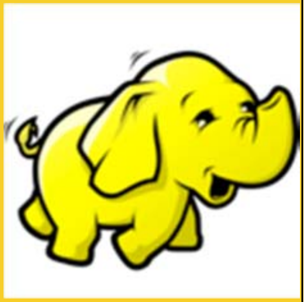
Apache Pig Joining Data-Sets

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/hadoop-tutorial/

Customized Java EE Training: http://courses.coreservlets.com/
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

more SERVLETS and JAVASERVER PAGES
MARTY HALL
2ND EDITION
core SERVLETS and JAVASERVER PAGES
Volume 1: Core Technologies
MARTY HALL · LARRY BROWN



For live Hadoop training, please see courses at http://courses.coreservlets.com/.

Taught by the author of this Hadoop tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 6 or 7 programming, custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - **Hadoop,** Spring, Hibernate/JPA, GWT, SOAP-based and RESTful Web Services

Contact hall@coreservlets.com for details

Agenda

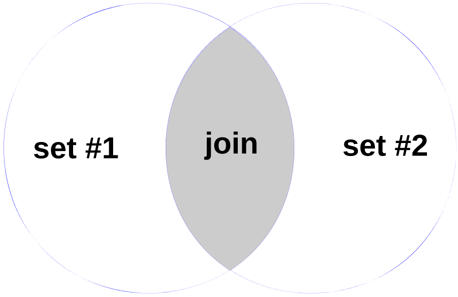
- Joining data-sets
- User Defined Functions (UDF)

Joins Overview

- Critical Tool for Data Processing
- Will probably be used in most of your Pig scripts
- Pigs supports
  - Inner Joins
  - Outer Joins
  - Full Joins

How to Join in Pig

- **Join Steps**
  1. Load records into a bag from input #1
  2. Load records into a bag from input #2
  3. Join the 2 data-sets (bags) by provided join key
- **Default Join is Inner Join**
  - Rows are joined where the keys match
  - Rows that do not have matches are not included in the result



Simple Inner Join Example

1: Load records into a bag from input #1

```
--InnerJoin.pig
posts = load '/training/data/user-posts.txt' using PigStorage(',')
       as (user:chararray,post:chararray,date:long);
```

1:Load records into a bag from input #2

Use comma as a separator

```
likes = load '/training/data/user-likes.txt' using PigStorage(',')
       as (user:chararray,likes:int,date:long);


userInfo = join posts by user, likes by user;
```

3: Join the 2 data-sets

When a key is equal in both data-sets then the rows are joined into a new single row; In this case when user name is equal

```
dump userInfo;
```

Execute InnerJoin.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ hdfs dfs -cat /training/data/user-likes.txt
user1,12,1343182026191
user2,7,1343182139394
user3,0,1343182154633
user4,50,1343182147364

$ pig $PLAY_AREA/pig/scripts-samples/InnerJoin.pig
(user1,Funny Story,1343182026191,user1,12,1343182026191)
(user2,Cool Deal,1343182133839,user2,7,1343182139394)
(user4,Interesting Post,1343182154633,user4,50,1343182147364)
```

user1, user2 and user4 are id that exist in both data-sets; the values for these records have been joined.

Field Names After Join

- **Join re-uses the names of the input fields and prepends the name of the input bag**
  - <bag_name>::<field_name>

```
grunt> describe posts;
posts: {user: chararray,post: chararray,date: long}
grunt> describe likes;
likes: {user: chararray,likes: int,date: long}

grunt> describe userInfo;
UserInfo: {
      posts::user: chararray,
      posts::post: chararray,
      posts::date: long,
      likes::user: chararray,
      likes::likes: int,
      likes::date: long}
```

Schema of the resulting Bag

Fields that were joined from ‘posts’ bag

Fields that were joined from ‘likes’ bag

Join By Multiple Keys

- **Must provide the same number of keys**
- **Each key must be of the same type**

```
--InnerJoinWithMultipleKeys.pig
posts = load '/training/data/user-posts.txt'
        using PigStorage(',')
        as (user:chararray,post:chararray,date:long);

likes = load '/training/data/user-likes.txt'
        using PigStorage(',')
        as (user:chararray,likes:int,date:long);

userInfo = join posts by (user,date), likes by (user,date);

dump userInfo;
```

Only join records whose user **and** date are equal

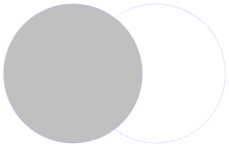
Execute InnerJoinWithMultipleKeys.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ hdfs dfs -cat /training/data/user-likes.txt
user1,12,1343182026191
user2,7,1343182139394
user3,0,1343182154633
User4,50,1343182147364

$ pig $PLAY_AREA/pig/scripts/InnerJoinWithMultipleKeys.pig
(user1,Funny Story,1343182026191,user1,12,1343182026191)
```

There is only 1 record in each data-set where both user and date are equal

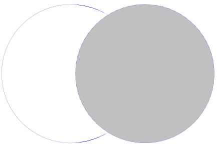
Outer Join

- **Records which will not join with the ‘other’ record-set are still included in the result**



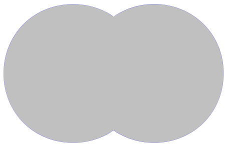
Left Outer

– Records from the first data-set are included whether they have a match or not. Fields from the unmatched (second) bag are set to null.



Right Outer

– The opposite of Left Outer Join: Records from the second data-set are included no matter what. Fields from the unmatched (first) bag are set to null.



Full Outer

– Records from both sides are included. For unmatched records the fields from the ‘other’ bag are set to null.

Left Outer Join Example

```
--LeftOuterJoin.pig
posts = load '/training/data/user-posts.txt'
      using PigStorage(',')
      as (user:chararray,post:chararray,date:long);

likes = load '/training/data/user-likes.txt'
      using PigStorage(',')
      as (user:chararray,likes:int,date:long);

userInfo = join posts by user left outer, likes by user;
dump userInfo;
```

Records in the posts bag will be in the result-set even if there isn’t a match by user in the likes bag

Execute LeftOuterJoin.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ hdfs dfs -cat /training/data/user-likes.txt
user1,12,1343182026191
user2,7,1343182139394
user3,0,1343182154633
User4,50,1343182147364

$ pig $PLAY_AREA/pig/scripts/LeftOuterJoin.pig
(user1,Funny Story,1343182026191,user1,12,1343182026191)
(user2,Cool Deal,1343182133839,user2,7,1343182139394)
(user4,Interesting Post,1343182154633,user4,50,1343182147364)
(user5,Yet Another Blog,13431839394,,,)
```

User5 is in the posts data-set
but NOT in the likes data-set

Right Outer and Full Join

```
--RightOuterJoin.pig
posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',')
      AS (user:chararray,post:chararray,date:long);
likes = LOAD '/training/data/user-likes.txt'
      USING PigStorage(',')
      AS (user:chararray,likes:int,date:long);
userInfo = JOIN posts BY user RIGHT OUTER, likes BY user;
DUMP userInfo;

--FullOuterJoin.pig
posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',')
      AS (user:chararray,post:chararray,date:long);
likes = LOAD '/training/data/user-likes.txt'
      USING PigStorage(',')
      AS (user:chararray,likes:int,date:long);
userInfo = JOIN posts BY user FULL OUTER, likes BY user;
DUMP userInfo;
```

Cogroup

- **Joins data-sets preserving structure of both sets**
- **Creates tuple for each key**
  - Matching tuples from each relationship become fields

```
--Cogroup.pig
posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',')
      AS (user:chararray,post:chararray,date:long);
likes = LOAD '/training/data/user-likes.txt'
      USING PigStorage(',')
      AS (user:chararray,likes:int,date:long);
userInfo = COGROUP posts BY user, likes BY user;
DUMP userInfo;
```

Execute Cogroup.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ hdfs dfs -cat /training/data/user-likes.txt
user1,12,1343182026191
user2,7,1343182139394
user3,0,1343182154633
User4,50,1343182147364

$ pig $PLAY_AREA/pig/scripts/Cogroup.pig
(user1,{(user1,Funny Story,1343182026191)},{(user1,12,1343182026191)})
(user2,{(user2,Cool Deal,1343182133839)},{(user2,7,1343182139394)})
(user3,{},{(user3,0,1343182154633)})
(user4,{(user4,Interesting Post,1343182154633)},{(user4,50,1343182147364)})
(user5,{(user5,Yet Another Blog,13431839394)},{})
```

Tuple per key

First field is a bag which came from posts bag (first data-set); second bag is from the likes bag (second data-set)

Cogroup with INNER

- Cogroup by default is an OUTER JOIN
- You can remove empty records with empty bags by performing INNER on each bag
  - ‘INNER JOIN’ like functionality

```
--CogroupInner.pig
posts = LOAD '/training/data/user-posts.txt'
       USING PigStorage(',')
       AS (user:chararray,post:chararray,date:long);
likes = LOAD '/training/data/user-likes.txt'
       USING PigStorage(',')
       AS (user:chararray,likes:int,date:long);
userInfo = COGROUP posts BY user INNER, likes BY user INNER;
DUMP userInfo;
```

Execute CogroupInner.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ hdfs dfs -cat /training/data/user-likes.txt
user1,12,1343182026191
user2,7,1343182139394
user3,0,1343182154633
User4,50,1343182147364

$ pig $PLAY_AREA/pig/scripts/CogroupInner.pig
(user1,{(user1,Funny Story,1343182026191)},{(user1,12,1343182026191)})
(user2,{(user2,Cool Deal,1343182133839)},{(user2,7,1343182139394)})
(user4,{(user4,Interesting Post,1343182154633)},{(user4,50,1343182147364)})
```

Records with empty bags are removed

User Defined Function (UDF)

- **There are times when Pig’s built in operators and functions will not suffice**
- **Pig provides ability to implement your own**
  - Filter
    - Ex: res = FILTER bag BY udfFilter(post);
  - Load Function
    - Ex: res = load 'file.txt' using udfLoad();
  - Eval
    - Ex: res = FOREACH bag GENERATE udfEval($1)
- **Choice between several programming languages**
  - Java, Python, Javascript

Implement Custom Filter Function

- **Our custom filter function will remove records with the provided value of more than 15 characters**
  - filtered = FILTER posts BY isShort(post);
- **Simple steps to implement a custom filter**
  1. Extend FilterFunc class and implement exec method
  2. Register JAR with your Pig Script
     - JAR file that contains your implementation
  3. Use custom filter function in the Pig script

1: Extend FilterFunc

- **FilterFunc class extends EvalFunc**
  - Customization for filter functionality
- **Implement exec method**
  - public Boolean exec(Tuple tuple) throws IOException
  - Returns false if the tuple needs to be filtered out and true otherwise
  - Tuple is a list of ordered fields indexed from 0 to N
    - We are only expecting a single field within the provided tuple
    - To retrieve fields use tuple.get(0);

1: Extend FilterFunc

```
public class IsShort extends FilterFunc{
  private static final int MAX_CHARS = 15;

  @Override
  public Boolean exec(Tuple tuple) throws IOException {
    if ( tuple == null || tuple.isNull() || tuple.size() == 0 ){
       return false;
    }
    Object obj = tuple.get(0);
    if ( obj instanceof String){
       String st = (String)obj;
       if ( st.length() > MAX_CHARS ){
              return false;
       }
       return true;
    }
    return false;
  }
}
```

extend FilterFunc and implement exec function

Default to a single field within a tuple

Pig's CHARARRAY type will cast to String

Filter out Strings shorter than 15 characters

Any Object that can not cast to String will be filtered out

2: Register JAR with Pig Script

- **Compile your class with filter function and package it into a JAR file**
- **Utilize REGISTER operator to supply the JAR file to your script**

```
REGISTER HadoopSamples.jar
```

  - The local path to the jar file
  - Path can be either absolute or relative to the execution location
  - Path must NOT be wrapped with quotes
  - Will add JAR file to Java's CLASSPATH

3: Use Custom Filter Function in the Pig Script

- **Pig locates functions by looking on CLASSPATH for fully qualified class name**

```
filtered = FILTER posts BY pig.IsShort(post);
```

- **Pig will properly distribute registered JAR and add it to the CLASSPATH**
- **Can create an alias for your function using DEFINE operator**

```
DEFINE isShort pig.IsShort();
...
...
filtered = FILTER posts BY isShort(post);
...
```

Script with Custom Function

```
--CustomFilter.pig
REGISTER HadoopSamples.jar
DEFINE isShort pig.IsShort();
```

Pig custom functions are packaged in the JAR and can be used in this script

Create a short alias for your function

```
posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',')
      AS (user:chararray,post:chararray,date:long);
```

Script defines a schema; post field will be of type chararray

```
filtered = FILTER posts BY isShort(post);
dump filtered;
```

Execute CustomFilter.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ pig pig/scripts/CustomFilter.pig
(user1,Funny Story,1343182026191)
(user2,Cool Deal,1343182133839)
```

Posts whose length exceeds 15 characters have been filtered out

Filter Function and Schema

- **What would happen to pig.IsSort custom filter if the schema was NOT defined in the script**

```
--CustomFilter-NoSchema.pig
REGISTER HadoopSamples.jar
DEFINE isShort pig.IsShort();

posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',');
```

LOAD does not define schema

Since no schema defined will need to reference second field by an index

```
filtered = FILTER posts BY isShort($1);
dump filtered;
```

Execute CustomFilter-NoSchema.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ pig pig/scripts/CustomFilter-NoSchema.pig
$
```

Why did CustomFilter-NoSchema.pig produce no results?

Why did CustomFilter-NoSchema.pig Produce no Results?

- **Recall that the script doesn’t define schema on LOAD operation**

```
posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',');
filtered = FILTER posts BY isShort($1);
```

- **When type is not specified Pig default to bytearray – DataByteArray class**
- **Recall our custom implementation IsShort.exec**

```
Object obj = tuple.get(0);
if ( obj instanceof String){
  ...
  ...
}
return false;
```

Since script never defined schema obj will be of type DataByteArray and filter will remove ALL records

Make IsShort Function Type Aware

- **Override getArgToFuncMapping method on EvalFunc, parent of FilterFunc**
  - Specify expected type of the functions parameter(s)
  - Method returns a List of User Defined Functions (UDF) specifications – FuncSpec objects
  - Each object represents a parameter field
  - In our case we just need to provide a single FuncSpec object to describe field’s type

```
filtered = FILTER posts BY isShort($1);
```

FuncSpec object will describe function’s parameter

GetArgToFuncMapping method of IsShortWithSchema.java

```
@Override
public List<FuncSpec> getArgToFuncMapping()
                    throws FrontendException {
  List<FuncSpec> schemaSpec = new ArrayList<FuncSpec>();

  FieldSchema fieldSchema = new FieldSchema(
       null,
       DataType.CHARARRAY);
```

First argument is field alias and is ignored for type conversion

Second argument is the type – CHARARRAY that will cast to String

```
  FuncSpec fieldSpec = new FuncSpec(
       this.getClass().getName(),
       new Schema(fieldSchema));
```

Name of the function

Schema for the function; in this case just one field

```
  schemaSpec.add(fieldSpec);
  return schemaSpec;
}
```

Returns FuncSpec object that describes metadata about each field

CustomFilter-NoSchema.pig

```
--CustomFilter-NoSchema.pig
REGISTER HadoopSamples.jar
DEFINE isShort pig.IsShortWithSchema();

posts = LOAD '/training/data/user-posts.txt'
      USING PigStorage(',');

filtered = FILTER posts BY isShort($1);

dump filtered;
```

Improved implementation of filter with type specification

This Pig script still does NOT specify type of the function's parameter

Execute CustomFilter-NoSchema.pig

```
$ hdfs dfs -cat /training/data/user-posts.txt
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394

$ pig pig/scripts/CustomFilter-WithSchema.pig
(user1,Funny Story,1343182026191)
(user2,Cool Deal,1343182133839)
```

Improved implementation specified the parameter type to be CHARARRAY which will then cast to String type

Wrap-Up

Summary

- **We learned about**
  - Joining data-sets
  - User Defined Functions (UDF)

Questions?