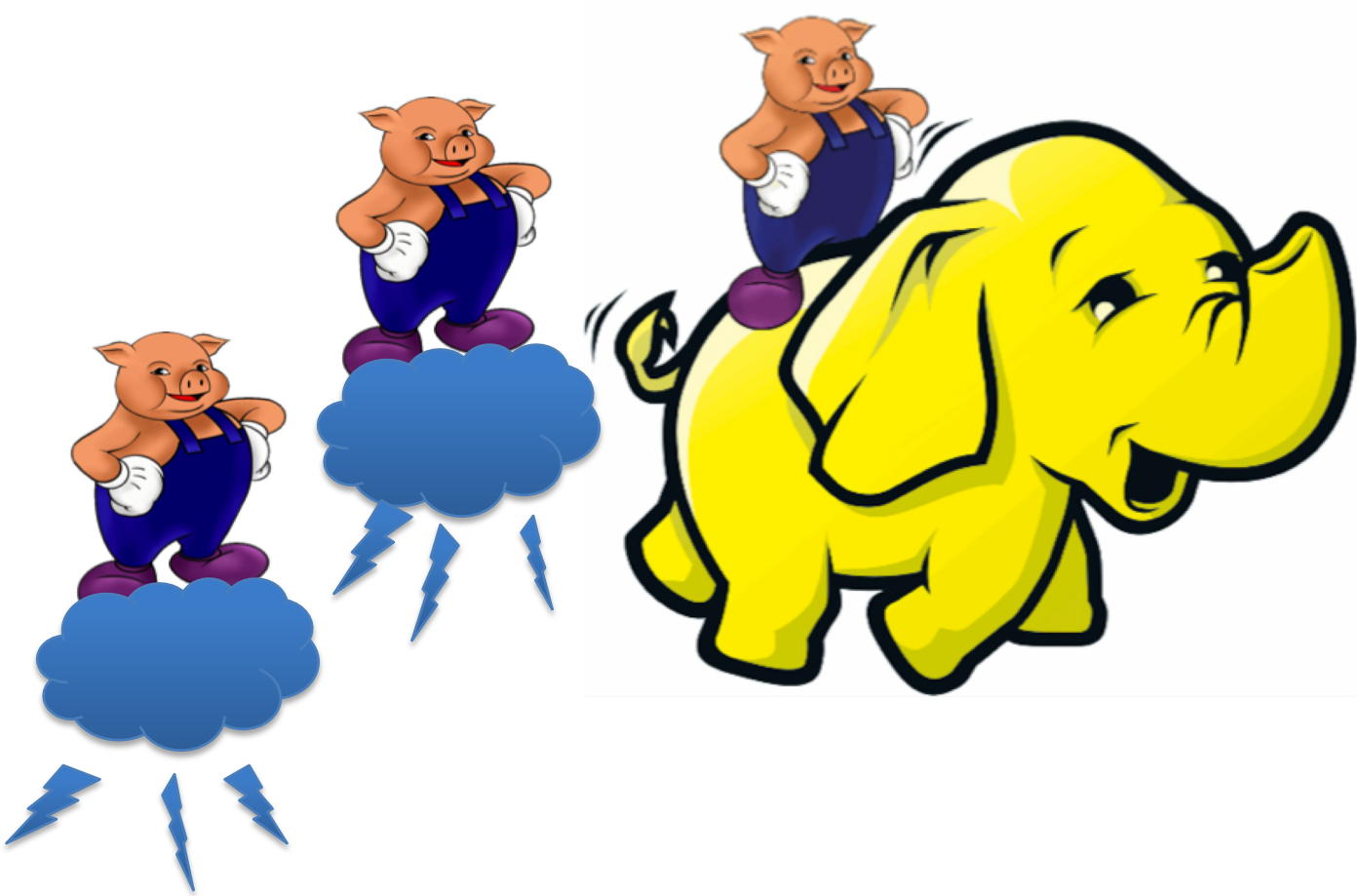
Pig Functions



Pig Functions

- The built-in functions.
- User defined functions (UDFs).

Pig Built-in Functions

- Pig built-in functions can be used directly. These include load, store, filter and many other functions that we already saw.
- Two main properties differentiate built-in functions from user defined functions (UDFs).
  - First, built-in functions don't need to be **registered** because Pig knows where they are.
  - Second, built-in functions don't need to be **qualified** when they are used because Pig knows where to find them and what they are.
- For a list of these functions, see a file, named *Pig_Built-in_Functions.pdf*, on the course website.

Some Example of Pig Built-in Functions

The AVG() Function

- The dataset: students GPA data of different terms.

```
John Smith, fall 2000, 3.9
John Smith, winter 2000, 3.8
John Smith, spring 2001, 4.0
John Smith, summer 2001, 3.9
Mary Clark, fall 2000, 3.5
Mary Clark, winter 2000, 3.0
Mary Clark, spring 2001, 3.8
Mary Clark, summer 2001, 3.9
```

- Calculate the average GPA of each student.

The AVG() Function (cont'd)

Title: avg.pig

Pig script: PIG helper

```
1 A = LOAD '/student_gpa.txt'using PigStorage(',')
2     AS (name:chararray, term:chararray, gpa:float);
3 dump A; --check if the data is read in correctly
4
5 B = GROUP A BY name; --group the data by student name
6 DUMP B; --check the result
7
8 C = FOREACH B GENERATE group, AVG($1.gpa);
9 DUMP C;
```

The Result

The Job job_1429376865754_0068 has been started successfully.

You can always go back to Query History for results after the run.

A

```
(John Smith, fall 2000,3.9)
(John Smith, winter 2000,3.8)
(John Smith, spring 2001,4.0)
(John Smith, smmer 2001,3.9)
(Mary Clark, fall 2000,3.5)
(Mary Clark, winter 2000,3.0)
(Mary Clark, spring 2001,3.8)
(Mary Clark, smmer 2001,3.9)
```

B

```
(John Smith,{(John Smith, smmer 2001,3.9),(John Smith, spring 200
1,4.0),(John Smith, winter 2000,3.8),(John Smith, fall 2000,3.9)}
)
(Mary Clark,{(Mary Clark, smmer 2001,3.9),(Mary Clark, spring 200
1,3.8),(Mary Clark, winter 2000,3.0),(Mary Clark, fall 2000,3.5)}
)
```

C

```
(John Smith,3.900000035762787)
(Mary Clark,3.550000011920929)
```

The ROUND_TO() Function

- Returns the value of an expression rounded to a fixed number of decimal digits. Note that this function is different from the **ROUND()** function, which returns the value of an expression rounded to an integer.
- The syntax:

ROUND_TO(val, digits)

val: an expression whose result is type float or double: the value to round.

digits: an expression whose result is type int: the number of digits to preserve.

mode: an optional int specifying the rounding mode, according to the constants Java provides.

The ROUND_TO() Function (cont'd)

Title: avg_round_

Pig script: PIG helper

```
1 A = LOAD '/student_gpa.txt'using PigStorage(',')
2     AS (name:chararray, term:chararray, gpa:float);
3 dump A; --check if the data is read in correctly
4
5 B = GROUP A BY name; --group the data by student name
6 DUMP B; --check the result
7
8 C = FOREACH B GENERATE group, ROUND_TO(AVG($1.gpa),1);
9 DUMP C;
```

The ROUND_TO() Function: the Result

A

```
(John Smith, fall 2000,3.9)
(John Smith, winter 2000,3.8)
(John Smith, spring 2001,4.0)
(John Smith, smmer 2001,3.9)
(Mary Clark, fall 2000,3.5)
(Mary Clark, winter 2000,3.0)
(Mary Clark, spring 2001,3.8)
(Mary Clark, smmer 2001,3.9)
```

B

```
(John Smith,{(John Smith, smmer 2001,3.9),(John Smith, spring 200
1,4.0),(John Smith, winter 2000,3.8),(John Smith, fall 2000,3.9)}
)
(Mary Clark,{(Mary Clark, smmer 2001,3.9),(Mary Clark, spring 200
1,3.8),(Mary Clark, winter 2000,3.0),(Mary Clark, fall 2000,3.5)}
)
```

C

```
(John Smith,3.9)
(Mary Clark,3.6)
```

The MAX() and MIN() Functions

- Computes the maximum or minimum of the numeric values or chararrays in a single-column bag. MAX() or MIN() requires a preceding **GROUP ALL** statement for global maximums or minimums and a **GROUP BY** statement for group maximums or minimums.

Title: max_min.pi

Pig script: PIG helper

```
1 A = LOAD '/student_gpa.txt'using PigStorage(',')
2     AS (name:chararray, term:chararray, gpa:float);
3 dump A; --check if the data is read in correctly
4
5 B = GROUP A BY name; --group the data by student name
6 DUMP B; --check the result
7
8 C = FOREACH B GENERATE group, MAX($1.gpa), MIN($1.gpa);
9 DUMP C;
```

The MAX() and MIN() Function: the Result

A

```
(John Smith, fall 2000,3.9)
(John Smith, winter 2000,3.8)
(John Smith, spring 2001,4.0)
(John Smith, smmer 2001,3.9)
(Mary Clark, fall 2000,3.5)
(Mary Clark, winter 2000,3.0)
(Mary Clark, spring 2001,3.8)
(Mary Clark, smmer 2001,3.9)
```

B

```
(John Smith,{(John Smith, smmer 2001,3.9),(John Smith, spring 200
1,4.0),(John Smith, winter 2000,3.8),(John Smith, fall 2000,3.9)}
)
(Mary Clark,{(Mary Clark, smmer 2001,3.9),(Mary Clark, spring 200
1,3.8),(Mary Clark, winter 2000,3.0),(Mary Clark, fall 2000,3.5)}
)
```

C

```
(John Smith,4.0,3.8)
(Mary Clark,3.9,3.0)
```

MAX MIN

The SUBTRACT() Functions

- **SUBTRACT()** takes two bags as arguments and returns a new bag composed of the tuples of first bag are that not in the second bag.
- The implementation assumes that both bags being passed to the SUBTRACT() function will fit **entirely into memory** simultaneously; if this is not the case, it will still function but will be **very** slow.
- Find out the bag elements that are in the first bag but not in the second bag:

```
({(8,9),(0,1),(1,2)},{(8,9),(1,1)})
({(2,3),(4,5)},{(2,3),(4,5)})
({(3,7),(3,7)},{(2,2),(3,7)})
({(1,2),(3,4),(5,6),(7,8)},{(2,3),(1,2)})
```

The SUBTRACT() Functions

PIG helper ▾ UDF helper ▾

```
1 A = LOAD '/two_bags.txt' USING PigStorage(',')
2     AS (B1:bag{T1:(t1:int,t2:int)},
3         B2:bag{T2:(f1:int,f2:int)});
4 DUMP A;
5 DESCRIBE A;
6 X=FOREACH A GENERATE SUBTRACT(B1,B2);
7 DUMP X;
8
```

The SUBTRACT() Function: the Result

A

```
({(8,9),(0,1),(1,2)},{(8,9),(1,1)})
({(2,3),(4,5)},{(2,3),(4,5)})
({(3,7),(3,7)},{(2,2),(3,7)})
({(1,2),(3,4),(5,6),(7,8)},{(2,3),(1,2)})
A: {B1: {T1: (t1: int,t2: int)},B2: {T2: (f1: int,f2: int)}}
```

X

```
({(1,2),(0,1)})
({})
({})
({(5,6),(3,4),(7,8)})
```

The ENDSWITH() and STARTSWITH() Functions

- Tests inputs to determine if the first argument ends or starts with the string in the second argument. Returns true or false
- Syntax:

  ENDSWITH(string, testAgainst)

  STARTSWITH(string, testAgainst)
- Examples:

  ENDSWITH ('foobar', 'foo') → false

  ENDSWITH ('foobar', 'bar') → true

  STARTSWITH ('foobar', 'foo') → true

  STARTSWITH ('foobar', 'bar') → false

LTRIM(), RTRIM() and TRIM() Functions

- **LTRIM()**: Returns a copy of a string with only leading white space(s) removed.
- **RTRIM()**: Returns a copy of a string with only trailing white space(s) removed.
- **TRIM()**: Returns a copy of a string with leading and trailing white space(s) removed.

The SUBSTRING() Function

- Returns a substring from a given string.
- Syntax:

  SUBSTRING(string, startIndex, stopIndex)

  string: the string from which a substring will be extracted.

  startIndex: the index (type int) of the first character of the substring.

  stopIndex: the index (type int) of the character *following* the last character of the substring.

- Example:

  SUBSTRING("Cornell", 0, 4) → "Corn"

The TOTUPLE() Function

- Converts one or more expressions to type tuple.
- Syntax: TOTUPLE(expression [, expression ...])
- Example:

```
a = LOAD 'students' AS (name:chararray, age:int, gpa:float);
DUMP a;
```

(John,18,4.0)
(Mary,19,3.8)
(Bill,20,3.9)
(Joe,18,3.8)

```
b = FOREACH a GENERATE TOTUPLE(name, age, gpa);
DUMP b;
```

((John,18,4.0))
((Mary,19,3.8))
((Bill,20,3.9))
((Joe,18,3.8))

The TOBAG() Function

- Converts one or more expressions to type bag.
- Syntax: TOBAG(expression [, expression ...])
- Example:

```
a = LOAD 'students' AS (name:chararray, age:int, gpa:float);
DUMP a;
```

(John,18,4.0)
(Mary,19,3.8)
(Bill,20,3.9)
(Joe,18,3.8)

```
b = FOREACH a GENERATE TOBAG(name, gpa);
DUMP b;
```

({(John),(4.0)})
({(Mary),(3.8)})
({(Bill),(3.9)})
({(Joe),(3.8)})

The TOMAP() Function

- Converts key/value expression pairs into a map.
- Syntax: TOMAP(key-expression, value-expression
  [, key-expression, value-expression ...])
- Example:

```
a = LOAD 'students' AS (name:chararray, age:int, gpa:float);
        (John,18,4.0)
        (Mary,19,3.8)
        (Bill,20,3.9)
        (Joe,18,3.8)

b = FOREACH a GENERATE TOMAP(name, gpa);
DUMP b;
        ([John#4.0])
        ([Mary#3.8])
        ([Bill#3.9])
        ([Joe#3.8])
```

Pig UDFs

Pig UDFs

- Pig makes it easy for the users to add their *own processing logic* via User Defined Functions (UDFs). These are written in Java or Python. As a result, the processing capability of Pig can be significantly extended.
- Pig and Hadoop are implemented in Java, so Java is a natural choice for UDFs as well; however, you are not forced into Java. Also, writing UDFs in Java is often a heavyweight process.
- From Pig version 0.8 or later, UDFs can also be written in Python. Python UDFs consist of a single function that is used in place of the *exec* method of a Java function by compiling down to the JVM (Java virtual machine).

Python UDFs

- Pig uses Jython to execute Python UDFs, so they must be compatible with Python 2.5x and 2.7x but cannot use Python 3 features.
- The benefit is that Python UDFs can be compiled to Java bytecode and run with relatively little performance penalty.
- The good news is that it is easier to write UDF's using Python than writing them in Java.

Registering Python UDFs

- When you use your own UDF, you have to tell Pig where to look for that UDF. This is done via the **register** command.
- The register command is used to locate resources for Python UDFs that you use in your Pig Latin scripts. In this case you register a Python script that contains your UDF. The Python script must be in *your current directory*. For simplicity, in this class you may put all these, including your data file, at the system root.
- Your Python script is a normal text file (here, a CentOS file, which can be easily composed with the **vi** editor, or a python file written in your MacOS/Windows).
- Your data file must be loaded into the HDFS file system (using *hadoop fs -copyFromLocal …,* or Ambari's Files View).

Count the Number of Characters of Each Line of the poem.txt File with a Python UDF in Pig

```
There is Another Sky

Emily Dickinson

There is another sky,
Ever serene and fair,
And there is another sunshine,
Though it be darkness there;
Never mind faded forests, Austin,
Never mind silent fields -
Here is a little forest,
Whose leaf is ever green;
Here is a brighter garden,
Where not a frost has been;
In its unfading flowers
I hear the bright bee hum:
Prithee, my brother,
Into my garden come!
```

The Python UDF

- Write the Python UDF as a Python script (pyudf0.py) in the vi editor.
- You need to expose this function as a UDF with a Python **decorator** by adding the **@outputSchema** decorator. This specifies a schema for the return value.

```
@outputSchema("length:int")
def get_length(data):
    length= len(data)
    return length
```

The Pig Script to Use the Python UDF

Title: myudf0.pig

Pig script: PIG helper

```
1 REGISTER '/pyudf0.py' USING jython as myudf;
2
3 A = LOAD '/poem.txt' as (aline:chararray);
4 B = filter A by aline is not null;
5 C = FOREACH B GENERATE aline, myudf.get_length(aline);
6
7 DUMP C;
```

The Result

The Job job_1429376865754_0114 has been started successfully.
You can always go back to Query History for results after the run.

```
(There is Another Sky,20)
(Emily Dickinson,15)
(There is another sky,,21)
(Ever serene and fair,,21)
(And there is another sunshine,,30)
(Though it be darkness there;,28)
(Never mind faded forests, Austin,,33)
(Never mind silent fields -,26)
(Here is a little forest,,24)
(Whose leaf is ever green;,25)
(Here is a brighter garden,,26)
(Where not a frost has been;,27)
(In its unfading flowers,23)
(I hear the bright bee hum:,26)
(Prithee, my brother,,20)
(Into my garden come!,20)
```

An Improved UDF pyudf1.py

- This version counts both the number of characters and the number of words for each line of the text.
- Add "# of character = " and "# of words = " before the respective values.

```
@outputSchema("nums:chararray")
def get_length(data):
    words = data.split()
    num_chars = len(data)
    num_words = len(words)
    nums = '# of characters = ' + str(num_chars), \
    ' # of words = ' + str(num_words)
    return nums
```

The New Pig Script to Use the Python UDF

Title: myudf1.pig

Pig script: PIG helper

```
1 REGISTER '/pyudf1.py' USING jython as myudf;
2
3 A = LOAD '/poem.txt' as (aline:chararray);
4 B = filter A by aline is not null;
5 C = FOREACH B GENERATE aline, myudf.get_length(aline);
6
7 DUMP C;
```

The Result

The Job job_1429376865754_0118 has been started successfully.
You can always go back to Query History for results after the run.

```
(There is Another Sky,(# of characters = 20, # of words = 4))
(Emily Dickinson,(# of characters = 15, # of words = 2))
(There is another sky,,(# of characters = 21, # of words = 4))
(Ever serene and fair,,(# of characters = 21, # of words = 4))
(And there is another sunshine,,(# of characters = 30, # of words = 5))
(Though it be darkness there;,(# of characters = 28, # of words = 5))
(Never mind faded forests, Austin,,(# of characters = 33, # of words = 5))
(Never mind silent fields -,(# of characters = 26, # of words = 5))
(Here is a little forest,,(# of characters = 24, # of words = 5))
(Whose leaf is ever green;,(# of characters = 25, # of words = 5))
(Here is a brighter garden,,(# of characters = 26, # of words = 5))
(Where not a frost has been;,(# of characters = 27, # of words = 6))
(In its unfading flowers,(# of characters = 23, # of words = 4))
(I hear the bright bee hum:,(# of characters = 26, # of words = 6))
(Prithee, my brother,,(# of characters = 20, # of words = 3))
(Into my garden come!,(# of characters = 20, # of words = 4))
```

Apply the Same UDF to a Different Data File

Title: mypyudf2.pig

Pig script: PIG helper

```
1 REGISTER '/pyudf1.py' USING jython as myudf;
2
3 A = LOAD '/ss.txt' as (aline:chararray);
4 B = filter A by aline is not null;
5 C = FOREACH B GENERATE aline, myudf.get_length(aline);
6
7 DUMP C;
```

The Result

```
(THE SONNETS,(# of characters = 11, # of words = 2))
(by William Shakespeare,(# of characters = 22, # of words = 3))
(                    1,(# of characters = 22, # of words = 1))
(  From fairest creatures we desire increase,,(# of characters = 44, # of words = 6))
(  That thereby beauty's rose might never die,,(# of characters = 45, # of words = 7))
(  But as the riper should by time decease,,(# of characters = 42, # of words = 8))
(  His tender heir might bear his memory:,(# of characters = 40, # of words = 7))
(  But thou contracted to thine own bright eyes,,(# of characters = 47, # of words = 8))
(  Feed'st thy light's flame with self-substantial fuel,,(# of characters = 55, # of words = 7))
(  Making a famine where abundance lies,,(# of characters = 39, # of words = 6))
(  Thy self thy foe, to thy sweet self too cruel:,(# of characters = 48, # of words = 10))
(  Thou that art now the world's fresh ornament,,(# of characters = 47, # of words = 8))
(  And only herald to the gaudy spring,,(# of characters = 38, # of words = 7))
(  Within thine own bud buriest thy content,,(# of characters = 43, # of words = 7))
(  And tender churl mak'st waste in niggarding:,(# of characters = 46, # of words = 7))
(    Pity the world, or else this glutton be,,(# of characters = 44, # of words = 8))
(    To eat the world's due, by the grave and thee.,(# of characters = 50, # of words = 10))
(                    2,(# of characters = 22, # of words = 1))
(  When forty winters shall besiege thy brow,,(# of characters = 44, # of words = 7))
(  And dig deep trenches in thy beauty's field,,(# of characters = 46, # of words = 8))
(  Thy youth's proud livery so gazed on now,,(# of characters = 43, # of words = 8))
(  Will be a tattered weed of small worth held:,(# of characters = 46, # of words = 9))
(  Then being asked, where all thy beauty lies,,(# of characters = 46, # of words = 8))
(  Where all the treasure of thy lusty days;,(# of characters = 43, # of words = 8))
(  To say within thine own deep sunken eyes,,(# of characters = 43, # of words = 8))
(  Were an all-eating shame, and thriftless praise.,(# of characters = 50, # of words = 7))
(  How much more praise deserved thy beauty's use,,(# of characters = 49, # of words = 8))
(  If thou couldst answer 'This fair child of mine,(# of characters = 49, # of words = 9))
(  Shall sum my count, and make my old excuse',(# of characters = 45, # of words = 9))
(  Proving his beauty by succession thine.,(# of characters = 41, # of words = 6))
```