小象科技
ChinaHadoop.cn

Pig数据流执行引擎

讲师：董西成

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

Pig是什么

小象科技 ChinaHadoop.cn

- Hadoop上的数据流执行引擎（由Yahoo！开源）
  - ✓ 利用HDFS存储数据
  - ✓ 利用MapReduce处理数据
- 使用Pig Latin语言表达数据流
  - ✓ Pig Latin是一种新的数据流语言
  - ✓ Pig将Pig Latin语句转化为MapReduce作业
  - ✓ Pig Latin比MapReduce程序更易编写
- 直接产生动机：让MapReduce用起来更简单
  - ✓ 与Hive一致

Pig与Hive异同

小象科技 ChinaHadoop.cn

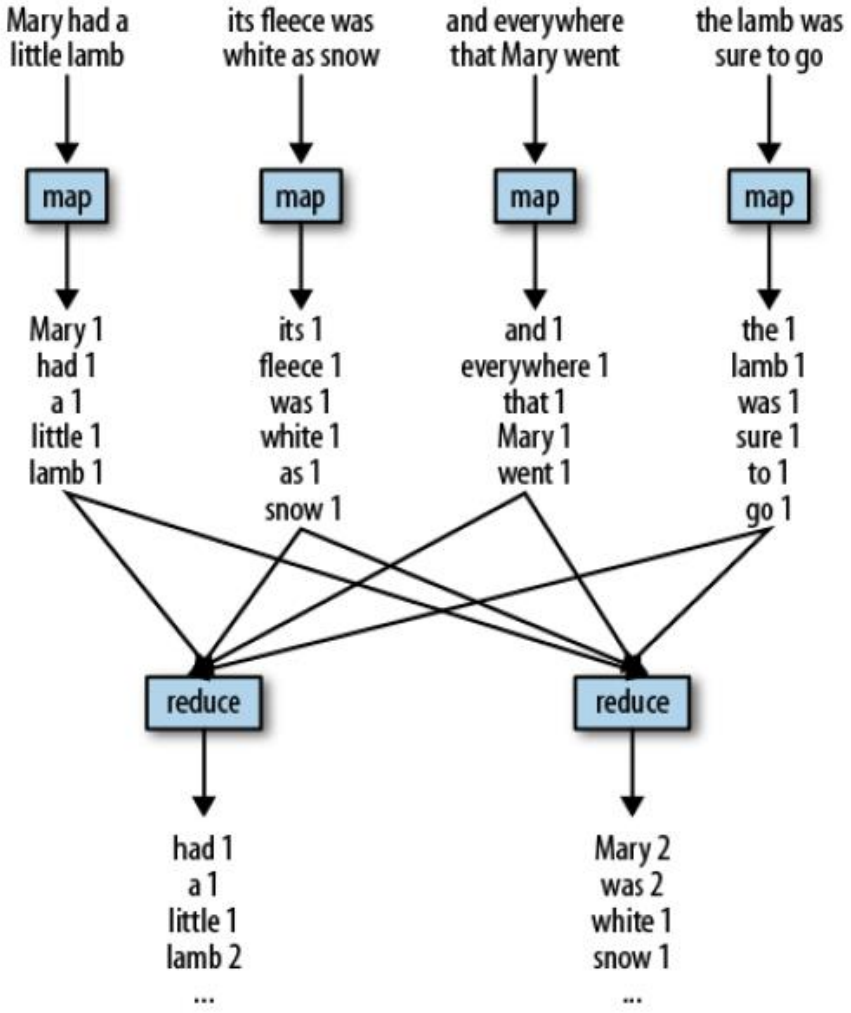
- 相同点
  - ✓ 运行在Hadoop之上；
  - ✓ 设计动机是为用户提供一种更简单的Hadoop上数据分析方式；
  - ✓ 解决相同问题的两个工具（yahoo！vs facebook）。
- 不同点
  - ✓ Hive要求待处理数据必须有Schema，而Pig则无此要求；
  - ✓ Hive有Server需要安装，Pig无Server不需要安装；
  - ✓ 编程语言不同，SQL与Pig Latin
    - SQL：得到什么样的结果，Pig Latin：如何处理数据
    - SQL：过程化语言，Pig Latin：数据流语言

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

WordCount问题



WordCount实现

小象科技 ChinaHadoop.cn

```
-- ① 加载数据
input = load '/input/data' as (line:chararray);
-- ② 将字符串分割成单词
words = foreach input generate
        flatten(TOKENIZE(line)) as word;
-- ③ 对单词进行分组
grpd = group words by word;
-- ④ 统计每组中单词数量
cntd = foreach grpd generate group,
       COUNT(words);
-- ⑤ 打印结果
dump cntd;
```

运行Pig Latin

小象科技 ChinaHadoop.cn

- 本地模式
  - ✓ pig_path/bin/pig –x local wordount.pig
- 集群模式
  - ✓ PIG_CLASSPATH=hadoop_conf_dir pig_path/bin/pig wordcount.pig
- 其他使用方式
  - ✓ pig -e fs –copyFromLocal local_path hdfs_path
  - ✓ pig hdfs://nn.mydomain.com:9020/myscripts/script.pig
  - ✓ pig –Dmapreduce.task.profile=true wordount.pig
  - ✓ pig –P myproperty.properties wordcount.pig

通常结合Oozie等使用

小象科技 ChinaHadoop.cn

- Oozie是什么
  - ✓作业流调度系统
  - ✓根据配置对作业进行周期调度或者定时调度
  - ✓同时支持MapReduce、Hive、Pig等
  - ✓包含监控、报警等功能
- 怎样让Oozie调度pig作业
  - ✓ Oozie将Pig作业封装成了action
  - ✓ 可让Pig作业与MapReduce、Sqoop等连用

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

数据模型

- 可通过LOAD、FOREACH和STREAM三个操作符将数据模式化
- 可为每个模式指名每个字段的类型，也可以不指名，默认是bytearray
- 举例如下：

```
A = LOAD ‘data’ AS (f1:int, f2:int); // 提倡写法
A = LOAD ‘data’ AS (f1, f2); //f1和f2均为bytearray
A = LOAD ‘data’  //可以这样定义，但是schema未知
--
X = FOREACH A GENERATE f1+f2 AS x1:int;
--
A = LOAD 'data';
B = STREAM A THROUGH `stream.py -n 5` AS (f1:int, f2:int);
```

数据类型—简单数据类型

小象科技 ChinaHadoop.cn

类型	大小	举例
int	4 byte 有符号整型	20
long	8 byte 有符号整型	20
float	单精度浮点型	20
double	双度浮点型	20
chararray	UTF-8编码的字符串	hello world
bytearray	Byte数组	
boolean	布尔类型	true/false
datetime	时间	1970-01-01T00:00:00.000+00:00
biginteger	Java BigInteger	200000000000
bigdecimal	Java BigDecimal	33.4567833213234412334442

数据类型—复杂数据类型

小象科技 ChinaHadoop.cn

类型	大小	举例
tuple	有序数据集	(19,2)
bag	Tuple构成的集合	{(19,2), (18,1)}
map	Key/value集合	[open#apache]

复杂数据类型—map

小象科技 ChinaHadoop.cn

- 数据格式

>cat data;

[open#apache]

[apache#hadoop]

- 将数据加载到map中

A = LOAD ‘data’ AS (M:map []);

A = LOAD ‘data’ AS (M:[])

复杂数据类型—tuple

- 数据格式

>cat data

(3,8,9) (mary,19)

(1,4,7) (john,18)

(2,5,8) (joe,18)

- 将数据加载到tuple中

```
A = LOAD data AS
(F:tuple(f1:int,f2:int,f3:int),T:tuple(t1:chararray,t2:int));
DESCRIBE A;
A: {F: (f1: int,f2: int,f3: int),T: (t1: chararray,t2: int)}
```

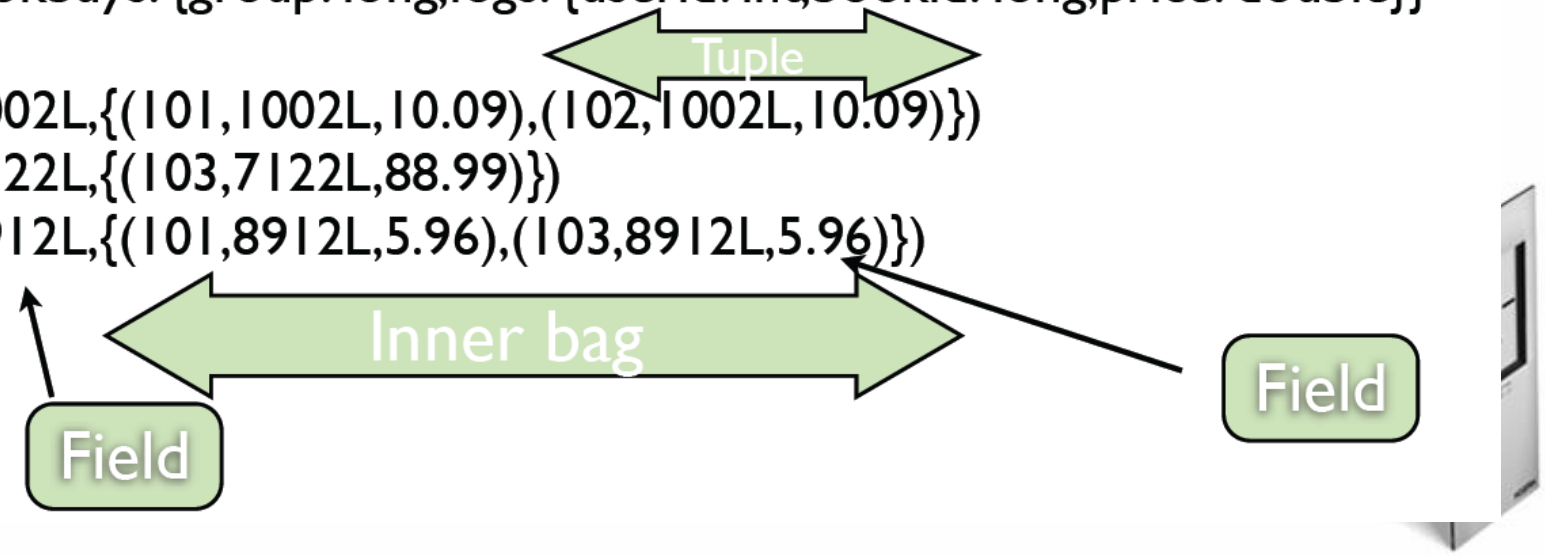
复杂数据类型—tuple与bag

```
$ cat logs
101     1002    10.09
101     8912    5.96
102     1002    10.09
103     8912    5.96
103     7122    88.99
```

```
$ cat groupbooks.pig
logs = LOAD 'logs' AS
  (userid: int, bookid: long, price: double);
bookbuys = GROUP logs BY bookid;
DESCRIBE bookbuys;
DUMP bookbuys;
```

```
$ pig -x local groupbooks.pig
bookbuys: {group: long,logs: {userid: int,bookid: long,price: double}}

(1002L,{(101,1002L,10.09),(102,1002L,10.09)})
(7122L,{(103,7122L,88.99)})
(8912L,{(101,8912L,5.96),(103,8912L,5.96)})
```



常用操作

- Foreach
- Filter
- Group
- Order by
- Distinct
- Join
- Limit
- Sample
- parallel

load操作

小象科技 ChinaHadoop.cn

- 加载/logs/2014/04目录下的日志，并保存到clicklogs中：

clicklogs = LOAD 'logs/2014/04';

- 对字段进行重命名

clicklogs = LOAD ‘logs/2014/04‘ AS (userid: int, url: chararray);

- 默认字段间分隔符是TAB，可修改分隔符：

LOAD ‘logs/2014/04’ USING PigStorage(‘,’)

- 注意：Load操作并不会真正的执行，直到遇到a dump/store操作

Describe, Dump与Store

小象科技 ChinaHadoop.cn

- Describe用于描述变量的schema：

```
describe combotimes;
combotimes: {group: chararray,
  enter: {time: chararray,userid: chararray},
  exit: {time: chararray,userid: chararray,cost: double}}
```

- **Dump将结果打印到终端上：**

```
DUMP combotimes;
```

- **Store将结果保存到目录或者文件中：**

```
STORE combotimes INTO ‘result/2014’;
```

foreach

- **FOREACH <bag> GENERATE <data>**
  - Iterate over each element in the bag and produce a result
  - Ex: grunt> result = FOREACH bag GENERATE f1;

```
grunt> records = LOAD 'data/a.txt' AS (c:chararray, i:int);
grunt> dump records;
(a,1)
(d,4)
(c,9)
(k,6)
grunt> counts = foreach records generate i;
grunt> dump counts;
(1)
(4)
(9)
(6)
```

For each row emit 'i' field

Tokenize函数

- **Splits a string into tokens and outputs as a bag of tokens**
  - Separators are: space, double quote("), coma(,) parenthesis(()), star(*)

```
grunt> linesOfText = LOAD 'data/c.txt' AS (line:chararray);
grunt> dump linesOfText;
(this is a line of text)
(yet another line of text)
(third line of words)

grunt> tokenBag = FOREACH linesOfText GENERATE TOKENIZE(line);

grunt> dump tokenBag;
({(this),(is),(a),(line),(of),(text)})
({(yet),(another),(line),(of),(text)})
({(third),(line),(of),(words)})
grunt> describe tokenBag;
tokenBag: {bag_of_tokenTuples: {tuple_of_tokens: (token: chararray)}}
```

Split each row line by space and return a bag of tokens

Each row is a bag of words produced by TOKENIZE function

group

```
$ cat starnames
1       Mintaka
2       Alnitak
3       Epsilon Orionis
```

```
$ cat starpositions
1       R.A. 05h 32m 0.4s, Dec. -00 17' 57"
2       R.A. 05h 40m 45.5s, Dec. -01 56' 34"
3       R.A. 05h 36m 12.8s, Dec. -01 12' 07"
```

```
$ cat starsandpositions.pig
names = LOAD 'starnames' as (id: int, name: chararray);
positions = LOAD 'starpositions' as (id: int, position: chararray);
nameandpos = GROUP names BY id, positions BY id;
DESCRIBE nameandpos;
DUMP nameandpos;
```

```
nameandpos: {group: int,names: {id: int,name: chararray},
  positions: {id: int,position: chararray}}
(1,{(1,Mintaka)},{(1,R.A. 05h 32m 0.4s, Dec. -00 17' 57")})
(2,{(2,Alnitak)},{(2,R.A. 05h 40m 45.5s, Dec. -01 56' 34")})
(3,{(3,Epsilon Orionis)},{(3,R.A. 05h 36m 12.8s, Dec. -01 12' 07")})
```

Join


```
$ cat starsandpositions2.pig
names = LOAD 'starnames' as (id: int, name: chararray);
positions = LOAD 'starpositions' as (id: int, position: chararray);
nameandpos = JOIN names BY id, positions BY id;
DESCRIBE nameandpos;
DUMP nameandpos;
```

```
nameandpos: {names::id: int,names::name: chararray,
positions::id: int,positions::position: chararray}

(1,Mintaka,1,R.A. 05h 32m 0.4s, Dec. -00 17' 57")
(2,Alnitak,2,R.A. 05h 40m 45.5s, Dec. -01 56' 34")
(3,Epsilon Orionis,3,R.A. 05h 36m 12.8s, Dec. -01 12' 07")
```

其他（一）

小象科技 ChinaHadoop.cn

- Filter（过滤）：

```
-- filter_matches.pig
divs = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray,
date:chararray, dividends:float);
startswithcm = filter divs by symbol matches 'CM.*';
```

- Distinct（去重）：

```
-- dictinct.pig
daily = load 'NYSE_daily' as (exchange:chararray, symbol:chararray);
uniq = distinct daily;
```

- Limit（返回N条结果）：

```
--limit.pig
divs = load 'NYSE_dividends';
first10 = limit divs 10;
```

其他（二）

小象科技 ChinaHadoop.cn

- Sample（采样，百分比）：

```
--sample.pig
divs = load 'NYSE_dividends';
some = sample divs 0.1;
```

- Parallel（设置reduce task个数）：

```
--parallel.pig
daily = load 'NYSE_daily' as (exchange, symbol, date, open, high, low, close,
        volume, adj_close);
bysymbl = group daily by symbol parallel 10;
```

- Order by（排序）：

```
--order2key.pig
daily = load 'NYSE_daily' as (exchange:chararray, symbol:chararray,
        date:chararray, open:float, high:float, low:float,
        close:float, volume:int, adj_close:float);
bydatensymbol = order daily by date desc, symbol;
```

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

Join优化

小象科技 ChinaHadoop.cn

- fragment-replicate join或者shuffle join或者map-side join（一个大表一个小表）：

```
--repljoin.pig
daily = load 'NYSE_daily' as (exchange:chararray, symbol:chararray,
          date:chararray, open:float, high:float, low:float,
          close:float, volume:int, adj_close:float);
divs = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray,
         date:chararray, dividends:float);
jnd = join daily by (exchange, symbol), divs by (exchange, symbol)
       using 'replicated';
```

注：后一个表是小表

- Skew data join（一个表某个key特别多）：

```
users = load 'users' as (name:chararray, city:chararray);
cinfo = load 'cityinfo' as (city:chararray, population:int);
jnd = join cinfo by city, users by city using 'skewed';
```

注：后一个表是skew 表

Join优化

小象科技 ChinaHadoop.cn

- 两个有序表join：

```
--mergejoin.pig
daily = load 'NYSE_daily_sorted' as (exchange:chararray, symbol:chararray,
        date:chararray, open:float, high:float, low:float,
        close:float, volume:int, adj_close:float);
divs = load 'NYSE_dividends_sorted' as (exchange:chararray, symbol:chararray,
       date:chararray, dividends:float);
jnd = join daily by symbol, divs by symbol using 'merge';
```

Pig参数调优

小象科技 ChinaHadoop.cn

- pig.cachedbag.memusage：

Map task或者reduce task中，bag数据占用内存达到JVM内存的一定百分比后，数据将被溢写到磁盘上，默认值是0.1；

- mapred.compress.map.output

是否对Map Task中间结果进行压缩，如果设置为true，可进一步使用mapred.map.output.compression.codec.设置采用的压缩器；

- pig.noSplitCombination

是否将不及一个block大小的多个小文件交由一个Task处理，默认是false，注意，这会降低数据本地性。

- 其他参数调优

MapReduce级别的buffer调优，包括io.sort.mb等

主要内容

小象科技 ChinaHadoop.cn

1. Pig是什么
2. Pig使用
3. Pig Latin
4. Pig调优
5. 总结

总结

- Pig是为简化MapReduce分析数据而提出的数据分析工具；
- Pig是无模式化的，且仅是一个客户端程序，便于使用（通常跟Oozie等作业流调度系统结合使用）；
- Pig提供了强大的数据分析运算符，能够完成复杂的数据分析；
- Pig与Hive功能类似，可选其一使用。

小象科技 ChinaHadoop.cn

联系我们：

- 新浪微博：ChinaHadoop
- 微信公号：ChinaHadoop

让你的数据产生价值！