# Apache Pig

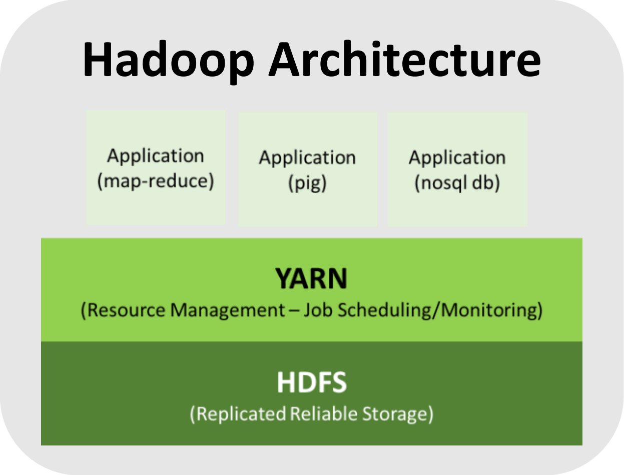
https://pig.apache.org/

**Venkatesh Vinayakarao**
venkateshv@cmi.ac.in
http://vvtesh.co.in

Chennai Mathematical Institute

Making Pig Fly – Thejas Nair.

# Recap



## Hadoop Architecture

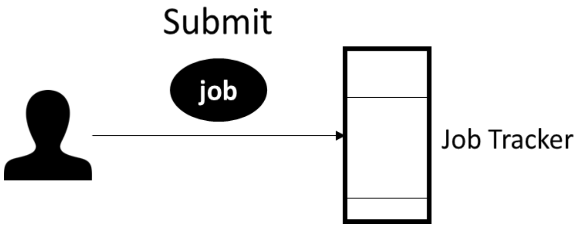
Application (map-reduce)

Application (pig)

Application (nosql db)

**YARN**
(Resource Management – Job Scheduling/Monitoring)

**HDFS**
(Replicated Reliable Storage)

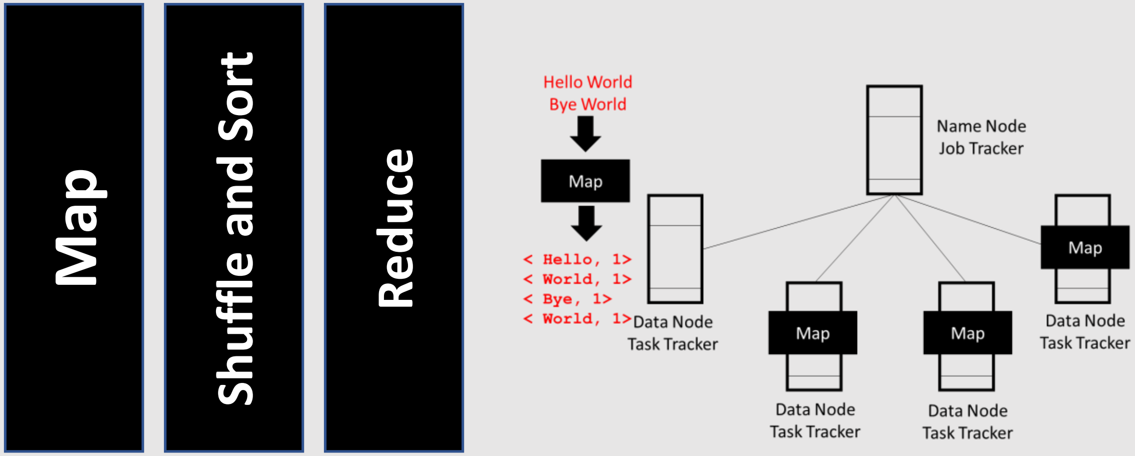


Submit

job

Job Tracker

## Map-Reduce Model

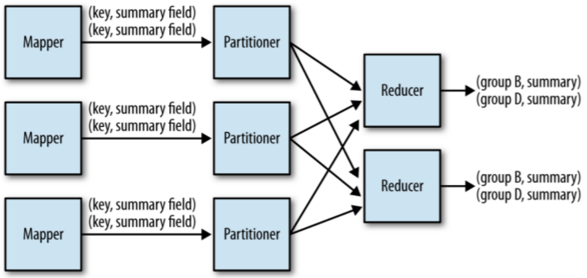


**Map**

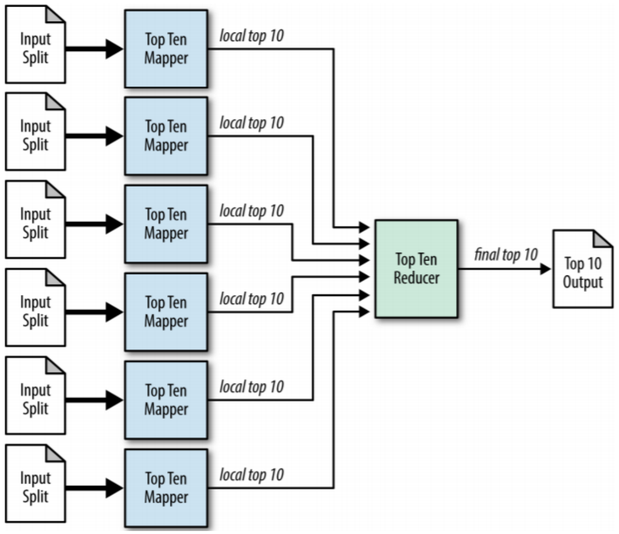
**Shuffle and Sort**

**Reduce**

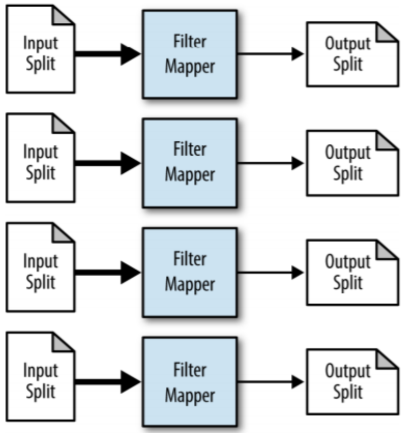
# Map-Reduce Patterns


Mapper
Mapper
Mapper
(key, summary field)
(key, summary field)
Partitioner
Partitioner
Partitioner
Reducer
Reducer
(group B, summary)
(group D, summary)

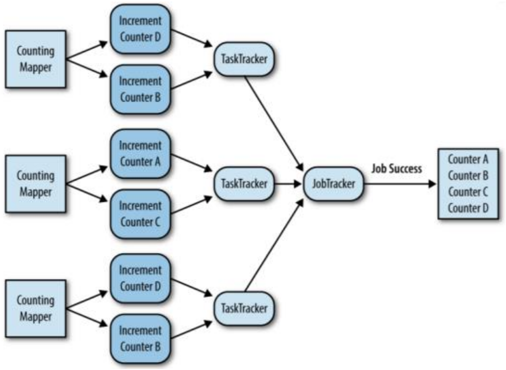
**Summarization**


Input Split
Top Ten Mapper
local top 10
Top Ten Reducer
final top 10
Top 10 Output

**Top 10**

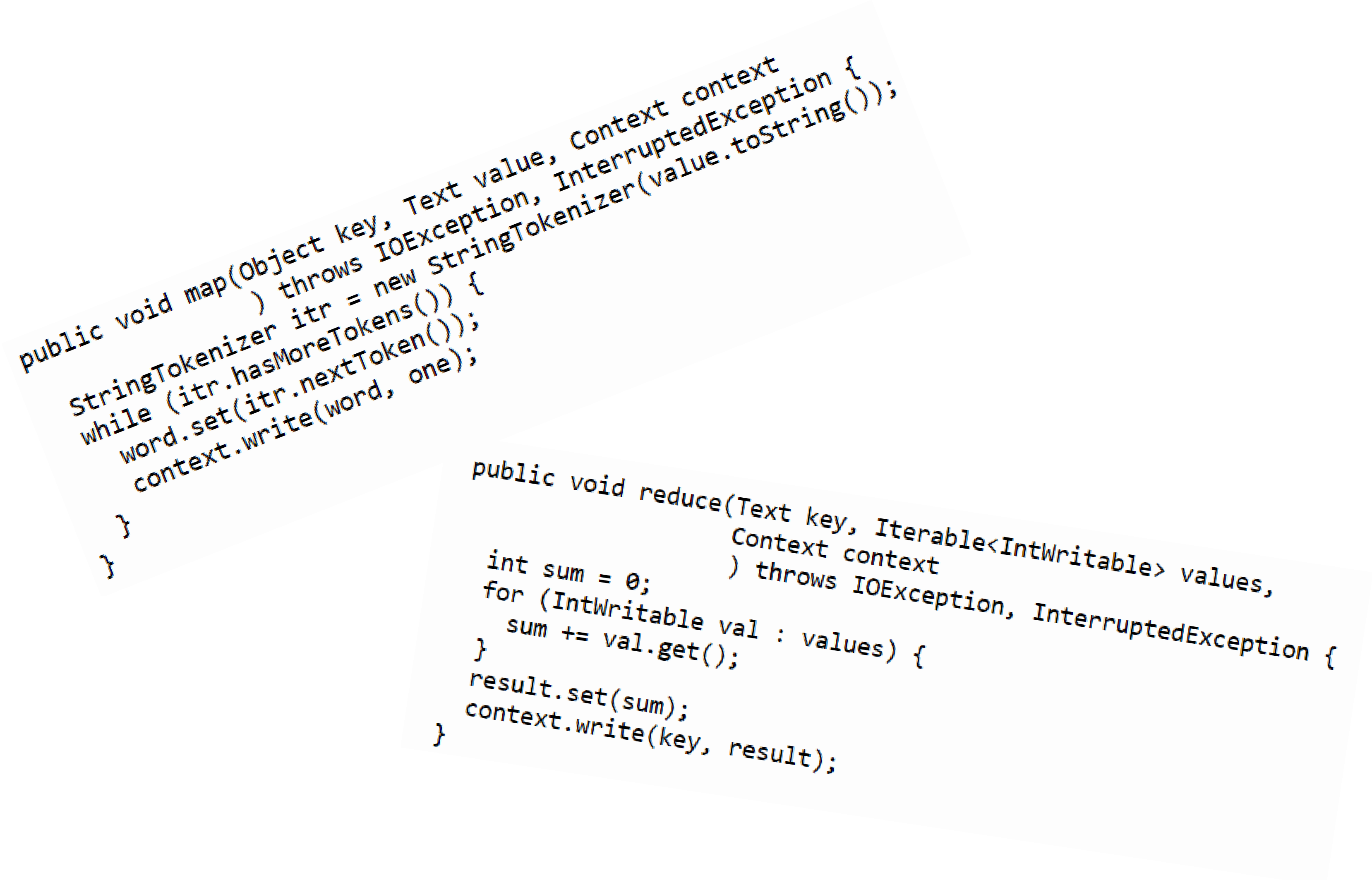

Input Split
Filter Mapper
Output Split

**Filtering**


Counting Mapper
Increment Counter D
Increment Counter B
Increment Counter A
Increment Counter C
TaskTracker
JobTracker
Job Success
Counter A
Counter B
Counter C
Counter D

**Counting**

# Code



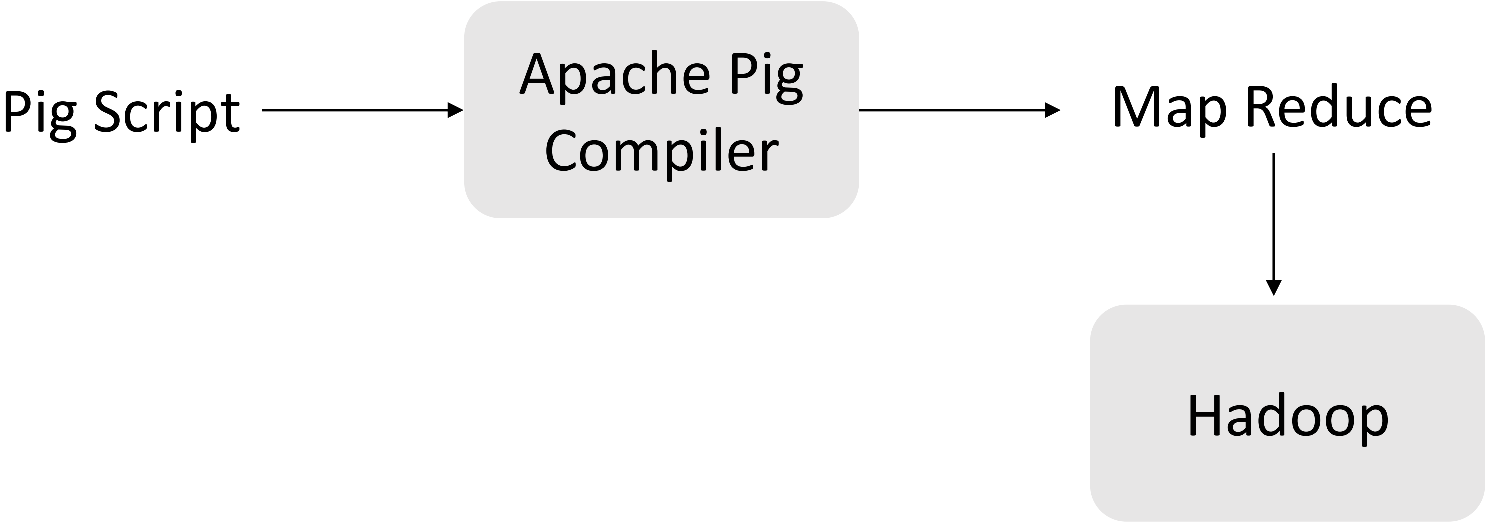
```
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
  StringTokenizer itr = new StringTokenizer(value.toString());
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
  }
}
```

```
public void reduce(Text key, Iterable<IntWritable> values,
                   Context context
                   ) throws IOException, InterruptedException {
  int sum = 0;
  for (IntWritable val : values) {
    sum += val.get();
  }
  result.set(sum);
  context.write(key, result);
}
```

# But…

What if…
We are not good at coding?

# Scripting instead of Coding



# A Sample Pig Script

LOAD Command Syntax

```
LOAD 'data' [USING function] [AS schema];
```

```
A = LOAD 'student' USING PigStorage()
                    AS (name:chararray, age:int, gpa:float);
B = FOREACH A GENERATE name;
DUMP B;
```

Read: https://pig.apache.org/docs/r0.16.0/basic.html#load

# Benefits & Limitations

- Benefits
  - 10 lines of Pig Latin (approx.) = 200 lines in Java
  - 15 minutes in Pig Latin (approx.) = 3 hours in Java
    - Simple
    - Easy
    - Quick to Code
  - Provides in-built functions to load, process and print data.
  - Similar to SQL
    - Can perform join and order by
- Limitations
  - Slower than Map-Reduce

# Pig in Real-World

- Yahoo uses it extensively (>70% of jobs)
- Facebook – Process Logs
- Twitter – Process Logs
- eBay – Data processing for intelligence
- …

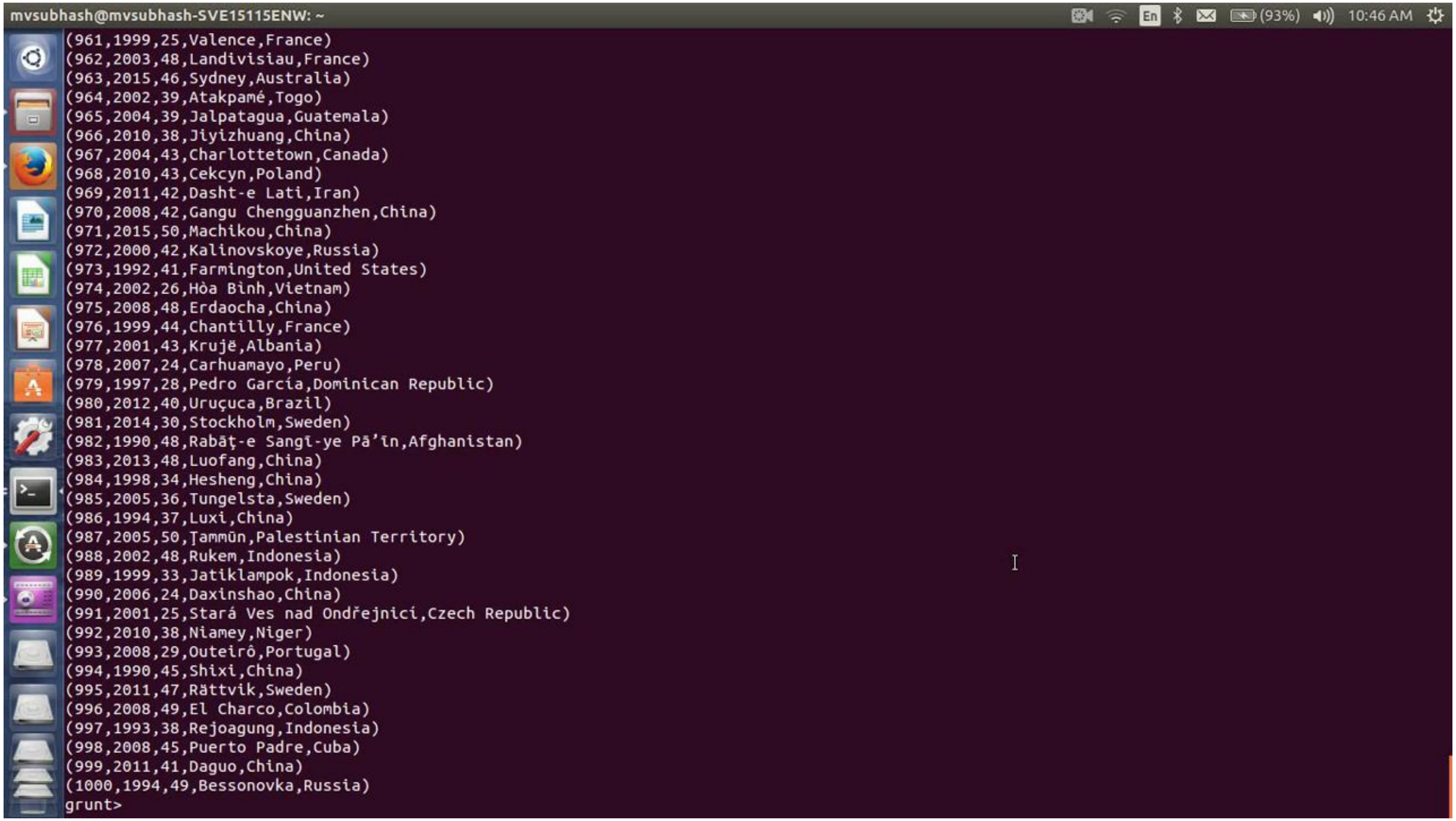
# Grunt Shell

```
$ pig -x local
... - Connecting to ...
grunt>
```

Or

```
pig –x local id.pig
```

# Tutorial



```
(961,1999,25,Valence,France)
(962,2003,48,Landivisiau,France)
(963,2015,46,Sydney,Australia)
(964,2002,39,Atakpamé,Togo)
(965,2004,39,Jalpatagua,Guatemala)
(966,2010,38,Jiyizhuang,China)
(967,2004,43,Charlottetown,Canada)
(968,2010,43,Cekcyn,Poland)
(969,2011,42,Dasht-e Lati,Iran)
(970,2008,42,Gangu Chengguanzhen,China)
(971,2015,50,Machikou,China)
(972,2000,42,Kalinovskoye,Russia)
(973,1992,41,Farmington,United States)
(974,2002,26,Hòa Bình,Vietnam)
(975,2008,48,Erdaocha,China)
(976,1999,44,Chantilly,France)
(977,2001,43,Krujë,Albania)
(978,2007,24,Carhuamayo,Peru)
(979,1997,28,Pedro García,Dominican Republic)
(980,2012,40,Uruçuca,Brazil)
(981,2014,30,Stockholm,Sweden)
(982,1990,48,Rabāţ-e Sangī-ye Pā’īn,Afghanistan)
(983,2013,48,Luofang,China)
(984,1998,34,Hesheng,China)
(985,2005,36,Tungelsta,Sweden)
(986,1994,37,Luxi,China)
(987,2005,50,Ţammūn,Palestinian Territory)
(988,2002,48,Rukem,Indonesia)
(989,1999,33,Jatiklampok,Indonesia)
(990,2006,24,Daxinshao,China)
(991,2001,25,Stará Ves nad Ondřejnicí,Czech Republic)
(992,2010,38,Niamey,Niger)
(993,2008,29,Outeirô,Portugal)
(994,1990,45,Shixi,China)
(995,2011,47,Rättvik,Sweden)
(996,2008,49,El Charco,Colombia)
(997,1993,38,Rejoagung,Indonesia)
(998,2008,45,Puerto Padre,Cuba)
(999,2011,41,Daguo,China)
(1000,1994,49,Bessonovka,Russia)
grunt>
```

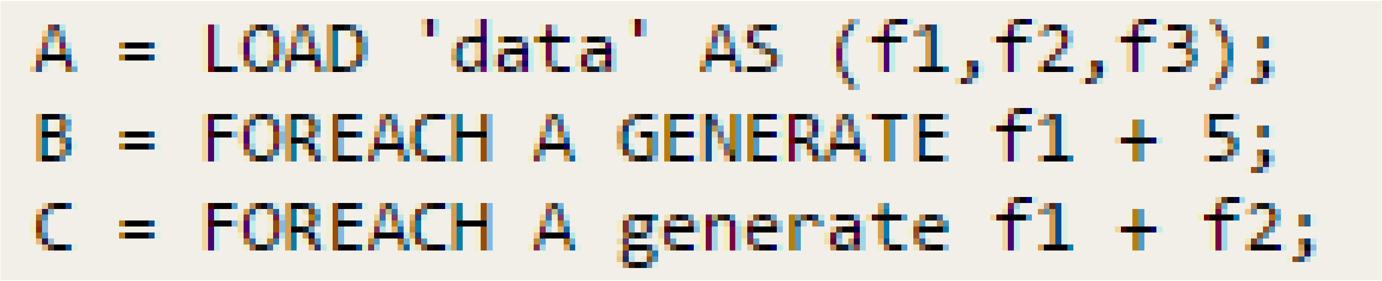
# Pig Philosophy

- Pigs eat anything
  - Input can be of a variety of formats
- Pigs live anywhere
  - Not only for hadoop
- Pigs are domestic animals
  - Easy to master
- Pigs fly
  - Ultimately map-reduce code. Improving performance is a priority to the pig team.

# Welcome to the World of Pig

- Pig Latin
    - For the language
- Grunt
    - For the shell
- Piggy-bank
    - For the shared reusable modules

# More Examples



```
A = LOAD 'data' AS (f1,f2,f3);
B = FOREACH A GENERATE f1 + 5;
C = FOREACH A generate f1 + f2;
```

# Referencing Fields

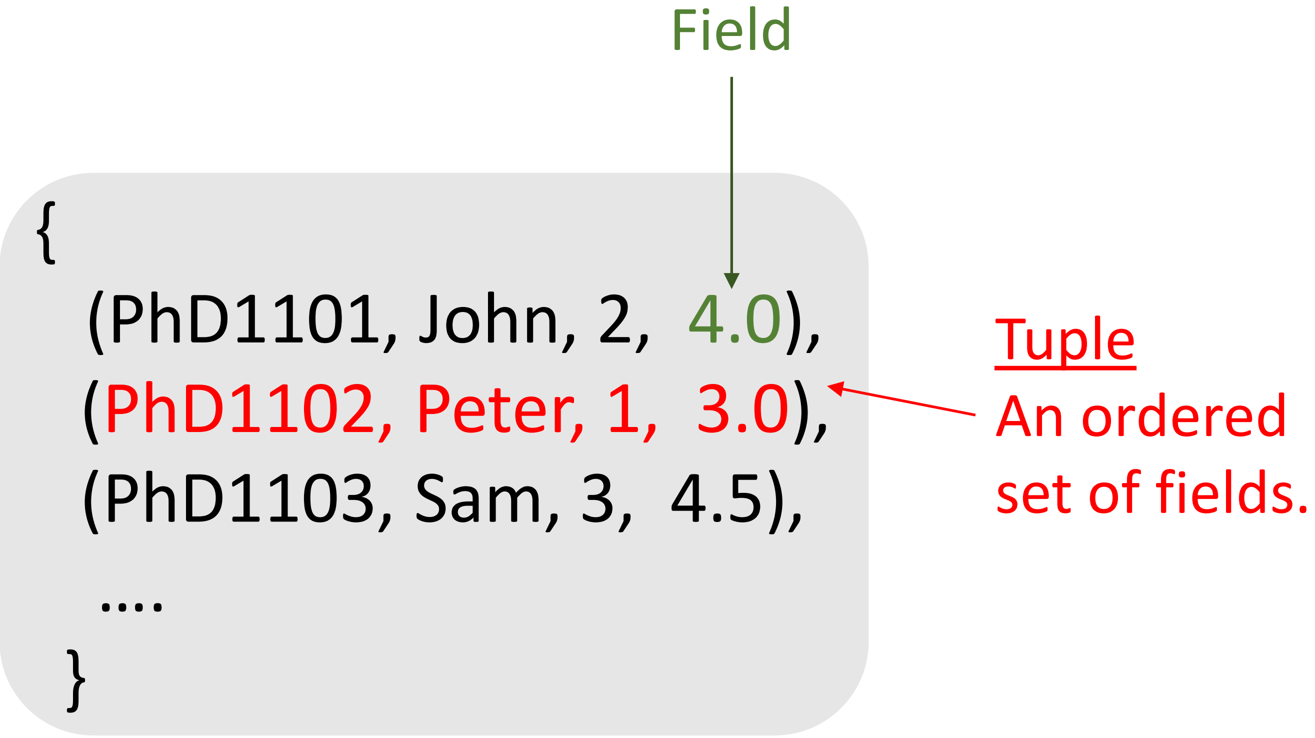
```
A = LOAD 'student' USING PigStorage() AS
          (name:chararray, age:int, gpa:float);
X = FOREACH A GENERATE name,$2;
DUMP X;
```

```
(John,4.0F)
(Mary,3.8F)
(Bill,3.9F)
(Joe,3.8F)
```

# Data Types

- Scalar Types:
  - Int, long, float, double, boolean, null, chararray, bytearray;
- Complex Types:
  - Field, Tuple and Relation/Bag
  - Map [key#value]

# Data Types in Pig Latin



Field

```
{
  (PhD1101, John, 2,  4.0),
  (PhD1102, Peter, 1,  3.0),
  (PhD1103, Sam, 3,  4.5),
   ....
 }
```

Tuple

An ordered set of fields.

Relation/Bag

An ordered set of tuples.

# Load and Dump

```
A = LOAD 'data' AS (f1:int,f2:int,f3:int);
DUMP A;

(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
```

**Input**

```
(3,8,9) (4,5,6)
(1,4,7) (3,7,5)
(2,5,8) (9,5,8)
```

```
A = LOAD 'data' AS (
        t1:tuple(t1a:int, t1b:int,t1c:int),
        t2:tuple(t2a:int,t2b:int,t2c:int)
        );
DUMP A;
```

**Output**

```
((3,8,9),(4,5,6))
((1,4,7),(3,7,5))
((2,5,8),(9,5,8))
```

**Guess the output**

```
X = FOREACH A GENERATE
t1.t1a,t2.$0;
DUMP X;
```

# The Answer

```
X = FOREACH A GENERATE t1.t1a,t2.$0;
DUMP X;
```

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

# Tuples

```
A = LOAD 'data' as (f1:int,
          f2:tuple(t1:int,t2:int,t3:int));
DUMP A;
```

```
(1,(1,2,3))
(2,(4,5,6))
(3,(7,8,9))
(4,(1,4,7))
(5,(2,5,8))
```

# Map

Data

```
328;ADMIN HEARNG;[street#939 W El Camino,city#Chicago,state#IL]
43;ANIMAL CONTRL;[street#415 N Mary Ave,city#Chicago,state#IL]
```

Usage

```
grunt> departments = LOAD ‘somefile’
      AS (dept_id:int, dept_name:chararray, address:map[]);

grunt> dept_addr = FOREACH departments
      GENERATE dept_name,
                        address#'street' as street,
                        address#'city' as city,
                        address#'state' as state;
```

https://www.hadoopinrealworld.com/beginners-apache-pig-tutorial-map/

# Operations

- Loading data
  - LOAD loads input data
  - Lines=LOAD ‘input/access.log’ AS (line: chararray);
- Projection
  - FOREACH … GENERTE … (similar to SELECT)
  - takes a set of expressions and applies them to every record.
- Grouping
  - GROUP collects together records with the same key
- Dump/Store
  - DUMP displays results to screen, STORE save results to file system
- Aggregation
  - AVG, COUNT, MAX, MIN, SUM

# Example

- students = LOAD 'student.txt’ USING PigStorage('\t’) AS (studentid: int, name:chararray, age:int, gpa:double);
- studentid = FOREACH students GENERATE studentid, name;

# Filter

**Data:**

```
year,product,quantity
-----------------------
2000, iphone, 1000
2001, iphone, 1500
2002, iphone, 2000
```

```
grunt> A = LOAD '/user/hadoop/sales' USING PigStorage(',')
AS (year:int,product:chararray,quantity:int);
grunt> B = FILTER A BY quantity >= 1500;
grunt> DUMP B;
```

# How to run Pig Scripts?

- Local mode
  - Local host and local file system is used
  - Neither Hadoop nor HDFS is required
  - Useful for prototyping and debugging
- MapReduce mode
  - Run on a Hadoop cluster and HDFS
- Batch mode - run a script directly
  - Pig –x local my_pig_script.pig
  - Pig –x mapreduce my_pig_script.pig
- Interactive mode  use the Pig shell to run script
  - Grunt> Lines = LOAD '/input/input.txt' AS (line:chararray);
  - Grunt> Unique = DISTINCT Lines;
  - Grunt> DUMP Unique;

# Flatten

Let the Input -> (a,(b,c)) be in A.

B = foreach A generate $0 , flatten ($1)

Output -> (a,b,c)

# Tokenize

- Input
  - 001,Raj Reddy,21,Hyderabad
  - 002,Raj Chatterjee,22,Kolkata
  - 003,Raj Khanna,22,Delhi

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt' USING
PigStorage(',') as (id:int, name:chararray, age:int,  city:chararray);

grunt> student_name_tokenize = foreach student_details  Generate
TOKENIZE(name);

grunt> Dump student_name_tokenize;
```

# Output

```
({(Raj),(Reddy)})
({(Raj),(Chatterjee)})
({(Raj),(Khanna)})
```

Splits a string. Creates tuples of names. Outputs the bag.

# Store

```
STORE student INTO ' hdfs://localhost:9000/pig_Output/ ' USING PigStorage (',');
```

You can write your own functions! In this class, we will use the built-in PigStorage.

# Word Count

```
Lines=LOAD ‘input/hadoop.log’ AS (line: chararray);
Words = FOREACH Lines GENERATE
            FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE  group,
            COUNT(Words);
Results = ORDER Words BY Counts DESC;
Top5 = LIMIT Results 5;
STORE Top5 INTO /output/top5words;
```

# User Defined Functions

- What is UDF
  - Way to do an operation on a field or fields
  - Called from within a pig script
  - Currently all done in Java
- Why use UDF
  - You need to do more than grouping or filtering
  - Maybe more comfortable in Java land than in SQL/Pig Latin

# UDF in Pig

```
-- myscript.pig
REGISTER myudfs.jar;
A = LOAD 'student_data' AS (name: chararray, age: int, gpa: float);
B = FOREACH A GENERATE myudfs.UPPER(name);
DUMP B;
```

# Simple UDF

```
public class UPPER extends EvalFunc<String>   {
    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();
      } catch(Exception e)  {
         throw new IOException("Caught exception", e);
      }
 }  }
```

Source: https://pig.apache.org/docs/r0.10.0/udf.html

# Creating the Jar

jar -cf exampleudf.jar exampleudf

Know where have you placed this jar.

In Pig Script:

- REGISTER ‘…path to jar’;
- DEFINE SIMPLEUPPER exampleudf.UPPER();
- … now you can use this method.

https://pig.apache.org/docs/latest/basic.html#define-udfs

# Thank You!

Appendix: Exam and Presentations

# Presentation

- A good presentation
  - Has a nice flow.
    - Motivation – History – Context/Domain – Key Ideas – Demo – Summary.
  - Uses original content and original examples.
  - Sets a strong agenda, and faithfully meets it.
  - Explains any technical terms introduced.
  - Tests student understanding.
  - Occupies 45 - 55 mins + 10 - 15 mins for Q & A.
  - Starts on-time.
  - Includes demos if applicable.
  - Keep it engaging and thought provoking.
  - Refers to additional content, books, wikis, etc.

I do not evaluate on how much you know. I evaluate the presentation based on how much it helped the audience in learning something new, important and interesting.

# Mid-Term Exam

- 90 Minutes
- 40 Marks (Weighted down to 20% overall)
  - 10 x 2-Mark Multiple-Choice Questions (+2 for right answer and -1 for wrong answer).
  - 4 * 3-Mark Questions
  - 2 * 4-Mark Questions

**Expected Median Score = 24/40 (**= 12% Overall Weight**)**