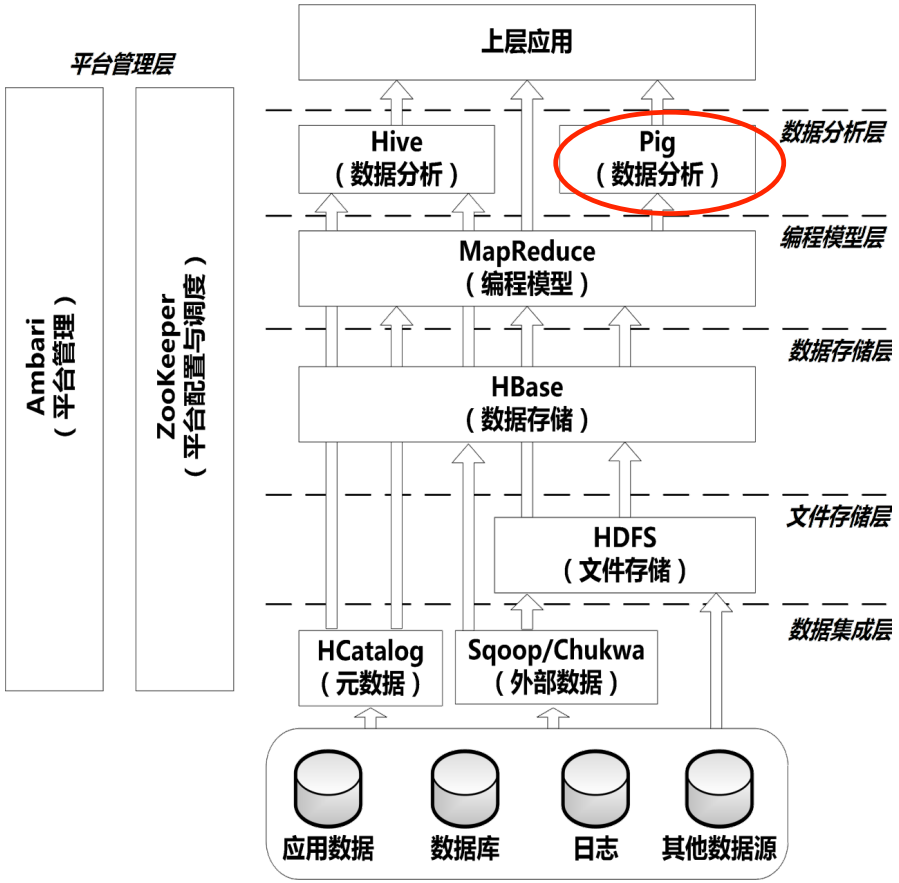

# C10. Pig

刘军（liujun@bupt.edu.cn）
北京邮电大学 数据科学中心

# Pig的位置



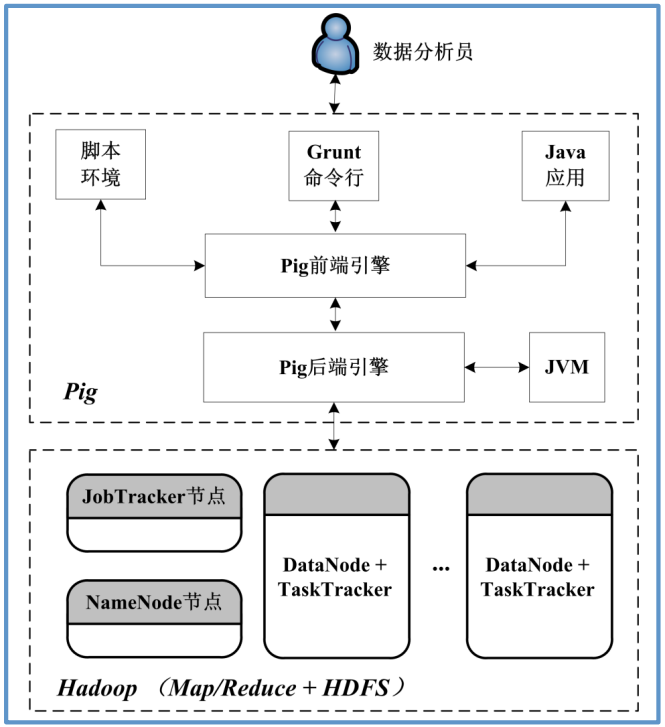

- 数据分析员 vs. 程序员
  - **数据抽象**
  - **操作方式**
  - **执行环境**

# Pig的定义

- 来源： Christopher Olston, Benjamin Reed, Utkarsh Srivastava, et al., Yahoo!, "Pig Latin: A Not-so-foreign Language for Data Processing", ACM SIGMOD, 2008.
- Why Pig ？（http://pig.apache.org/）
  - Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.
  - Pig's language layer currently consists of a textual language called Pig Latin.

# Pig架构与组件

- 客户端应用，可独立于集群
  - pig.properties文件中的参数
    fs.default.name=hdfs://hdfs_host
    /mapred.job.tracker=hadoop_host:8021




- **三种运行方式**
  - Grunt命令行
  - 脚本文件
  - 应用程序
- **前端引擎**：
  - 负责对Pig Latin语句进行语法检测、语句解析、逻辑检测和代码优化等任务
  - 将命令语句转化为对应的逻辑计划（Logical Plan），消除使用三种接口输入命令语句所造成的表达差异。
- **后端引擎**：
  - 以前端引擎输出的逻辑计划为输入，将其转换为特定运行环境可运行的代码
    - 本地JVM运行环境：这种方式下所有输入文件和命令执行都在节点本地执行
    - Hadoop集群运行环境：后端引擎将逻辑计划转换为一系列MapReduce作业，然后在Hadoop集群中运行

# Pig Latin语言

- Pig Latin语言是使用Pig进行数据分析的核心
  - 一种脚本语言
  - 基于Pig Latin编写的程序由一系列的数据操作（operation）或数据变换（transformation）组成
  - 指令由Pig执行环境进行解释和翻译，转换成一系列MapReduce作业执行
  - 关键点：
    - ✓ 语法
    - ✓ 数据类型
    - ✓ 运算符
    - ✓ 内置函数
    - ✓ 自定义函数

# Pig Latin操作命令（1）- 关系操作

- Pig中的所有任务执行都是通过操作命令体现的，分为：
  - 关系操作
  - 诊断操作
  - UDF操作
  - 系统操作
- 关系操作（Relational Operator）
  - 读写数据和分析数据的操作集合
  - 每一个关系操作语句产生的结果称为“关系”（relation），可以类比为数据库中的一张表

| 类别 | 类型 | 操作 | 说明 |
|---|---|---|---|
| 关系操作 | 加载与存储 | load | 从文件系统或其他存储设备中加载数据存入关系 |
| | | store | 将关系数据存入文件系统或其他存储设备中 |
| | | dump | 将关系操作输出到控制台 |
| | 过滤 | filter | 从关系中过滤掉不需要的行 |
| | | distinct | 从关系中删除重复的行 |
| | | foreach...generate | 在关系中增加或删除字段 |
| | | stream | 使用外部程序对关系进行转换 |
| | | sample | 从关系中随机抽取数据样本 |
| | 分组与连接 | join | 将多个关系进行连接 |
| | | cogroup | 在多个关系中进行数据分组 |
| | | group | 在一个关系中进行数据分组 |
| | | cross | 对多个关系进行交叉乘积运算 |
| | 排序 | order | 根据一个或多个字段对关系进行排序 |
| | | limit | 限制关系的输出元组个数 |
| | 合并与分割 | union | 合并多个关系 |
| | | split | 把一个关系分割为多个关系 |

# Pig Latin操作命令（2）- 诊断操作和UDF操作

- 诊断操作
  - 用于对运行环境的执行情况进行显示

| 类别 | 类型 | 操作 | 说明 |
| --- | --- | --- | --- |
| 诊断操作 | 描绘关系 | describe | 输出关系的模版（schema） |
| | 执行计划 | explain | 输出逻辑和物理执行计划 |
| | | illustrate | 使用生成的输入数据子集显示逻辑计划的示例结果 |

- UDF操作
  - 用于支持用户自定义函数的实现

| 类别 | 类型 | 操作 | 说明 |
| --- | --- | --- | --- |
| UDF操作 | 注册 | register | 在运行环境中注册一个JAR文件 |
| | 定义 | define | 为宏、自定义函数、streaming脚本或命令定义别名 |
| | 导入 | import | 从单独的文件中导入宏到脚本中 |

# Pig Latin操作命令（3）- 系统操作

- 系统操作：
  - 文件系统操作：用于对Hadoop文件系统的文件和目录进行管理
  - 作业系统操作：用于控制脚本程序转换后产生的MapReduce作业

| 类别 | 类型 | 操作 | 说明 |
|---|---|---|---|
| 系统操作 | 文件系统 | cat | 输出一个或多个文件的内容 |
| | | cd | 进入某个目录 |
| | | copyFromLocal | 将本地文件或目录复制到Hadoop文件系统 |
| | | copyToLocal | 将文件或目录从Hadoop文件系统复制到本地 |
| | | cp | 把一个文件或目录复制到另一个目录下 |
| | | fs | 进入Hadoop文件系统的命令交互环境 |
| | | ls | 输出文件列表信息 |
| | | mkdir | 创建目录 |
| | | mv | 移动文件或目录 |
| | | pwd | 输出当前所在目录的路径 |
| | | rm | 删除文件或目录 |
| | | rmf | 强行删除文件或目录 |
| | 作业系统 | kill | 终止某个MapReduce作业 |
| | | exec | 在一个新的Grunt Shell中以批处理模式执行脚本 |
| | | help | 显示可用的命令和选项的帮助信息 |
| | | quit | 退出交互环境 |
| | | run | 在当前Grunt Shell中执行脚本 |
| | | set | 设置Pig选项和MapReduce作业属性 |
| | | sh | 在Grunt环境中执行shell指令 |

# Pig实例（1）- 源数据

- /data/log.txt

| IP | Host | SP | Traffic |
|---|---|---|---|
| 125.39.127.20 | b8.photo.store.qq.com | qq | 10 |
| 125.39.127.21 | b5.photo.store.qq.com | qq | 15 |
| 125.39.127.30 | b25.photo.store.qq.com | qq | 25 |
| 125.39.127.21 | b40.photo.store.qq.com | qq | 15 |
| 125.39.127.30 | b32.photo.store.qq.com | qq | 10 |
| 125.39.127.30 | s6.photo.store.qq.com | qq | 15 |
| 122.228.243.250 | q.i02.wimg.taobao.com | taobao | 100 |

- 分析目标：每个IP访问qq.com的不同host的流量占总流量百分比

# Pig实例（2）- 加载（LOAD）

将数据加载到pig中

```
grunt> records = LOAD '/data/log.txt' AS (ip:chararray, host:chararray, sp:chararray, traffic:int);

grunt> describe records;
records:{ip: chararray,host: chararray,sp: chararray, traffic: int}

grunt> dump records;
```

```
(125.39.127.20,b8.photo.store.qq.com,qq,10)
(125.39.127.21,b5.photo.store.qq.com,qq,15)
(125.39.127.30,b25.photo.store.qq.com,qq,25)
(125.39.127.21,b40.photo.store.qq.com,qq,15)
(125.39.127.30,b32.photo.store.qq.com,qq,10)
(125.39.127.30,s6.photo.store.qq.com,qq,15)
(122.228.243.250,q.i02.wimg.taobao.com,taobao,100)
```

records

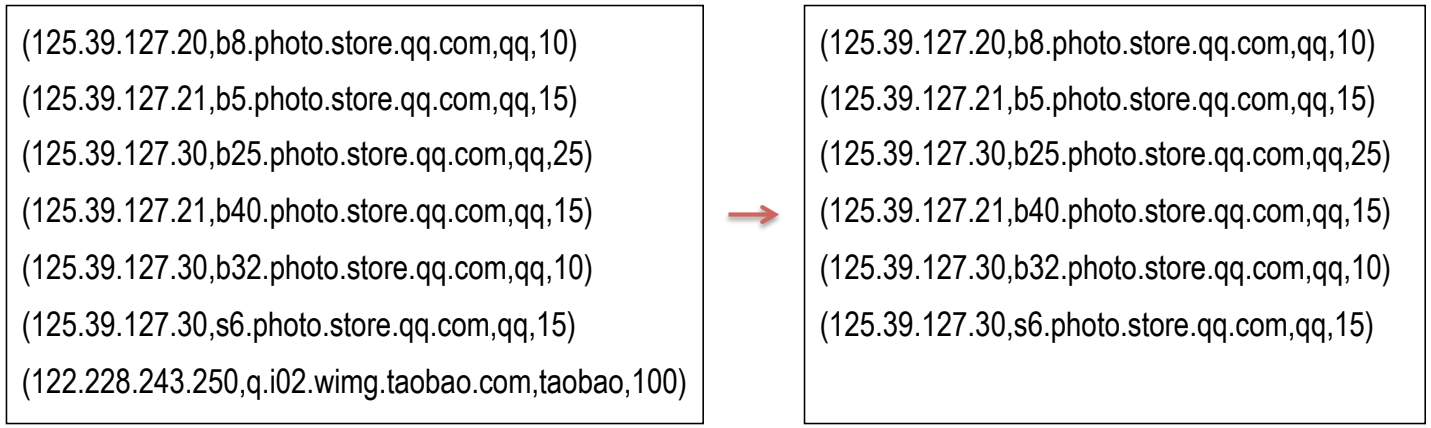
# Pig实例（3）- 过滤（FILTER）

只选取host包含qq的记录

```
grunt> filter_records = FILTER records BY sp MATCHES '.*qq.*';

grunt> describe filter_records;
filter_records: {ip: chararray,host: chararray,sp: chararray, traffic: int}

grunt> dump filter_records;
```




records　　filter_records

# Pig实例（4）- 排序（ORDER）

按IP排序

```
grunt> order_records = ORDER filter_records BY ip;

grunt> describe order_records;
order_records: {ip: chararray,host: chararray,sp: chararray, traffic: int}

grunt> dump order_records;
```

| filter_records |
| --- |
| (125.39.127.20,b8.photo.store.qq.com,qq,10) |
| (125.39.127.21,b5.photo.store.qq.com,qq,15) |
| (125.39.127.30,b25.photo.store.qq.com,qq,25) |
| (125.39.127.21,b40.photo.store.qq.com,qq,15) |
| (125.39.127.30,b32.photo.store.qq.com,qq,10) |
| (125.39.127.30,s6.photo.store.qq.com,qq,15) |

| order_records |
| --- |
| (125.39.127.20,b8.photo.store.qq.com,qq,10) |
| (125.39.127.21,b5.photo.store.qq.com,qq,15) |
| (125.39.127.21,b40.photo.store.qq.com,qq,15) |
| (125.39.127.30,b25.photo.store.qq.com,qq,25) |
| (125.39.127.30,b32.photo.store.qq.com,qq,10) |
| (125.39.127.30,s6.photo.store.qq.com,qq,15) |

# Pig实例（5）- 分组（GROUP）

按IP分组

```
grunt> group_records = GROUP order_records BY ip;

grunt> describe group_records;
group_records: {group: chararray, order_records: {(ip: chararray, host: chararray,
sp: chararray, traffic: int)}}

grunt> dump group_records;
```

```
(125.39.127.20,b8.photo.store.qq.com,qq,10)
(125.39.127.21,b5.photo.store.qq.com,qq,15)
(125.39.127.21,b40.photo.store.qq.com,qq,15)
(125.39.127.30,b25.photo.store.qq.com,qq,25)
(125.39.127.30,b32.photo.store.qq.com,qq,10)
(125.39.127.30,s6.photo.store.qq.com,qq,15)
```

order_records

```
(125.39.127.20,{(125.39.127.20,b8.photo.store.qq.com,qq,10)})
(125.39.127.21,{(125.39.127.21,b5.photo.store.qq.com,qq,15),
   (125.39.127.21,b40.photo.store.qq.com,qq,15)})
(125.39.127.30,{(125.39.127.30,b25.photo.store.qq.com,qq,
   25),(125.39.127.30,b32.photo.store.qq.com,qq,10),
   (125.39.127.30,s6.photo.store.qq.com,qq,15)})
```

group_records

# Pig实例（6）- 循环处理（FOREACH）

对每个IP计算总流量

```
grunt> count_records = FOREACH group_records GENERATE group, COUNT(order_records.ip) as count,
 SUM(order_records.traffic) as sumTraffic;

grunt> describe count_records;
count_records: {group: chararray, count: long, sumTraffic: long}

grunt> dump count_records;
```

```
(125.39.127.20,{(125.39.127.20,b8.photo.store.qq.com,qq,10)})
(125.39.127.21,{(125.39.127.21,b5.photo.store.qq.com,qq,15),
   (125.39.127.21,b40.photo.store.qq.com,qq,15)})
(125.39.127.30,{(125.39.127.30,b25.photo.store.qq.com,qq,25),
   (125.39.127.30,b32.photo.store.qq.com,qq,10),
   (125.39.127.30,s6.photo.store.qq.com,qq,15)})
```

group_records

```
(125.39.127.20,1,10)
(125.39.127.21,2,30)
(125.39.127.30,3,50)
```

count_records

# Pig实例（7） - 联结（JOIN）

将order_records和count_records按照ip和group进行联结

```
grunt> join_records = JOIN order_records BY ip, count_records BY group;
grunt> describe join_records;
join_records: {order_records::ip: chararray, order_records ::host: chararray,
order_records ::sp: chararray, order_records::traffic:int, count_records::group:chararray,
count_records::count: long, count_records::sumTraffic: long}
grunt> dump join_records;
```

(125.39.127.20,b8.photo.store.qq.com,qq,10)
(125.39.127.21,b5.photo.store.qq.com,qq,15)
(125.39.127.21,b40.photo.store.qq.com,qq,15)
(125.39.127.30,b25.photo.store.qq.com,qq,25)
(125.39.127.30,b32.photo.store.qq.com,qq,10)
(125.39.127.30,s6.photo.store.qq.com,qq,15)

(125.39.127.20,1,10)
(125.39.127.21,2,30)
(125.39.127.30,3,50)

→

(125.39.127.20,b8.photo.store.qq.com,qq,10,125.39.127.20,1,10)
(125.39.127.21,b5.photo.store.qq.com,qq,15,125.39.127.21,2,30)
(125.39.127.21,b40.photo.store.qq.com,qq,15,125.39.127.21,2,30)
(125.39.127.30,b25.photo.store.qq.com,qq,25,125.39.127.30,3,50)
(125.39.127.30,b32.photo.store.qq.com,qq,10,125.39.127.30,3,50)
(125.39.127.30,s6.photo.store.qq.com,qq,15,125.39.127.30,3,50)

join_records

# Pig实例（8）- 结果

每个IP访问qq.com的host的流量占总流量比例

```
grunt> end_records = FOREACH join_records GENERATE order_records::ip, count_records::count, order_records::host,
 order_records::sp, order_records::traffic, (double)order_records::traffic/(double)count_records::sumTraffic as percent:double;

grunt> describe end_records;

end_records：{order_records::ip: chararray,count_records::count: long, order_records ::host: chararray, order_records::sp:
 chararray, percent: double}

grunt> dump end_records;
```

(125.39.127.20,b8.photo.store.qq.com,qq,10,125.39.127.20,1,10)
(125.39.127.21,b5.photo.store.qq.com,qq,15,125.39.127.21,2,30)
(125.39.127.21,b40.photo.store.qq.com,qq,15,125.39.127.21,2,30)
(125.39.127.30,b25.photo.store.qq.com,qq,25,125.39.127.30,3,50)
(125.39.127.30,b32.photo.store.qq.com,qq,10,125.39.127.30,3,50)
(125.39.127.30,s6.photo.store.qq.com,qq,15,125.39.127.30,3,50)

join_records

(125.39.127.20,1,b8.photo.store.qq.com,qq,10,1.0)
(125.39.127.21,2,b5.photo.store.qq.com,qq,15,0.5)
(125.39.127.21,2,b40.photo.store.qq.com,qq,15,0.5)
(125.39.127.30,3,b25.photo.store.qq.com,qq,25,0.5)
(125.39.127.30,3,b32.photo.store.qq.com,qq,10,0.2)
(125.39.127.30,3,s6.photo.store.qq.com,qq,15,0.3)

end_records

# Pig Latin用户自定义函数

- 用户自定义函数（User Defined Function，UDF）
  - 支持使用三种语言开发用户自定义函数
    - ✓ Java、Python和JavaScript
  - Java编写的自定义函数功能最强大，可以覆盖数据处理的全过程
    - ✓ 编写扩展基本处理函数类的自定义类代码
    - ✓ 使用UDF操作的register命令注册包含自定义类的JAR包
    - ✓ 在代码中使用自定义函数类

# Pig Latin用户自定义函数示例

- 需要找出域名中包含s6的记录
- 扩展过滤函数
  - 所有的过滤函数都是FilterFunc的子类
  - 需要扩展FilterFunc子类，并实现类的exec()接口
- 步骤：
  - 编译并打包到JAR文件filterFunc.jar中
  - 在Pig中注册此JAR文件

    grunt> register filterFunc.jar;
  - 调用此自定义过滤函数

    grunt> s6QQLog = filter log by
    com.example.pig.lss6(host);

    grunt> dump s6QQLog;

    (125.39.127.30,3,s6.photo.store.qq.com,qq,15,0.3)

```
1:  public class lss6 extends FilterFunc {
2:    public Boolean exec(Tuple tuple) {
3:      if (tuple == null || tuple.size() == 0) {
4:        return false;
5:      }
6:      try{
7:        Object object = tuple.get(0);
8:        if (object==null){
9:          return false;
10:       }
11:       String str = (String) object;
12:       return str.index("s6")>=0?true:false;
13:     }catch(ExecException e){
14:       throw new IOException；
15:     }
16:   }
17: }
```


刘军
北京邮电大学 信通院 宽带网络监控教研中心
北邮本部 明光楼七层
邮件地址：liujun@bupt.edu.cn
新浪微博：北邮刘军
电　　话：010-62283742