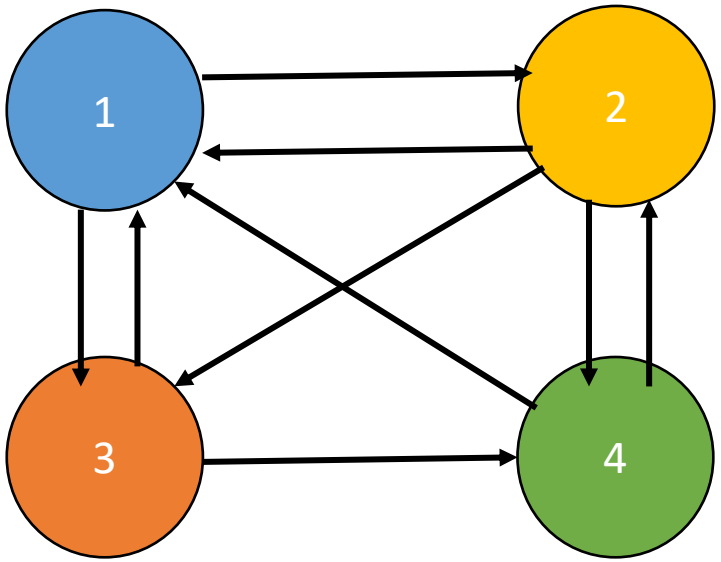
Lets call the relative importance of each node - → ***importance*** represented as vector ***v***

Our first guess at this is called $\boldsymbol{v_0}$

$$\boldsymbol{v_0} = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$ Node importance

BIG DATA

Random surfer - movement



As random surfer moves once

Compute chance they will land up in each of the nodes

For each node we compute the probability of ending up at that node based on previous node

Multiply transition matrix and ***v***

0	1/3	1/2	1/2
1/2	0	0	1/2
1/2	1/3	0	0
0	1/3	1/2	0

v_0

1/4
1/4
1/4
1/4

=

1/3*1/4+1/2*1/4+1/2*1/4
½*1/4+1/2*1/4
½*1/4+1/3*1/4
1/3*1/4+1/2*1/4

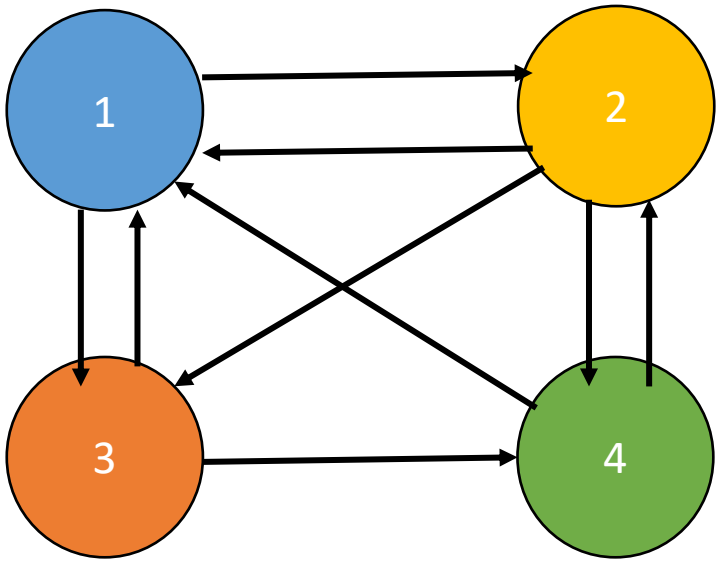
=

v_1

1/3
1/4
5/24
5/24

BIG DATA

Random surfer - movement



With each move of the random surfer, we will repeat the computation

Multiply transition matrix and ***v***

0	1/3	1/2	1/2
1/2	0	0	1/2
1/2	1/3	0	0
0	1/3	1/2	0

v_1

1/3
1/4
5/24
5/24

= v_2

Page Rank : As an Eigen vector problem

BIG DATA

Iterative Algorithms

- x satisfies the equation to the right.
  - Generally: some set of equations for x.
- How do we solve for x?
- Iterative algorithm: widely used in Big Data
  - i = number of times we have looped,
  - x_i = value of x on i^{th} iteration,
  - Calculate some "error term" based upon x_i .
    - Shows how far x_i is from the correct value.
    - E.g., $Ax_i - x_i$
  - Derive x_{i+1} based upon x_i and error term.
  - Loop over steps 3 and 4 until error term is small.
- Iterative algorithm calculates $x_0\ x_1\ x_2\ x_3$ … which eventually converges to a good (or the good) solution.

$$A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

BIG DATA

Iterative Algorithms: Fixed Point Iteration

- Suppose the equation that x satisfies has the form x=f(x)
- x may be a matrix or vector, then the following simplified iterative algorithm frequently works

- Fixed point iteration
  - Initialize x_0 to some value
  - Calculate $x_{i+1} = f(x_i)$
  - Loop over step 2 until either
    - error term = $|x_{i+1}-x_i|$ is small i.e., not much change between x_{i+1} and x_i
    - or for some maximum number of iterations.

https://en.wikipedia.org/wiki/Fixed-point_iteration#:~:text=In%20numerical%20analysis%2C%20fixed%2Dpoint,the%20fixed%20point%20iteration%20is

BIG DATA

Eigenvalues and eigenvectors

- Multiplying a *mxn* matrix by an *n* element vector gives an *m* element vector.
- If matrix is *nxn,* the product will also be an *n* element vector. $A\mathbf{v} = \lambda\mathbf{v}$

- Suppose
  - Multiplying ***A*** by ***v*** gives back the same vector apart from a scale change .
  - λ is called the eigenvalue and ***v*** the eigenvector.
  - Many problems (e.g., page rank) can be converted into finding eigenvalues of a matrix.

Page Rank as Eigenvector Problem – Eigenvector Computation

- Initialize $\boldsymbol{v}$ to be a vector of equal values.

- Loop
  - $\boldsymbol{v}_{i+1} = \boldsymbol{A}\boldsymbol{v}_i$

- Until there is little difference between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$
- At this point
  - $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{v}$

- $\boldsymbol{v}$ is the eigenvector and page rank of the transition matrix of the web graph.

Page Rank as Eigenvector Problem – Overview

- We want to compute the *importance* (page rank) of each Web page.

- Assume that
  - If a page has importance *I*.
  - *n* links to other pages
  - It distributes it's importance *I* among the *n* links equally (*I*/*n*)
- Use this assumption to calculate page rank

Page Rank : Map Reduce Implementation

Implementation in Map Reduce

- We need support for
  - Handling multiple files as input to the mapper
    - The initial page rank
    - Part of the Transition Matrix
  - Iteration over multiple matrix-vector multiplication rounds

BIG DATA

Hadoop Multiple Input Files – mapper

```
public class multiInputFile extends Configured implements Tool
{

 public static class CounterMapper extends Mapper
 {
  public void map(LongWritable key, Text value, Context context)
  throws IOException, InterruptedException
  {
   String[] line=value.toString().split("\t");

   context.write(new Text(line[0]), new Text(line[1]));
  }
 }


 public static class CountertwoMapper extends Mapper
 {
  public void map(LongWritable key, Text value, Context context)
  throws IOException, InterruptedException
  {
   String[] line=value.toString().split("\t");
   context.write(new Text(line[0]), new Text(line[1]));
  }
 }
```

Mappers write first word as key, 2nd word as value

BIG DATA

Hadoop Multiple Input Files – reducer

```
public static class CounterReducer extends Reducer
{
 String line=null;

 public void reduce(Text key, Iterable values, Context context )
 throws IOException, InterruptedException
 {

  for(Text value:values)
  {
   line = value.toString();
  }

  context.write(key, new Text(line));
 }
}
```

Loop over values
Write each value as separate record

BIG DATA

Hadoop Multiple Input Files – job

```
 public int run(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = new Job(conf, "aggprog");
 job.setJarByClass(multiInputFile.class);
 MultipleInputs.addInputPath(job,new Path(args[0]),TextInputFormat.class,CounterMap
per.class);
 MultipleInputs.addInputPath(job,new Path(args[1]),TextInputFormat.class,Countertwo
Mapper.class);

 FileOutputFormat.setOutputPath(job, new Path(args[2]));
 job.setReducerClass(CounterReducer.class);
 job.setNumReduceTasks(1);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(Text.class);

 return (job.waitForCompletion(true) ? 0 : 1);

 }

 public static void main(String[] args) throws Exception {


  int ecode = ToolRunner.run(new multiInputFile(), args);
  System.exit(ecode);


 }

}
```

Multiple input class

Set input file for each mapper

Iteration

- One MR step produces a estimate of ***v***
  - ***The file is stored in HDFS***

- At the end of the iteration, we need to compare this with the previous iteration estimate of ***v***

- After that, we use this file as the input for the next step of MR.

THANK YOU

K V Subramaniam, Usha Devi

Dept. of Computer Science and Engineering

subramaniamkv@pes.edu

ushadevibg@pes.edu

BIG DATA

Big Data Algorithms

– Relational Operations

K V Subramaniam
Computer Science and Engineering

BIG DATA

Class Overview

- Relational algebra overview
- Select and Project with MR
- Set Operations
- Join
- Grouping and Aggregation
- Case Study: HIVE

Relational Operations

BIG DATA

Summary of Relational Algebra

- Relations
  - Tables; columns = attributes
  - Rows = tuples; $R(A_1, A_2, \ldots, A_n)$
- Relational Operators
  - Selection: σ_C *(R)* select from *R* according to condition *C*
  - Projection σ_S *(R)* select from *R* subset of attributes *S*
  - Union, intersection, difference
  - Natural join
  - Grouping: partition *R* according to attributes *G*
  - Aggregation: *SUM, COUNT, AVG, MAX, MIN*

BIG DATA

Simple Problem

id	Name	Role	Team
1	Virat Kohli	Captain	RCB
2	Gautham Gambhir	Captain	KKR
3	Anil Kumble	Coach	MI
4	Virender Sehwag	Coach	KXIP

1. Write an SQL query to list
   a) All the details for the Coaches.
   b) Only the names of the coaches
   c) Total #coaches.
2. What type of a relational operation are we using?

BIG DATA

Simple Problem

- Data in a database is well structured.
- Instead assume that data is stored in a file in HDFS as shown below.

```
1, Virat Kohli, Captain, RCB
2, Gautham Gambhir, Captain, KKR
3, Anil Kumble, Coach, MI
4, Virender Sehwag, Coach, KXIP
```

- Now we need to perform the query.

Select and Project in MapReduce

Selection

- map
  - Read each row t of table
  - Check if it satisfies condition *C*
  - If so, output (*t*,*t*)
- Reduce
  - do nothing

How will the map output look for the query 1a ?

Map

<"3, Anil Kumble, Coach, MI","3, Anil Kumble, Coach, MI">
<"4, Virender Sehwag, Coach,KXIP","4, Virender Sehwag, Coach,KXIP">

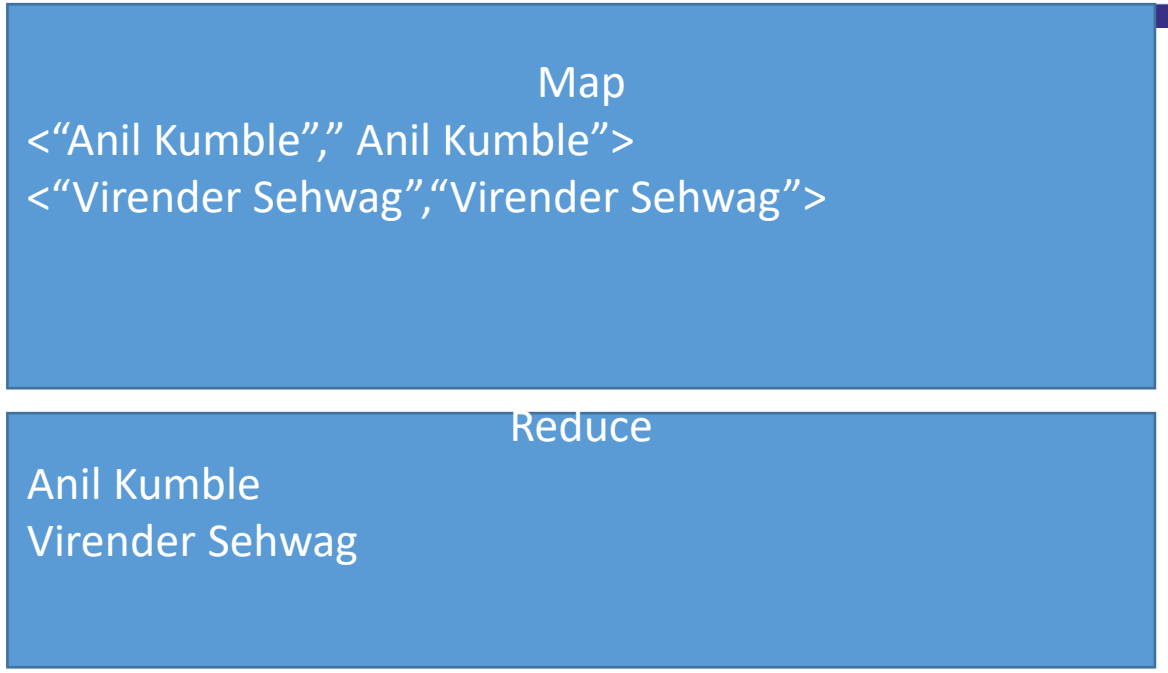
Reduce

3, Anil Kumble, Coach, MI
4, Virender Sehwag, Coach,KXIP

BIG DATA

Projection

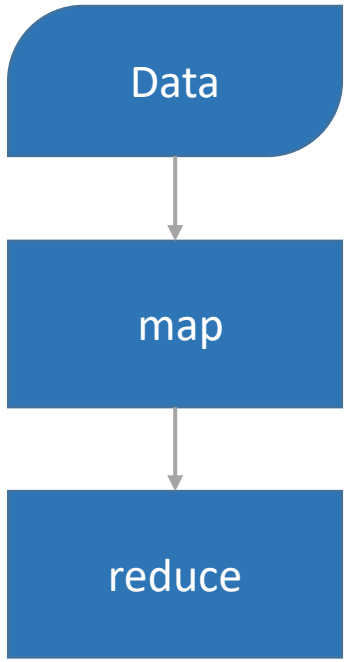
- map
  - Read each row t of table
  - Calculate subset of attributes *t'*
  - Output (*t',t'*)
- reduce
  - Eliminate duplicates
    - (*t',*[*t',t',t'*]) -> (*t',t'*)



Relational Operations requiring two input files – Set operations

Class Exercise - Union...

- Union *R U S*
  - *R* and *S* have the same structure
  - Output records in *R* or *S*
  - *We need to solve a problem here*
- The basic problem :
  - Need to read in 2 input files
  - produce 1 output file
- Reading multiple input files
  - Same problem in Matrix-vector multiplication
  - Need to read in matrix ***M*** and vector ***v***

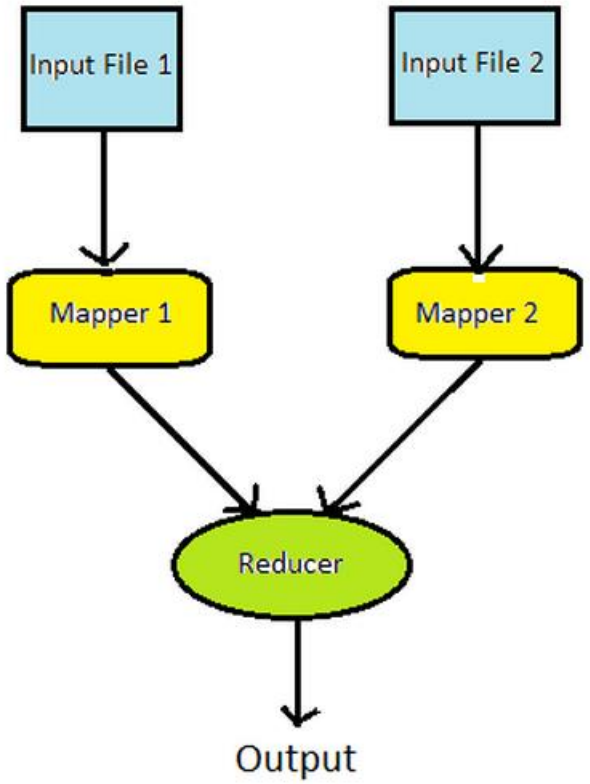


One Solution

- map reads in one file
- If we have multiple input files, we can try to combine them into one file
  - This can be used as input to map
- Problem: multiple input files can be combined in many ways
  - We can have all the records of one file followed by all the records of another file
  - Or merge the two files and sort the records
  - Files may not have the same structure

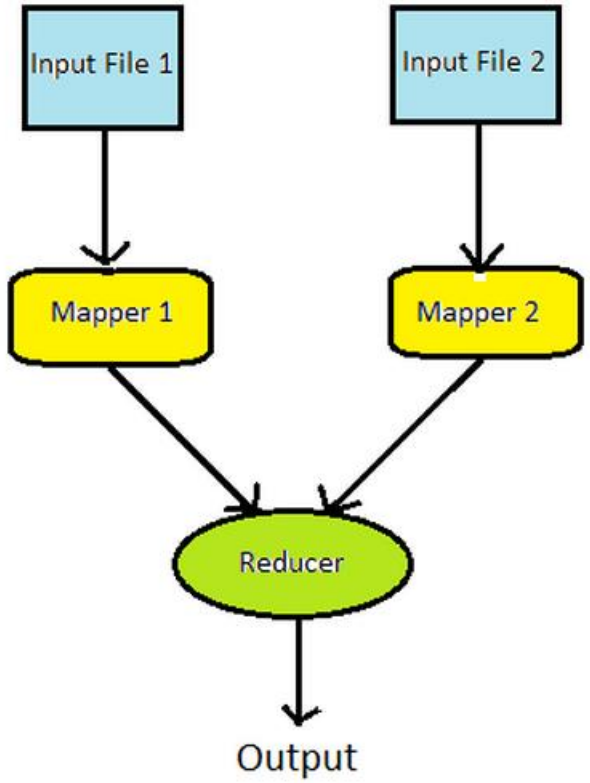
The Hadoop Solution

- Use map reduce itself to merge the files
- Example
  - We have two mappers
  - Each mapper reads in one file
  - Each mapper writes a key-value pair
  - The reducer uses the keys to merge the files
  - This output can be used as input to subsequent map reduce passes



Union...

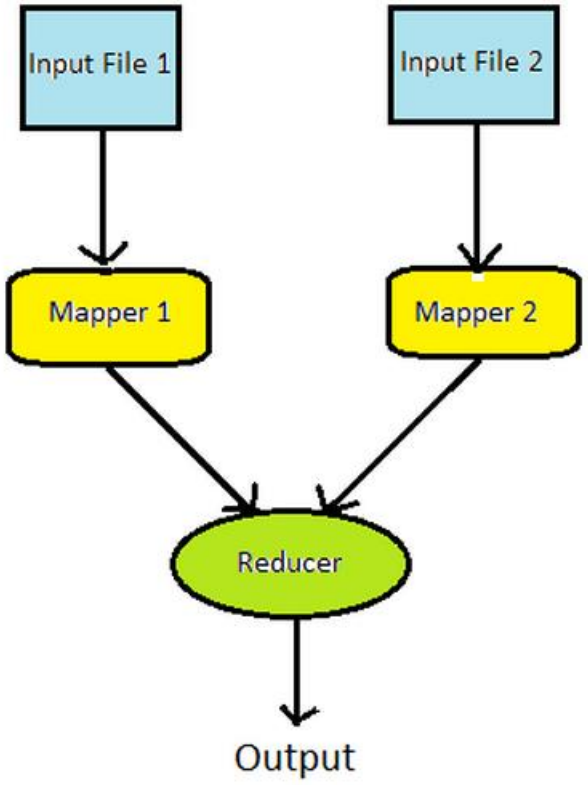
- Union *R U S*
  - Need to read in 2 input files and produce 1 output file
  - Need to have 2 mappers reading different input files
  - Can be done using *MultipleInput* option in Hadoop
    - MultipleInputs.
      - addInputPath (job,
      - new Path (args[0]),
      - TextInputFormat.class,
      - Mapper1.class);



BIG DATA

...Union

- mapper 1
  - Read each row of table *R*
  - Output (*t*,*t*)
- mapper 2
  - Read each row of table *S*
  - Output (*t*,*t*)
- reducer
  - Eliminate duplicates if any
    - (*t*,[*t*,*t*]) -> (*t*,*t*)



Sample Problem

- Given the following input for two files
  - File 1
    - A
    - B
    - C
    - D
  - File 2
    - A
    - E
    - F
    - C

- Show (for the Union algorithm)
  - Input and output of mapper 1
  - Input and output of mapper 2
  - Input and output of reducer

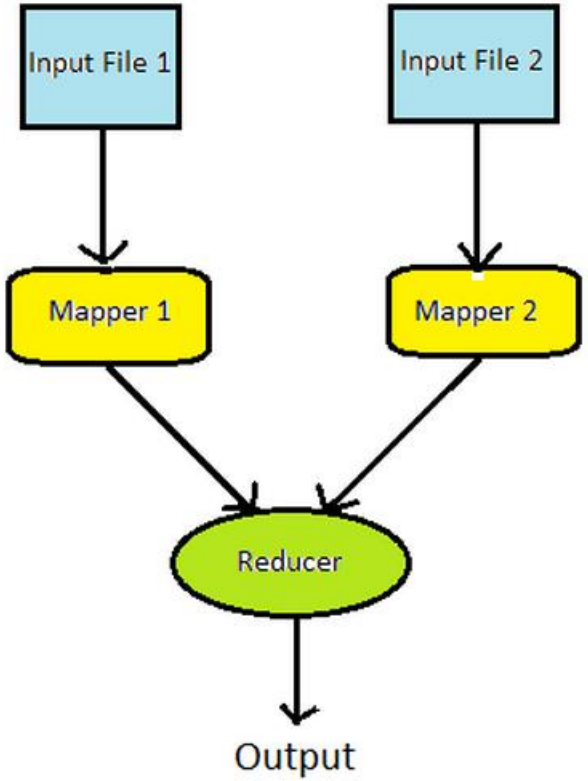
Solution

- Mapper 1 output
  - A , A
  - B, B
  - C, C
  - D, D
- Mapper 2 output
  - A, A
  - E, E
  - F, F
  - C, C

- Reducer Input
  - A, [A, A]
  - B, B
  - C, [C, C]
  - D, D
  - E, E
  - F, F
- Reducer Output
  - A
  - B
  - C
  - D
  - E
  - F

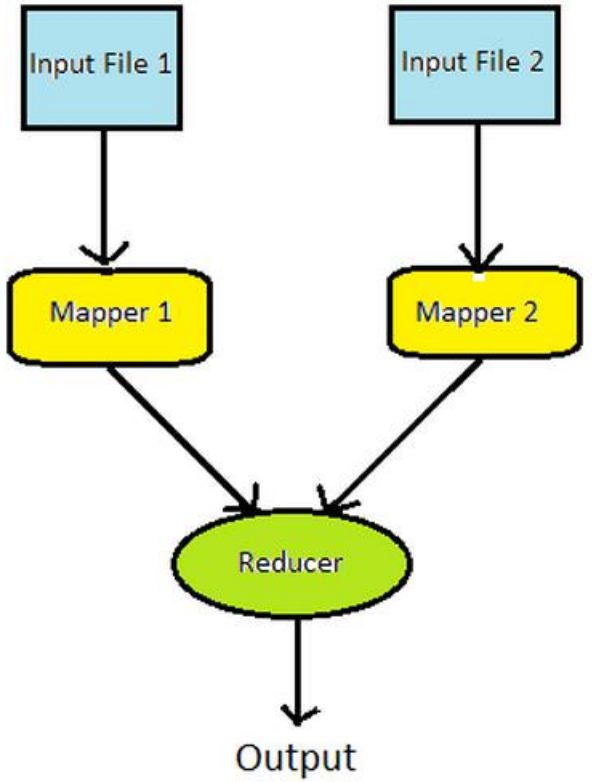
Intersection

- Compute $R \cap S$
- mapper 1
  - Read each row of table R
  - Output (t,t)
- mapper 2
  - Read each row of table S
  - Output (t,t)
- reducer
  - Output only duplicates
    - $(t,[t,t]) \rightarrow (t,t)$



Difference

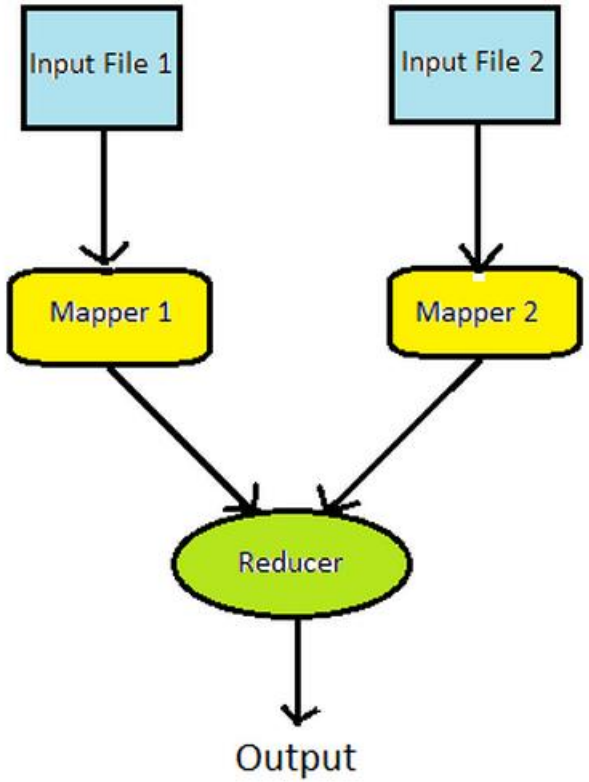
- Compute *R* – S
  - All rows in *R*, not in *S*
- mapper 1
  - Read each row of table *R*
  - Output (*t*,*R*)
- mapper 2
  - Read each row of table *S*
  - Output (*t*,*S*)
- reducer
  - Output only
    - (*t*,[R]) -> (*t*,*t*)



Join

BIG DATA
Natural Join

- Join R and S on attributes B
  - A, C are the other attributes in R,S
- mapper 1
  - Read (a,b) of R, output (b,(R,a))
- mapper 2
  - Read (b,c) of S, output (b,(S,c))
- reducer
  - for each pair b,(R,a)) and (b,(S,c)), output (a,b.c)



BIG DATA

Problem

- Given the following input for two files
  - Table Employee E(Name, age)
    - Gabbar 35
    - Viru 37
    - Jai 33
    - Baldev 44
    - Basanti 31
  - Table Dept D(Name, Dept)
    - Gabbar Bandit
    - Viru Hero
    - Jai Hero
    - Baldev Police
    - Basanti Heroine

- Show (for the Natural Join algorithm)
  - Input and output of mapper 1
  - Input and output of mapper 2
  - Input and output of reducer

Solution

- Mapper 1 Output
  - Gabbar, (E, 35)
  - Viru, (E, 37)
  - Jai, (E, 33)
  - Baldev, (E, 44)
  - Basanti, (E, 31)
- Mapper 2 Output
  - Gabbar, (D, Bandit)
  - Viru, (D, Hero)
  - Jai, (D, Hero)
  - Baldev (D, Police)
  - Basanti (D, Heroine

- Reducer Input
  - Gabbar, (E, 35), (D, Bandit)
  - Viru, (E, 37), (D, Hero)
  - Jai, (E, 33), (D, Hero)
  - Baldev, (E, 44), (D, Police)
  - Basanti, (E, 31), (D, Heroine)
- Reducer Output
  - Gabbar, 35, Bandit
  - Viru, 37, Hero
  - Jai, 33, Hero
  - Baldev, 44, Police
  - Basanti, 31, Heroine)
-

Grouping and Aggregation

BIG DATA
Grouping and Aggregation...

- For relation *R*(*A,B,C*) group by A and aggregate by function *f(*B)
  - Social networking site has a relationship *Friends* (*User, Friend, Date of friendship*)
  - Grouping by *User* and aggregating by *COUNT* (*Friend*) produces a table
    - First column is the *User*
    - Second column is the count of *Friends* for the *User*

...Grouping and Aggregation

- map
  - for each line (a,b,c) (e.g., User, Friend, Date)
  - Output (a,b) (e.g., User, Friend)
- reduce
  - Aggregate (a, [b1 , b2 , b3 , ...]) into (a, f(b1 , b2 , b3 , ...))

Case Study : HIVE

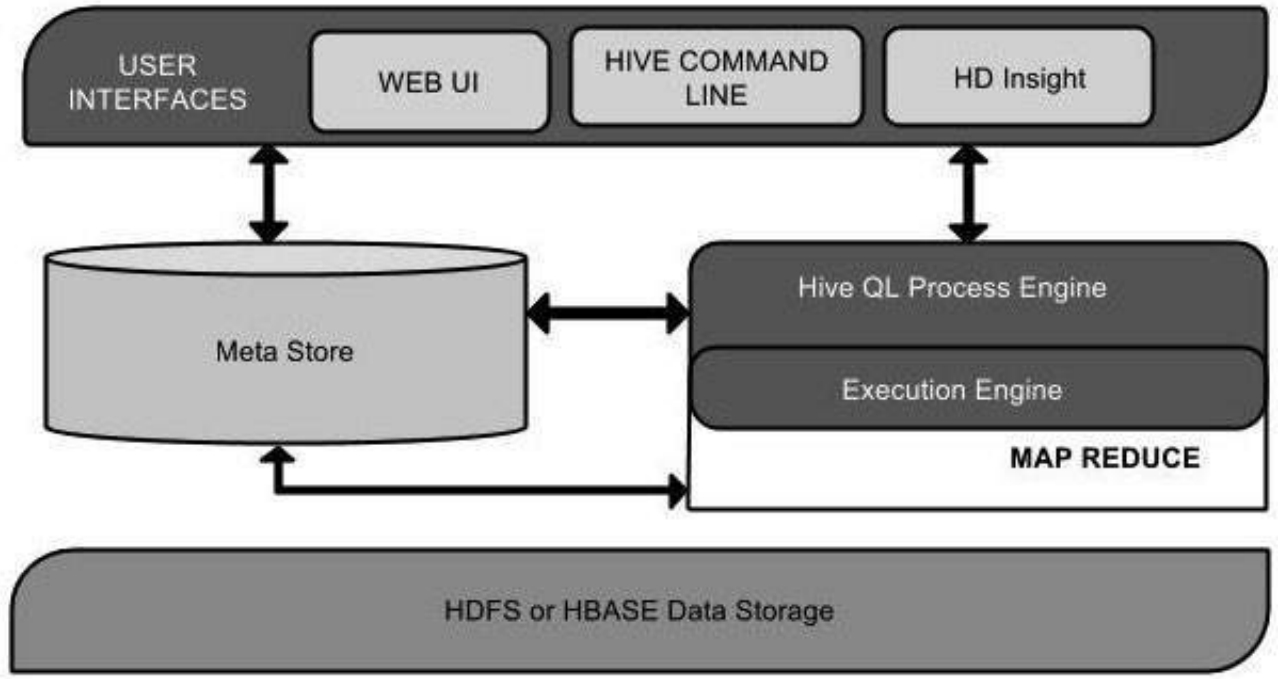
What is HIVE

- A system for querying and managing structured data built on top of Map/Reduce and Hadoop
- Facebook data
  - Structured logs with rich data types (structs, lists and maps)
  - A user base wanting to access this data in the language of their choice
  - A lot of traditional SQL workloads on this data (filters, joins and aggregations)
  - Other non SQL workloads

BIG DATA

What is HIVE

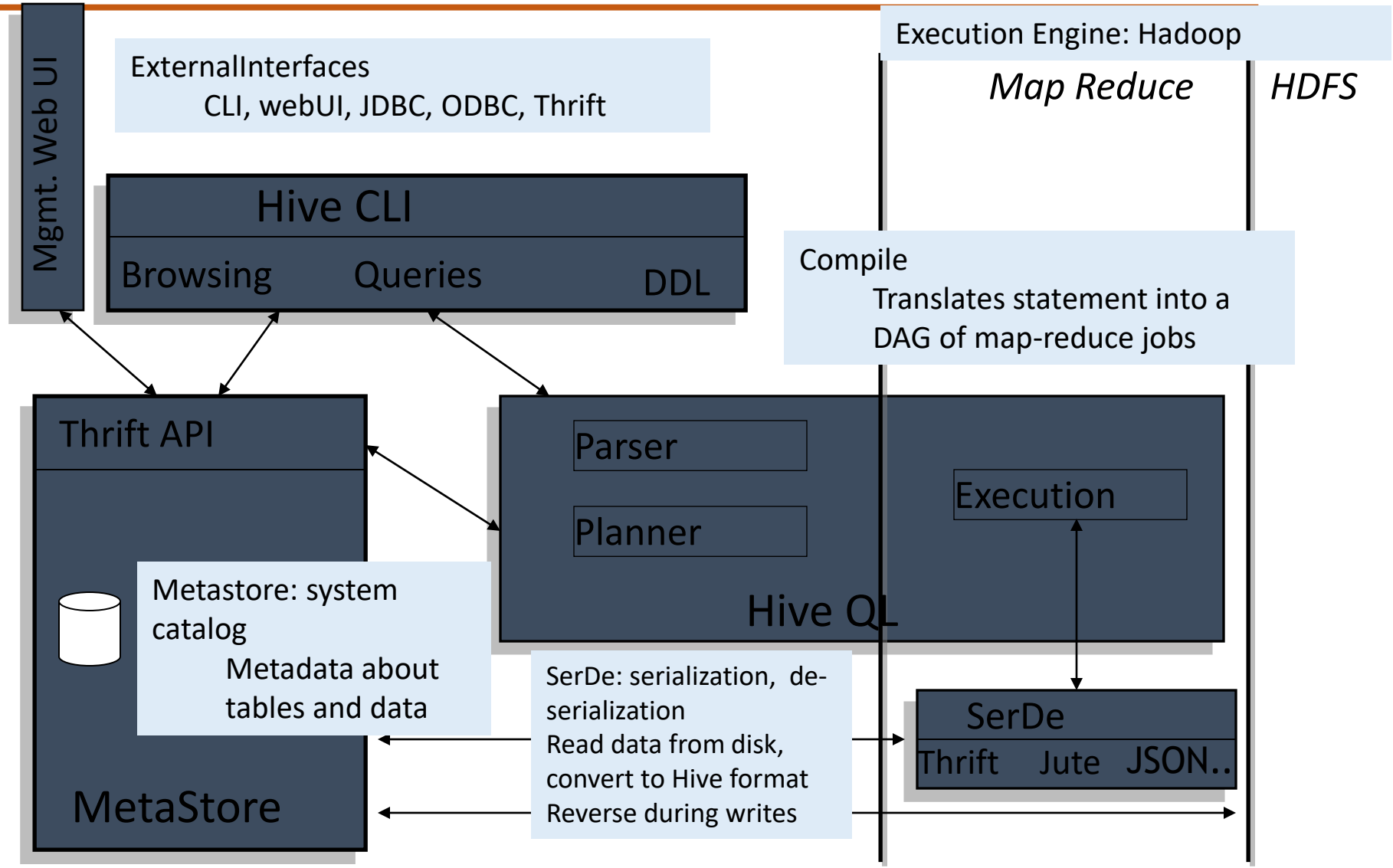
- Data is stored in HDFS
- Meta Store – for schema
- User submits SQL query
  - Converted to MR jobs



http://ewh.ieee.org/r6/scv/computer/nfic/2008/Facebook%20Hive%20by%20Ashish%20Thusoo.ppt

BIG DATA

HIVE Components



BIG DATA

Data Model

1. Hive data stored as HDFS files

3. Partition
   Part of table partitioned on values of columns
   Implemented as a n HDFS subdirectory
   If clicks is partitioned on column ds Data with a particular ds value 2008-03-25 will be stored in /hive/clicks/ds=2008-03-25
   If further divided on ctry value US will be stored in the directory /hive/clicks/ds=2008-03-25/ctry=US.

/hive/clicks
/hive/clicks/ds=2008-03-25
/hive/clicks/ds=2008-03-25/0

2. Table – similar to table in relational database
   Mapped to HDFS directory
   Data for table clicks is in the directory /hive/clicks
   For scalability, tables divided into multiple files and directories

4. Bucket
   Subdivision of partition
   Divided based on hash of a specified column

Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive: a warehousing solution over a map-reduce framework." *Proceedings of the VLDB Endowment* 2, no. 2 (2009): 1626-1629

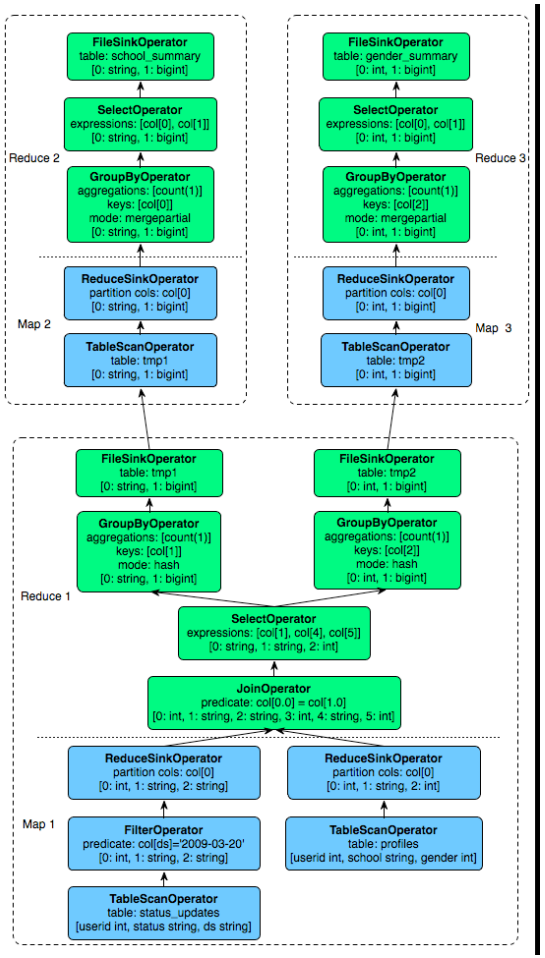
HIVE compilation example

```
FROM (SELECT a.status, b.school, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.userid = b.userid and
               a.ds='2009-03-20' )
      ) subq1
INSERT OVERWRITE TABLE gender_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.gender, COUNT(1) GROUP BY subq1.gender
INSERT OVERWRITE TABLE school_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```

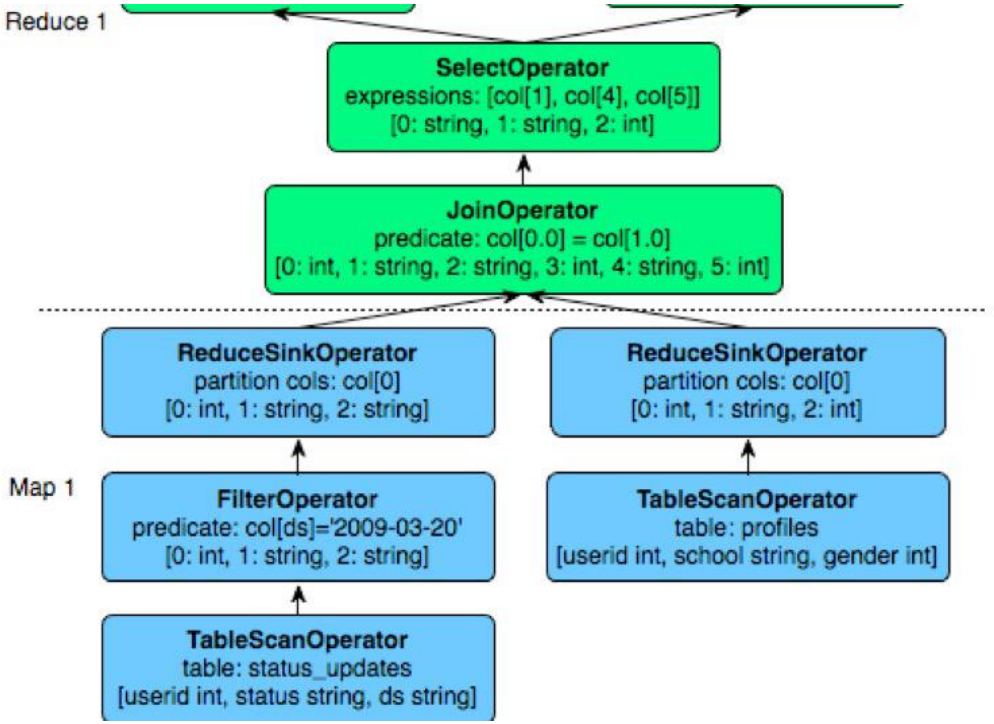
Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive: a warehousing solution over a map-reduce framework." *Proceedings of the VLDB Endowment* 2, no. 2 (2009): 1626-1629

BIG DATA

HIVE compilation example




```
FROM (SELECT a.status, b.school, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.userid = b.userid and
               a.ds='2009-03-20' )
      ) subq1
```



Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive: a warehousing solution over a map-reduce framework." *Proceedings of the VLDB Endowment* 2, no. 2 (2009): 1626-1629

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu

BIG DATA

Hands On Session - 2

HIVE

K V Subramaniam

Usha Devi B G

Dept of Computer Science and Engineering

BIG DATA

OVERVIEW

- HIVE is an open-source system for **querying and managing structured data** built on top of Hadoop.
- Hive supports queries expressed in a SQL-like declarative language.
- HiveQL, which are compiled into mapreduce jobs are executed using Hadoop.
- Metastore – A system catalog that contains schemas and statistics, which are useful in data exploration, query optimization and query compilation.

BIG DATA

Objective

- HIVE queries on a real world dataset.

BIG DATA

Problem Statement

- Find the frequency of books published each year from the data set.

"ISBN";"Book-Title";"Book-Author";"Year-Of-Publication";"Publisher";"Image-URL-S";"Image-URL-M";"Image-URL-L"

"0195153448";"Classical Mythology";"Mark P. O. Morford";"2002";"Oxford University Press";"http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0195153448.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg"

"0002005018";"Clara Callan";"Richard Bruce Wright";"2001";"HarperFlamingo Canada";"http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg"

BIG DATA

HIVE Hands-on

SPECIFICATIONS

1. Hadoop: 3.2
2. Java: 1.8
3. Hive : apache-hive-2.1.0
4. Dataset: Please download the dataset from the forum.

Step 1. To start the Hive Terminal:

a) Run,
   - *$ start-dfs.sh*
   - *$ start-yarn.sh* (Start hadoop)

b) $ *cd $HIVE_HOME*

c) Run Hive.

$ sudo bin/hive

OUTPUT Shell will look like

Logging initialized using configuration in jar:file:/usr/lib /hive/apache-hive-0.13.0-bin/lib/hive- common-0.13.0.jar!/ hive-log4j.properties
hive>

d) If hive command gives an error, try removing metastore_db
$ *rm -rf metastore_db* (It is present in the $HIVE_HOME)) directory or $HIVE_HOME/bin directory)
e) $ cd bin/
f) $ schematool -dbType derby -initSchema
g) Run hive again.

Step 2: To create a database

Syntax: create database <database name>;

Example: create database sample_database;

Step 3: To create a table

Syntax: create table <table name>(attribute_name_1 datatype, attribute_name_2 datatype) row format delimited fields terminated by '<delimiter type>';

Example: *create table sample_table(id INT, name string) row format delimited fields terminated by ' ';*

Step 4: To load data into table

Syntax: load data local inpath '<local absolute path to data.txt>' overwrite into table <table name>;

Example: *load data local inpath '/home/xyz/data.txt' overwrite into table sample_table;*

Step 5: Query the Hive Database.

Syntax*:* SELECT <attribute_name_1>, <attribute_name_2>
FROM <table_name > GROUP BY <attribute_name_2> ;

Problem Statement

- Find the number of cars in every city which use gas as a mode of fuel using Hive.

- **Columns of the Dataset :** The columns are indexed from [0-25] (Ex. Transmission is the 11th index)

- **Sample output :**

City	Number of Cars that use Gas
Bangalore	10
Chennai	12

- Actual output to be displayed as two columns on the terminal inside HIVE shell with each line of the answer having the pair <cityname> <number> .

THANK YOU

K V Subramaniam
Usha Devi B G

Department of Computer Science and Engineering

BIG DATA

Columnar Databases for Analytics

K V Subramaniam
Computer Science and Engineering

Hadoop Available Storage Types

HDFS and it's limitations

HDFS is good for batch processing (scans over big files)

Limitations

- Not good for record lookup
- Not good for incremental addition of small batches
- Not good for updates

BIG DATA

HIVE and it's limitations

- Doesn't store data
  - Uses HDFS or Hbase as actual store
- Provides an SQL Interface for querying data
- Data must be Structured: definite schema

Limitations

- Not good for record lookup
- Not good for incremental addition of small batches
- Not good for updates
- Not good for unstructured/semistructured data

BIG DATA

Hbase and Cassandra

- Built on the BigTable data model
- different in architecture

Advantages

- Fast record lookup
- Record level insertion
- Support for updates (Hbase creates new versions)
- Support for unstructured/semistructured data

Chang, Fay, et al. "Bigtable: A distributed storage system for structured data."*ACM Transactions on Computer Systems (TOCS)* 26.2 (2008): 4.

BIG DATA

Use case for different storage types

- HDFS
  - Unstructured data
  - Writes: no updates, only appends
  - Read entire file and analyze
- Hive
  - Structured data
  - Analytics via SQL
- HBase/Cassandra
  - Unstructured data
  - Arbitrary writes
  - Analytics

BIG DATA

Exercise

Which of these could be stored in HDFS, Hive or Hbase?

- Parsed transaction logs of user activity in a website where relevant fields from the log have been extracted
- Unparsed transaction logs of user activity
- Database of users and friends at a social website, which is periodically analyzed for social networking analysis

BIG DATA

Solution

Which of these could be stored in HDFS, Hive or Hbase?

- Parsed transaction logs of user activity in a website where relevant fields from the log have been extracted: **HIVE**
- Unparsed transaction logs of user activity : **HDFS**
- Database of users and friends at a social website, which is periodically analyzed for social networking analysis : **HBASE**

Columnar Storage

BIG DATA

Motivational Example – storage in DBMS

Row Key	Info:height	Info:age	School:House	School:sports
HarryPotter	4.5ft	11	Gryffindor	Quidditch
Voldemort	7ft	50	Slytherin	

- Row storage: DB is stored as a single file, one row per line
- Column storage: each column is a separate file, one value per line
- For using data, we need to perform an I/O to load data from disk
- Which method does less I/O for
  - Analyzing the relationship between age and earnings
    - Column storage
  - Adding a new row or read a row
    - Row storage

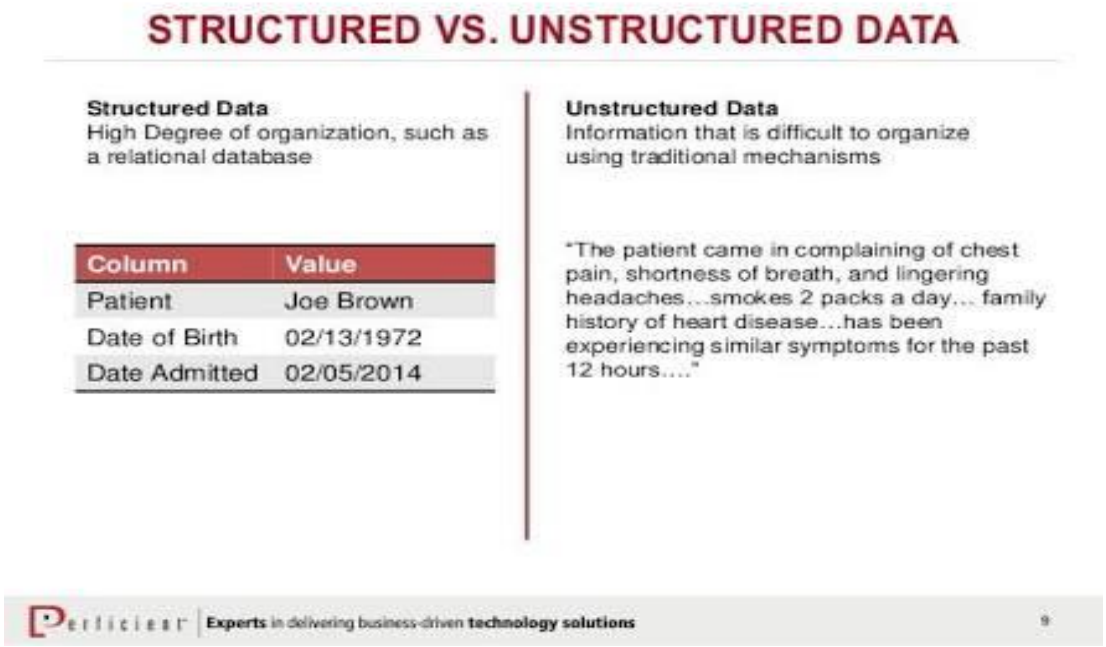
BIG DATA

History

- The first use of dbs were for transactions
  - Read a person's bank balance
  - Update bank balance
  - Row storage used since more efficient for transactions
- Column dbs
  - Became popular with Big Data systems
  - More efficient for analytics, particularly if db is large
  - To handle unstructured data

BIG DATA

Unstructured data



- How can the unstructured data above be stored in a structured relational db?
- How can it be stored in a unstructured db?

https://www.slideshare.net/perficientinc/ibm-watson-content-analytics-discover-hidden-value-in-your-unstructured-data

BIG DATA

Unstructured data

Customer id	Visit id	Date
		2-Oct 2017

Customer id	Visit id	Symptom id	Symptom
		1	Chest pain
		2	Headache

Structured db

Customer id	Visit id	Info
		Date: {2-Oct 2017} Symptoms: {Chest pain, Headache}

Unstructured db: simpler, more efficient

Hbase and Cassandra Data Model

BIG DATA

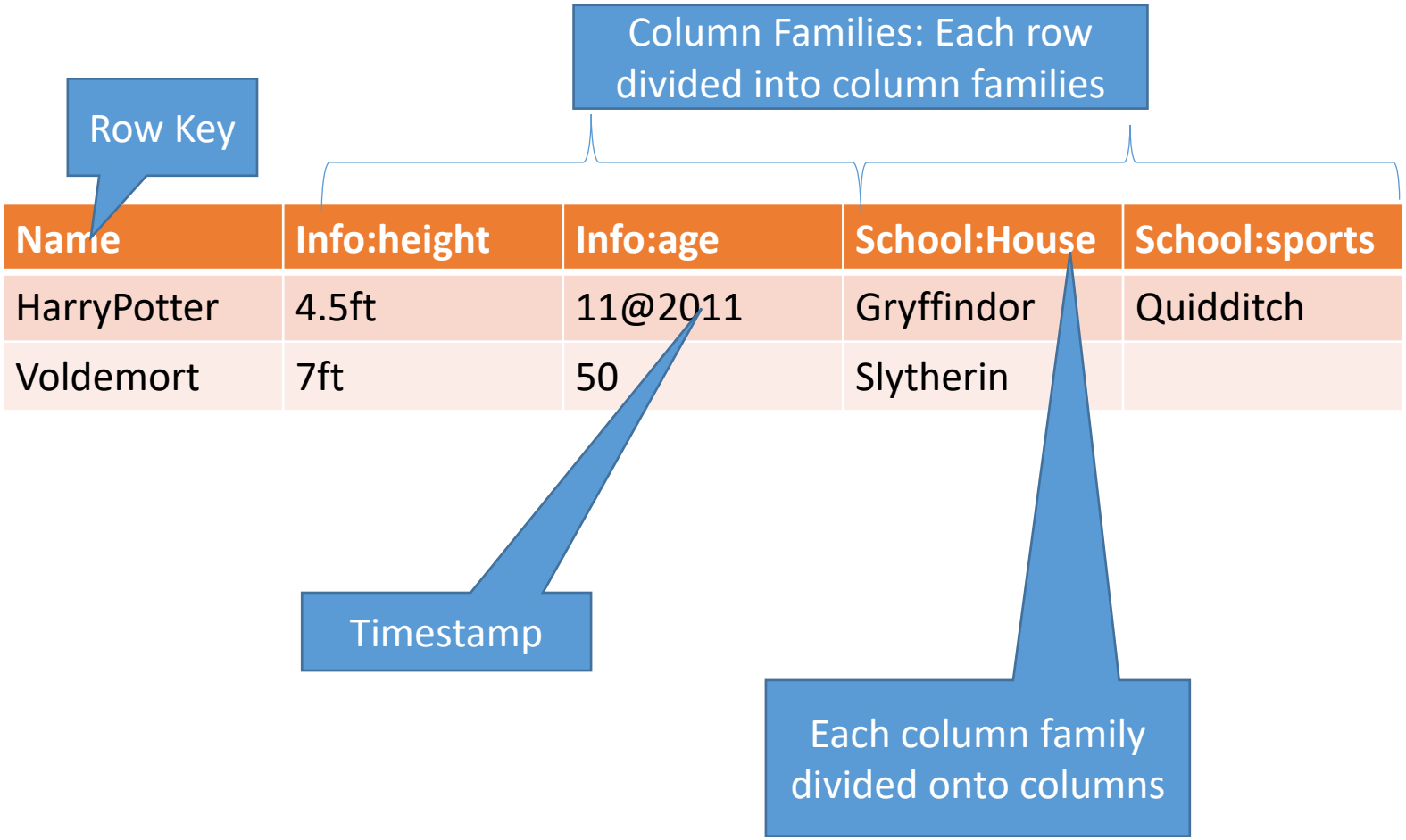
Hbase/Cassandra

- Hbase
  - Distributed column oriented database built on top of HDFS
  - Data is logically organized as rows/columns of a table
- Cassandra
  - Distributed database – peer to peer built by facebook
  - Inspired by Dynamo DB
  - Same data model as Hbase – inspired by BigTable

BIG DATA

Data Model – BigTable

- Key-Value pairs



BIG DATA

Column Families

HBase schema consists of several ***Tables***

Each table consists of a set of ***Column Families***

- Columns are not part of the schema

HBase has ***Dynamic Columns***

- Because column names are encoded inside the cells
- Different cells can have different columns

"School" column family has different columns in different cells

Name	Data
HarryPotter	Info:{height:"4.5ft", age: "11@2011"} School:{House:"Gryffindor", Sports:"Quidditch"}
Voldemort	Info:{height:"7ft", age: "50"} School:{House:"Sylitherin", Role:"Prefect"}

BIG DATA

Data Model – BigTable

Row Key	Data
HarryPotter	**Info**:{height:”4.5ft”, age: “11@2011”} **School**:{House:”Gryffindor”, Sports:”Quidditch”}
Voldemort	**Info**:{height:”7ft”, age: “50”} **School**:{House:”Sylytherin”, Role:”Prefect@1980, DarkLord@1995”}

Different types of data into different column families

Single column may have different values at different timestamps

BIG DATA

Data Model – BigTable

Column family named "anchor"

Column family named "Contents"

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>…"		
	t11	"<html>…"		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor:cnnsi.com"	"CNN"
	t13		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>…"		
	t5	"<html>…"		
	t3	"<html>…"		

Key

- Byte array
- Serves as the primary key for the table
- Indexed for fast lookup

Column Family

- Has a name (string)
- Contains one or more related columns

Column

- Belongs to one column family
- Included inside the row
  - ***familyName:columnName***

BIG DATA

Data Model – BigTable

Version Number

- Unique within each key
- By default→ System's timestamp
- Data type is Long

Value (Cell)

- Byte array (Hbase only)

Version number for each row

value

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>…"		
	t11	"<html>…"		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor:cnnsi.com"	"CNN"
	t13		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>…"		
	t5	"<html>…"		
	t3	"<html>…"		

Hbase Architecture

Hbase Architecture – Master Slave

Region

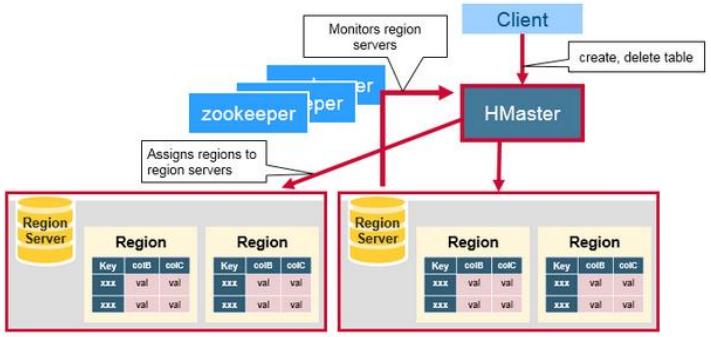
- A subset of a table's rows, like horizontal range partitioning
- Automatically done

RegionServer (many slaves)

- Manages data regions
- Serves data for reads and writes (***using a log***)
- Like datanode of HDFS

Master

- Responsible for coordinating the slaves
- Assigns regions, detects failures
- Admin functions
- Line namenode of HDFS



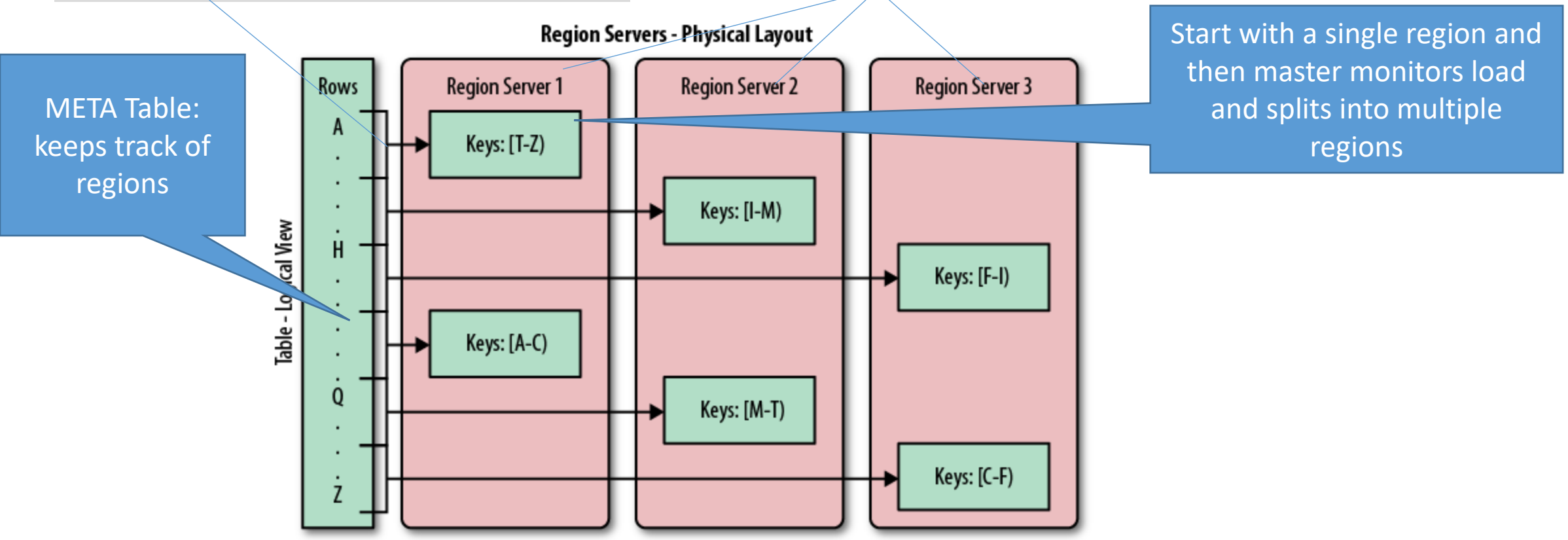
Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

BIG DATA

Regions and region servers

Tables horizontally partitioned into regions

Regions stored on region servers



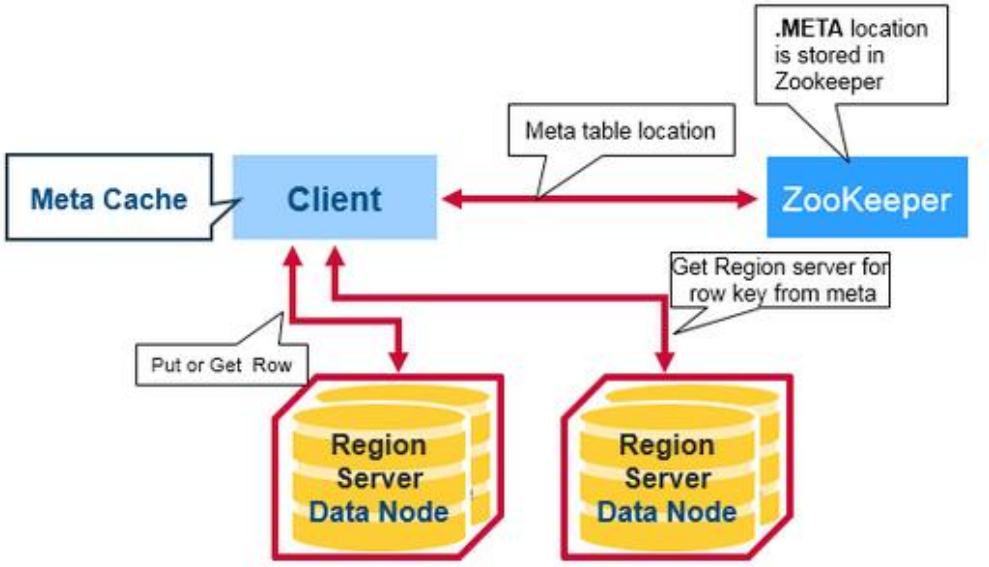
META Table: keeps track of regions

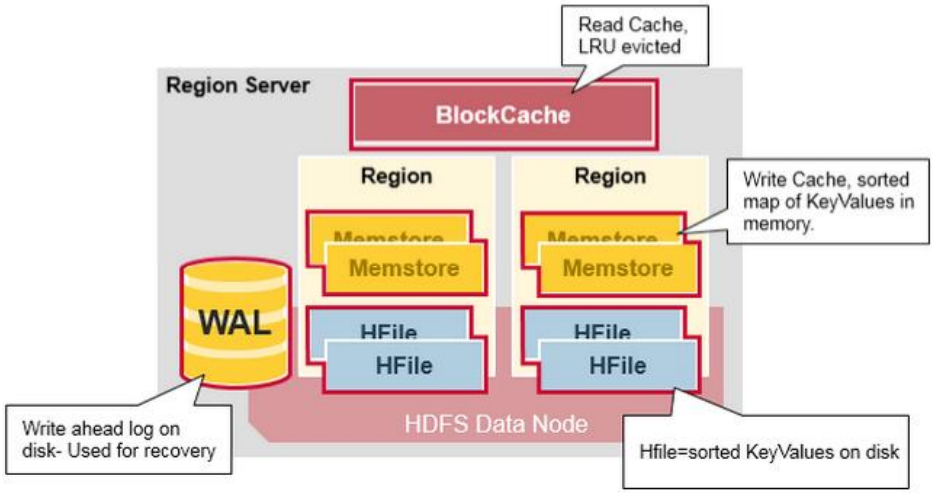
Start with a single region and then master monitors load and splits into multiple regions

Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

BIG DATA

Read/Write Operation





Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

Cassandra Architecture

Cassandra Architecture – peer to peer architecture

- Differences from Hbase
  - Request coordination over a partitioned dataset – no Master
  - Ring membership and failure detection – no Master
  - Local persistence (storage) engine – does not rely on HDFS
- Cqlsh – for performing queries

Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

Hbase usage

Hbase: Creating a new table

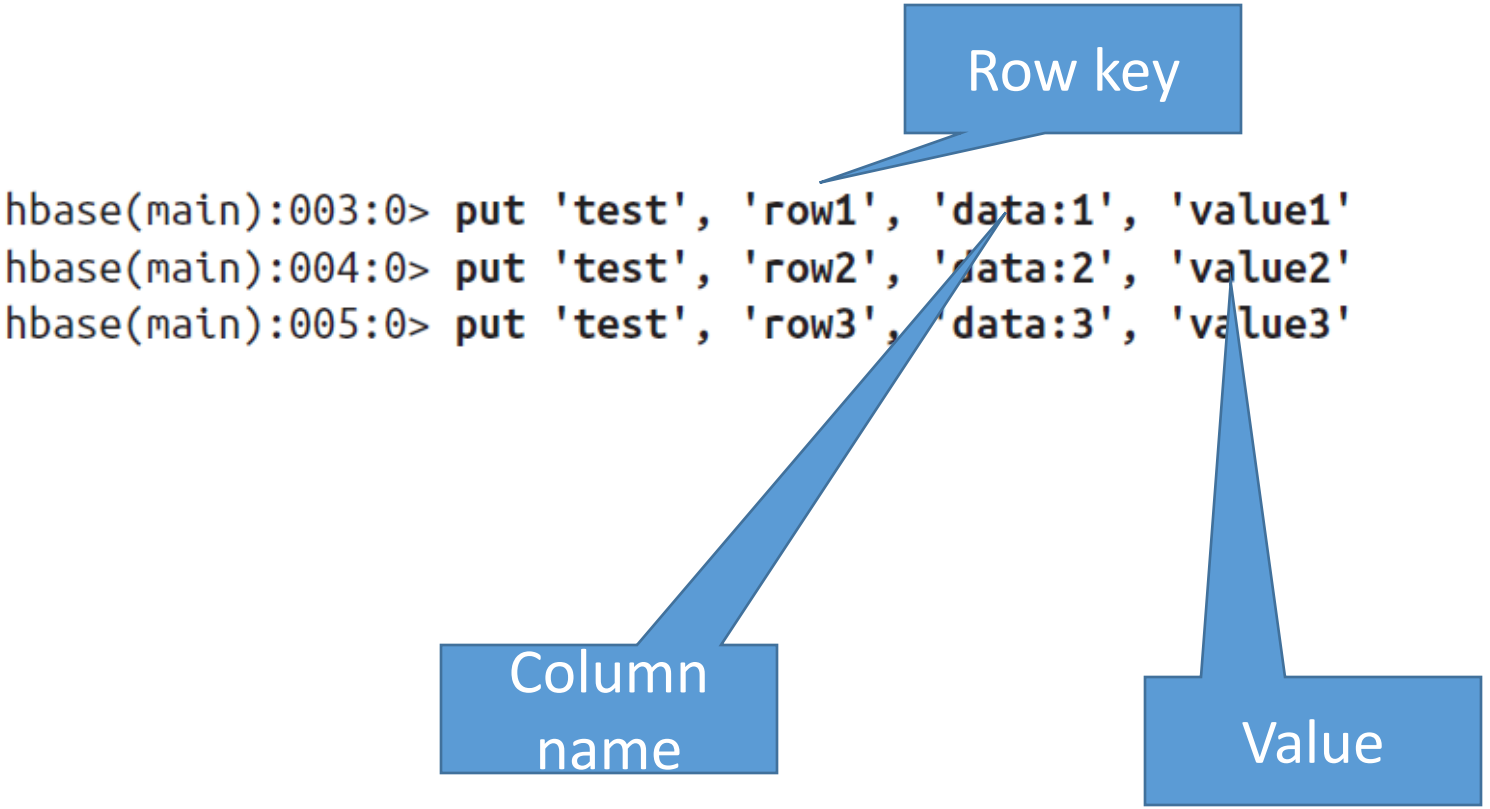
```
hbase(main):001:0> create 'test', 'data'
0 row(s) in 0.9810 seconds
```

Column family name

Table name

BIG DATA

Hbase inserting data



BIG DATA

Hbase: retrieving data

Getting a specific row

```
hbase(main):006:0> get 'test', 'row1'
COLUMN                          CELL
 data:1                         timestamp=1414927084811, value=value1
1 row(s) in 0.0240 seconds
hbase(main):007:0> scan 'test'
ROW                             COLUMN+CELL
 row1                           column=data:1, timestamp=1414927084811, value=value1
```

All rows

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu