Pig详解

目标

- 了解Pig的作用
- 了解Pig的设计思想
- 了解Pig的运行模式
- 掌握Pig Latin的基础知识
- 掌握Pig脚本的运行方法

Pig概述

Pig是Hadoop的一个扩展，它简化了Hadoop的编程，提供了一套高级数据处理语言，且保持了Hadoop易于扩展与可靠的特性。

Pig有两个主要的组成部分：
高级数据处理语言Pig Latin；
依据可供抽样的评价机制编译与运行Pig Latin脚本的编译器。

Pig Latin是Pig的脚本语言，主要特性：易于编程、自动优化、可扩展性

Pig的用途

Pig的主要应用场景是传统的数据流处理、原生数据研究和迭代处理。另外，Pig还可以用来处理离线用户数据，对用户行为进行预测，比如扫描所有的用户和网站的交互数据，将用户进行分类，对每一类建立一个数学模型，通过分析该模型，可以预测某类用户对各种类型的广告或新闻会作出怎样的反应，从而有针对性地展示这类用户感兴趣的广告。

Pig采用面向数据批处理的模式，因此，如果需要处理大量的数据，如GB或者TB数量级的数据，则Pig是个不错的选择，但对于需要写单条或者少量记录的任务，Pig并非十分适用。

Pig的设计思想

Pig的主要设计思想：
不管数据是否有元数据，Pig都可以操作；不管数据是关系型的、嵌套型的或者是非结构化的，Pig都可以操作；Pig还应该很容易扩展，不仅可以操作文件，还可以操作键值型的存储和数据库等。

Pig的设计目标并非是只能用于Hadoop平台。目前，Pig支持用户自定义字段类型转换函数、用户自定义聚合方法函数和用户自定义条件式函数，允许用户随时整合加入自己的代码。

Pig还有一个优化控制器，可以重新排列脚本中的操作过程以达到更好的性能，如果用户不希望进行这种优化，也可以很容易地将该控制器关闭，就不会改变执行过程。

Pig的运行模式-1

Pig有两种运行模式：Local模式和MapReduce模式，默认使用MapReduce模式。

当Pig在Local模式下运行时，Pig只访问本地一台主机；而当Pig在MapReduce模式下运行时，它将访问一个Hadoop集群和HDFS的安装位置，此时Pig将自动对这个集群进行分配和回收。

Pig的Local模式和MapReduce模式都有三种运行方式，分别为Grunt Shell方式、脚本文件方式和嵌入式程序方式。

Pig的运行模式-2

1. Local模式

在Local模式下，Pig在单个JVM中访问本地文件系统，该模式用于测试或处理小规模数据集。

1) Grunt Shell方式

在Linux终端中执行如下命，开启Pig的Grunt Shell窗口。

```
$pig –x local
```

Grunt Shell窗口与Windows系统中的Dos窗口非常类似，用户可以在这里逐条输入命令对数据进行操作。

Pig的运行模式-3

2) 脚本文件方式

该方式以脚本文件来进行Pig命令的批处理运行，这些脚本文件实际上是Pig命令的集合。

执行如下命令，可在本地模式下运行Pig脚本。

```
$Pig -x local script.Pig
```

其中，script.Pig是要运行的Pig脚本，这里需要正确指定该Pig脚本的位置，否则系统将无法识别，例如，假设Pig脚本位于/root/PigTmp目录下，则需要写出此脚本的完整路径“/root/PigTmp/script.Pig”。

Pig的运行模式-4

3) 嵌入式程序方式
与运行普通的Java程序相同，首先需要编写特定的Java程序，并将其编译生成对应的.class文件或package包，然后调用main函数运行该程序。
可以使用如下命令，对Java源文件进行编译。
$javac -cp Pig-*.*.*-core.jar local.java

Pig-*.*.*-core.jar为Java源文件编译过后的打包文件，位于Pig安装目录下，local.java为用户编写的Java源文件，两个文件都需要正确指定其位置。

编译完成后，Java会生成一个文件local.class，可以使用如下命令，调用此文件。
$ java -cp Pig-*.*.*-core.jar:. local

Pig的运行模式-5

2. MapReduce模式
在MapReduce 模式下，Pig可以访问整个Hadoop集群，处理大规模数据集。
1) Grunt Shell方式
在Linux终端上执行如下命令，就可以进入Grunt Shell的MapReduce模式。
$Pig -x mapreduce

2) 脚本文件方式
执行如下命令，可以在MapReduce模式下运行Pig脚本文件。
$Pig -x mapreduce script.Pig

3) 嵌入式程序方式
与Local模式相同，在MapReduce模式下运行嵌入式程序同样需要经过编译和执行两个步骤，可以使用如下命令完成相应操作。
javac -cp Pig-xxx-core.jar mapreduce.java
java -cp Pig-xxx -core.jar:. mapreduce

Pig Latin

Pig Latin是Pig针对MapReduce算法（框架）开发的一套Shell脚本，类似用户熟悉的SQL语句，这套脚本可以对加载完毕的数据进行排序、过滤、求和、分组（group by）、关联（joining）等操作。

Pig Latin是一种数据流语言，每一步处理都会产生一个新的数据集或者一个新的关系，比如下面这个脚本：

input=load'data'

该脚本中，“input”是加载数据集data之后的结果的关系名称，这里的关系名称即指通常所说的别名，但需要注意的是，关系名称与变量不同，一旦声明了，这个名称就是不变的了。

Pig中的另一个常见概念是字段名称，它是一个关系所包含的字段（或者称为列）的名称。关系名称和字段名称都必须以字母字符开头，后面可以跟上零个或多个字母、数字或者下划线。

Pig中的关系名称和字段名称是大小写敏感的，例如“A=load'foo';”和“a=load'foo';”是不等价的；但是，Pig中的关键字是不分大小写的，例如LOAD和load就是等价的。

Pig Latin——读写和检测操作符

Pig Latin中使用的读/写操作符如表：

操作符	说明
LOAD	语法：alias=LOAD 'file' [USING function] [AS schema]; 作用：从文件中装载数据。如果不使用USING选项，则默认使用PigStorage装载函数。数据可以使用AS选项给出schema
LIMIT	语法：alias=LIMIT alias n; 作用：限制元素个数为n，即当作用于alias时，LIMIT返回前n个元素，如不使用，则不能保证哪些元素会被返回
DUMP	语法：DUMP a1ias; 作用：在屏幕上显示数据
STORE	语法：STORE alias INTO 'directory' [USING function]; 作用：将一个关系中的数据存储到一个目录中，并把关系存储在以“part-nnnnn”为名的文件中，如果不使用USTNG选项特别指定，则默认使用PigStorage存储函数

Pig Latin入门例子：

在Pig安装目录下有一个文件tutoria1/data/excite-small.log，其中的数据分为3列，中间用制表符分隔，第一列为用户ID，第二列为Unix时间戳，第三列则为查询记录。首先，从该文件的4500条记录中，选取一段样本如下。

```
3F8AAC2372F6941C        970916091301  bac
3F8AAC2372F6941C        970916091354  blood alcohol content
3F8AAC2372F6941C        970916091425
3F8AAC2372F6941C        970916091545
3F8AAC2372F6941C        970916093448
3F8AAC2372F6941C        970916093544  breathalizers
3F8AAC2372F6941C        970916093551  breathalizers
3F8AAC2372F6941C        970916093642  breathalizers
3F8AAC2372F6941C        970916093724  minors in possesion
3F8AAC2372F6941C        970916093848  minors in possesion
3F8AAC2372F6941C        970916093904  mip
```

然后，在Grunt Shell中输入如下命令，将数据装载到一个称为log的别名中。

```
grunt> log = LOAD ‘tutoria1/data/excite-small. Log’ AS (user, time , query);
```

LIMIT命令允许指定有多少元组（行）用于返回结果。本例中，如果要查看别名log的4个元组，则可使用以下命令。

```
grunt> 1mt = LIMIT log 4;
grunt> DUMP 1mt;
```

返回结果如下。

```
2A9EABFB35F5B954        970916105432        +md foods +proteins
BED75271605EBD0C        970916001949        yahoo chat
BED75271605EBD0C        970916001954        yahoo chat
BED75271605EBD0C        970916003523        yahoo chat
```

执行以下命令，完成统计每个用户发起的查询个数。

```
grunt> log = LOAD 'tutorial/data/excite-small.log' AS (user:chararray, time:long, query:chararray) ;
grunt> grpd = GROUP log BY user;
grunt> cntd = FOREACH grpd GENERATE group , COUNT(log);
grunt> STORE cntd INTO 'output';
```

部分统计结果如下。

```
002BB5A52580A8ED        18
005BD9CD3AC6BB38        18
00A08A54CD03EB95        3
011ACA65C2BF70B2        5
01500FAFE317B7C0        15
0158F8ACC570947D        3
018FBF6BFB213E68        1
019E9463F6695963        10
```

Pig Latin——数据类型和schema

Pig有6个简单的原子类型和3个复杂的类型。原子类型包括数字标量、字符串和二进制对象，类型间转换的实现和常规方式相同，除非特别声明，字段默认为bytearray。

数据类型	说明
int	整数，存储一个4个字节大小的带符号整数，如36
long	长整型，存储一个8个字节大小的带符号整数，以一个结尾为L的整数来表示，如5000000000L
float	浮点数，用4个字节存储值，通过一个浮点数加上f来表示，例如3.14f
double	双精度浮点数，用8个字节存储值，可以使用简单的格式表示，例如3.25628
chararray	字符串或者字符数组，以加单引号的一系列字符来表示，如'food'，也可以通过转义符反斜杠表示一些特定的字符，例如\n表示回车
bytearray	一组字节，通过封装Java的byte[]的DataByteArray类来实现，没有办法单独定义一个bytearray常量

3个复杂数据类型是元组（tuple）、包（bag）和映射表（map）。

数据类型	说明	例子
tuple（元组）	一个定长的包含有序Pig数据元素的集合，一个元组相当于SQL中的一行，而元组的字段相当于SQL中的列，其表现形式为逗号分隔的字段，前后用小括号包裹	(hello world,12.5,-2,123)
bag（包）	元组的无序集合（允许元组重复），表示为逗号分隔的元组，前后由大括号包裹，一个包中的元组不必有相同的schema，甚至字段也不必相同	{(hello world,12.5,-2,123),(2.87, bye world,10)}
map（映射）	一组键值对，键必须是唯一的字符串（chararray），值可以是任意类型的数据	['name'#'bob', 'age'#21]

除了为字段声明类型，schema还可以为字段命名，使它们更易于引用，用户可以在LOAD、STREAM和FOREACH命令中使用AS关键字为关系定义字段名。
在定义schema时，如果遗漏了类型，Pig会默认将bytearray作为最常用的类型。也可以不设置字段的名称，此时该字段的状态为未命名，只能通过位置来引用它。

Pig Latin——表达式和函数

将表达式和函数应用于数据字段，可以计算出各种数值。最简单的表达式为常数值。表达式也可以引用一个字段的值，既可以通过名字直接引用命名字段的值，也可以使用$n命令引用一个未命名的字段，这里的n是该字段在元组中的位置，从0开始计数。例如，关系log有3个命名的字段，分别为user、time和query，则可以通过“time”或者“$1”来引用time字段。

Pig还支持标准算术表达式、比较表达式、条件表达式、类型转换表达式和布尔表达式，它们常见于大多数流行的编程语言中，如表

表达式	说明
12，19.2，'hello world'	常数值，没有小数点的数值为int型，数字之后带有l或L则为long型；带有小数点的数值为double型，数字之后有f或者F则为float型
+，-，*，/	加减乘除
+x，-x	负号（-），改变一个数字的符号
t(x)	将x的值转换为t类型
x%y	x被y除的余数
(x?y:z)	如果x为真则返回y，否则为z，该表达式必须用圆括号包裹
==，!=，<，>，<=，>=	分别为等于，不等于，小于，大于，不大于，不小于
x matches regex	与字符串x匹配的正则表达式
x is null x is not null	检查x是否为空
x and y， x or y not x	布尔值：与、或、非

Pig也支持函数，Pig的内置函数及用法如下：
avg用法——avg(expression)：计算单列值的平均数，忽略NULL值，在使用group all或group单列后可使用。
contact用法-———contact(expression1,expression2)：将两个字段的值拼接为一个字符串，如果其中一个为NULL，则结果用NULL表示。
count用法——count(expression)：统计在一个bag中所有元素的数量，不包含NULL值统计，同时需要以group的支持为前提。
diff用法——diff(expression1,expression2)，比较一个元组中的两个fields集合的差异性，与Linux或Python里面的diff函数类似。
IsEmpty用法——IsEmpty(expression)：判断一个bag或map是否为空（没有数据），可以用在filter数据过滤中。
max用法——max(expression)：计算单列中最大的数值，或者字符串的最大值（字典排序），同count一样需要group支持。
min用法——min(expression)：计算单列中最小的数值，或者字符串的最小值（字典排序），同count一样需要group支持。
size用法——size(expression)：计算任何Pig字符串的大小长度，或者集合类型的的长度。
subtract用法——subtract(expression1,expression2)：对两个bag里面的元组做差值操作。
sum用法——sum(expression)：对某列求和，需要提前使用group分组。
tokenize用法——tokenize(expression,'field_delimiter')：按照指定分隔符拆分一句话，然后转成一系列的words。

Pig Latin——关系型运算符

有两个文件A与B，
A文件内容如下。
0,1,2
1,3,4
B文件内容如下。
0,5,2
1,7,8

定义关系a与b，代码如下。

```
grunt> a = load 'A' using PigStorage(',') as ( a1:int, a2:int, a3:int);
grunt> b = load 'B ' using PigStorage (',') as (b1:int, b2:int, b3:int);
grunt> DUMP a;
(0,1,2)
(1,3,4)
grunt> DUMP b;
(0,5,2)
(1,7,8)
```

1. UNION和SPLIT

UNION将多个关系归并在一起，SPLIT则将一个关系分割为多个，代码如下。

```
grunt> c =UNION a , b ;
grunt> DUMP c;
(0,1,2)
(1,3,4)
(0,5,2)
(1,7,8)
grunt> SPLIT c INTO d IF $0 == 0 , e IF $0 == 1;
grunt> DUMP d;
(0,1,2)
(0,5,2)
grunt> DUMP e;
(1,3,4)
(1,7,8)
```

UNION运算符允许重复，可以使用DISTINCT运算符对关系进行去重（dis=distinct c;）。在c上的SPLIT操作将一个元组传给另一个关系，如果第一个字段($0)为0，则送到d，如果为1，则送到e。