Hortonworks

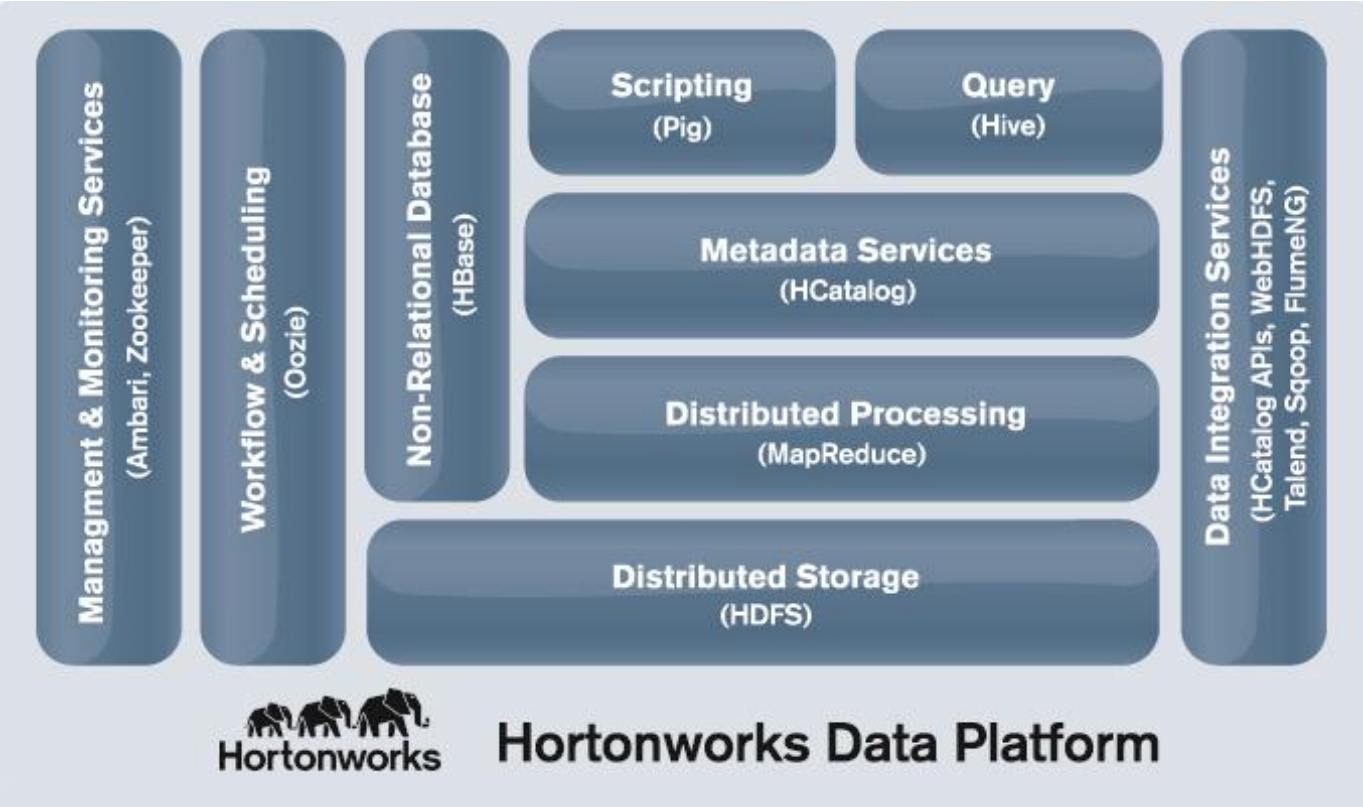
Apache Pig 性能优化

戴建勇
Hortonworks
daijy@hortonworks.com

Hortonworks简介

- **Yahoo! Cloud Computing Group**
- **Hadoop最早开发团队**
- **2011年7月成立**
- **从事Hadoop整个生态系统软件的开发**
  - 基于Apache
- **拥有数量众多的Hadoop Committer, 主导Apache Hadoop开发, 发行**

Hortonworks Data Platform



- ✓ 降低转换和使用风险
- ✓ 降低管理成本
- ✓ 轻松集成现有系统

Hortonworks Data Platform

- **基于Apache软件发行版**
- **完全免费http://hortonworks.com/download/**
- **有完整的技术支持和培训**

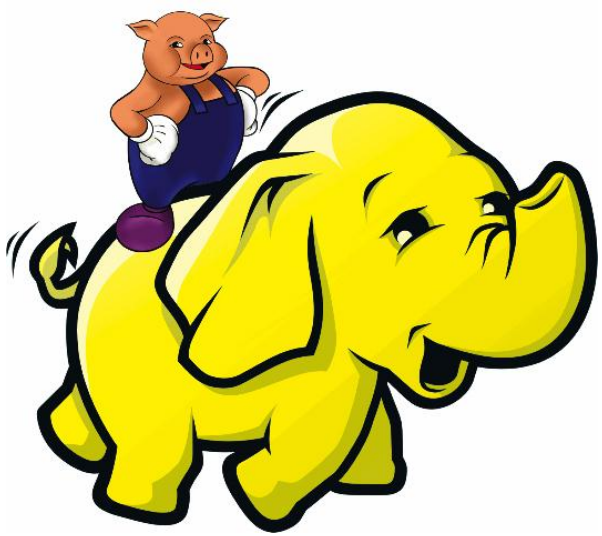
自我介绍

- **Member of Technical Staff in Hortonworks**
- **从事Apache Pig开发3年以上**
- **Apache Pig Committer / PMC Chair**
- **Apache HCatalog Committer / PMC member**

什么是Apache Pig

Pig Latin, 类SQL数据处理语言

在Hadoop上运行的Pig Latin执行引擎



Pig-latin-cup pic from http://www.flickr.com/photos/frippy/2507970530/

Pig-latin例子

- 查询: 所有被20到29岁网民访问的网址列表

```
USERS = load ‘users’ as (uid, age);

USERS_20s = filter USERS by  age >= 20 and age <= 29;

PVs = load ‘pages’ as (url, uid, timestamp);

PVs_u20s  = join USERS_20s by uid, PVs by uid;
```

Pig vs Hadoop

- **更快的开发**
  - 更少的代码
  - 常见操作的充分优化

Pic courtesy http://www.flickr.com/photos/shutterbc/471935204/

In Pig

```
Users = load ‘users’ as (name, age);
Fltrd = filter Users by
        age >= 18 and age <= 25;
Pages = load ‘pages’ as (user, url);
Jnd = join Fltrd by name, Pages by user;
Grpd = group Jnd by url;
Smmd = foreach Grpd generate group,
       COUNT(Jnd) as clicks;
Srtd = order Smmd by clicks desc;
Top5 = limit Srtd 5;
store Top5 into ‘top5sites’;
```

In Map Reduce

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
            new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
            new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

Pig vs Hive

- **过程化语言**
- **灵活性**
  - Schema不是必须的
  - 可扩充性
- **Pig的定位**

数据采集

数据加工
Pig

数据仓库
Hive

Pig用户和社区

- Pig主要用户
  - Yahoo!: 90%以上的MapReduce作业是Pig生成的
  - Twitter: 80%以上的MapReduce作业是Pig生成的
  - Linkedin: 大部分的MapReduce作业是Pig生成的
  - 其他主要用户: Salesforce, Nokia, AOL, comScore
- Pig的主要开发者
  - Hortonworks
  - Twitter
  - Yahoo!
  - Cloudera

Pig工具

- Piggybank
  - Pig的官方函数库
  - 主要由Pig用户维护
  - 目前随Pig一起发行
- Elephant bird: Twitter的Pig函数库
- DataFu: Linkedin的Pig函数库
- Ambros: Twitter的Pig作业监控系统
- Mortardata: 基于云的Pig集群管理系统

Pig简史

- 2008, “Pig Latin: A Not-So-Foreign Language for Data Processing”, SIGMOD, Chris Olson
- 2008, 源码提交Apache, Pig 0.1 release
- 2008, 成为Hadoop子项目
- 2010, 成为Apache一级子项目

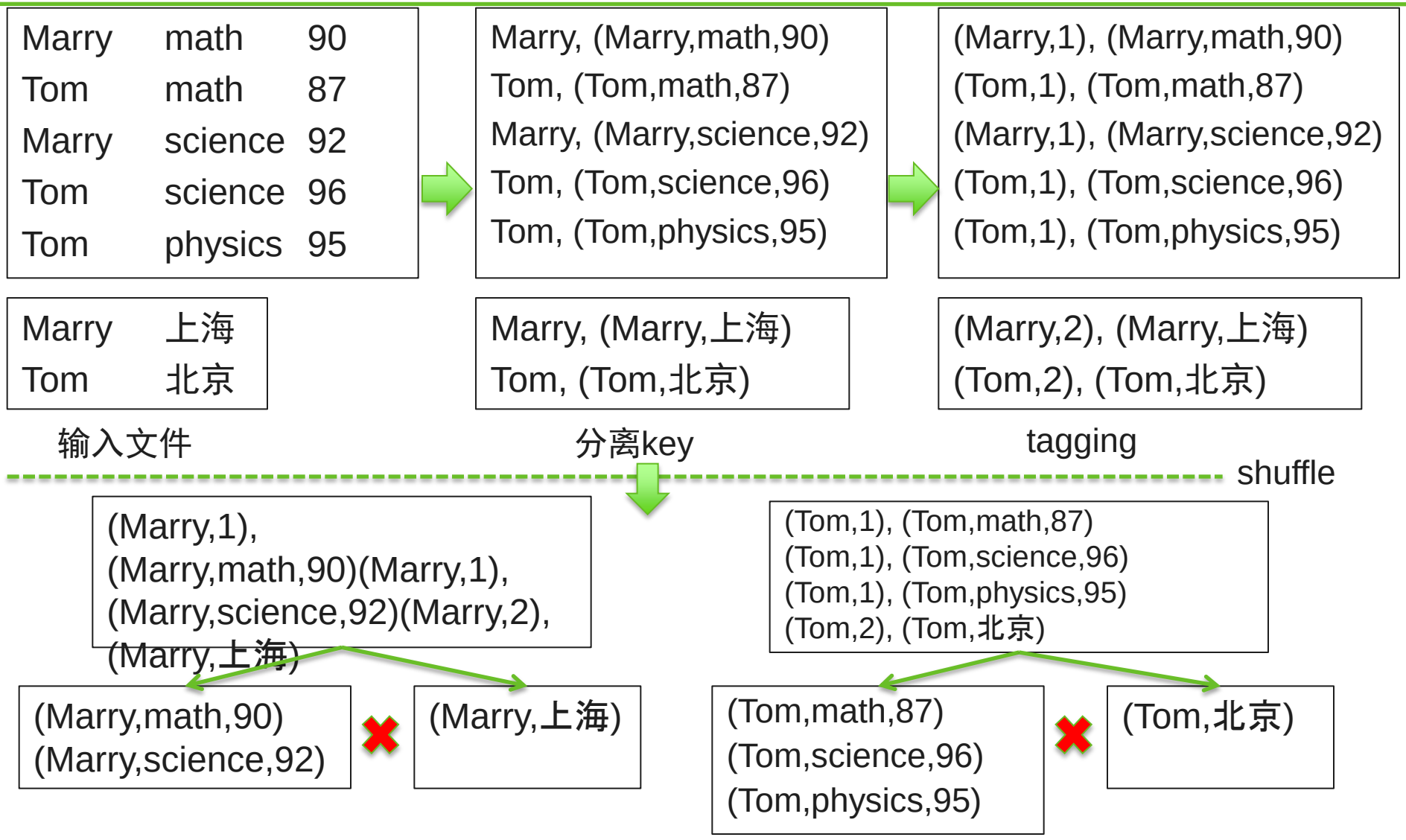
Pig版本

- 2010/12—0.8.0
  - Python UDF, Scalar, Custom partitioner
- 2011/7—0.9.0
  - Pig Embedding
- 2012/4—0.10.0
  - Boolean, Jruby
- ??—0.11.0
  - Cube/Rank, Datetime

Pig对常见MapReduce操作的实现

- Order by
- Join
  - Hash Join
  - Replicated Join
  - Skewed Join
  - Merge Join

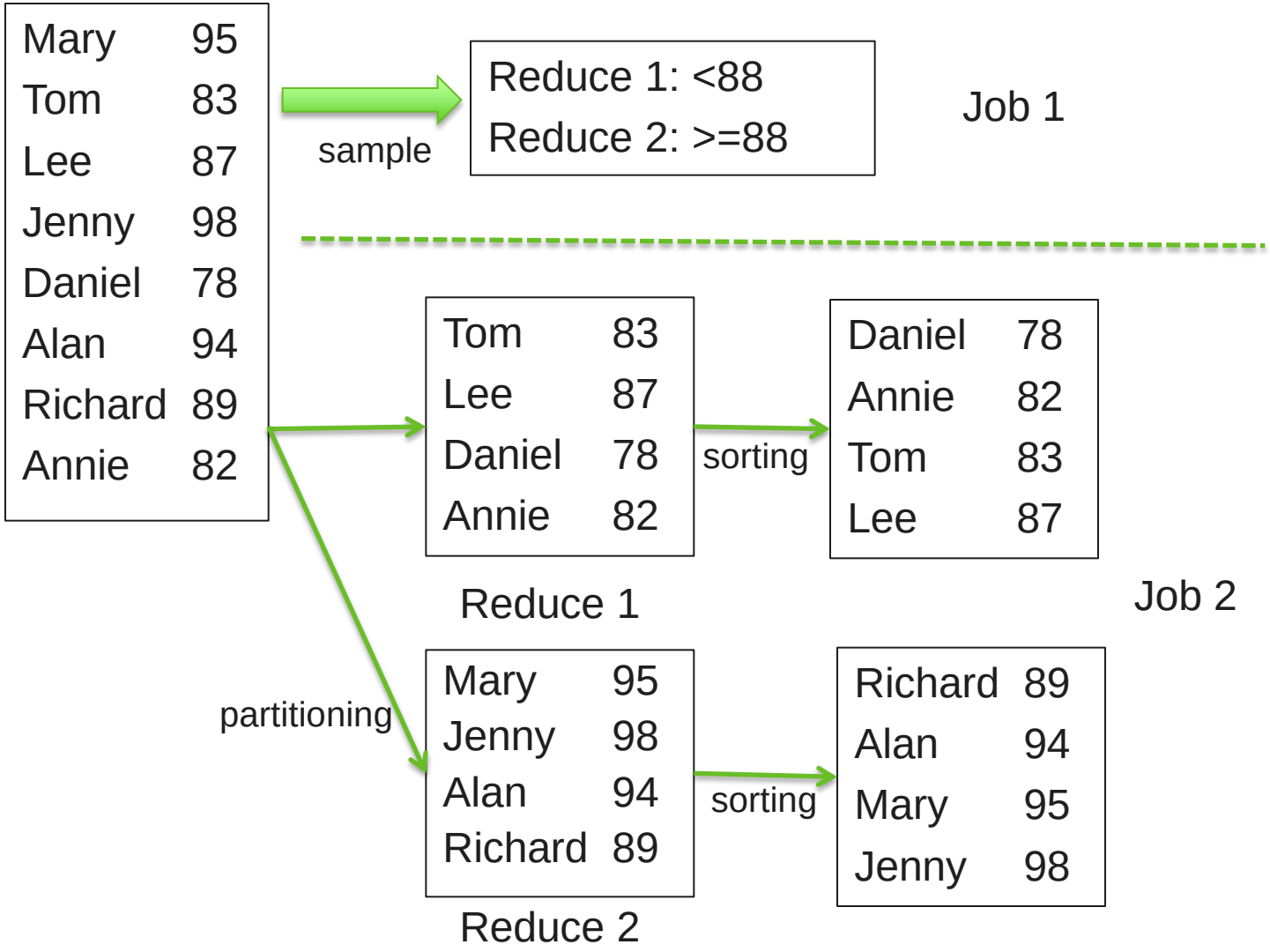
Hash Join的实现



Order by的实现

- 分布式实现
  - 每个Reduce实现一部分的排序
  - 全局Total Order
- Pig实现
  - Sample: 确定每个Reduce负责的数据区间
  - Partitioner: 把数据发送到不同的Reduce
  - Reduce: 排序

Order by的实现



Skewed Join的实现

- 解决超大key问题
  - Reduce运行缓慢，极耗内存
  - 思路：把key分配到不同的reduce
- Pig实现
  - Sample: 确定每个Key需要多少Reduce
  - Partitioner: 把数据发送到不同的Reduce
  - Reduce: 把右关系复制到每个Reduce, Reduce得到交叉结果

Merge Join的实现

- 思路
  - 先排序，再做Merge Sort
- Pig实现
  - 左，右关系必须预先排序
  - 右关系建立索引
  - Map Side
    - 通过索引迅速找到右关系开始位置
    - 左，右关系进行Merge

充分利用Combiner

- Combiner
  - 送往reduce之前先进行汇总
  - 减少Map/Reduce之间传送的数据量
- 常用的Pig函数已经进行了Combiner优化
  - COUNT, SUM, AVG, TOP
- 更好的方法
  - Combiner的问题：序列化开销过高
  - Pig 0.10：在map里直接做汇总，取消combiner

基于规则的优化器

- Column pruner
- Push up filter
- Push down flatten
- Push up limit
- Partition pruning
- 全局优化器

Column Pruner

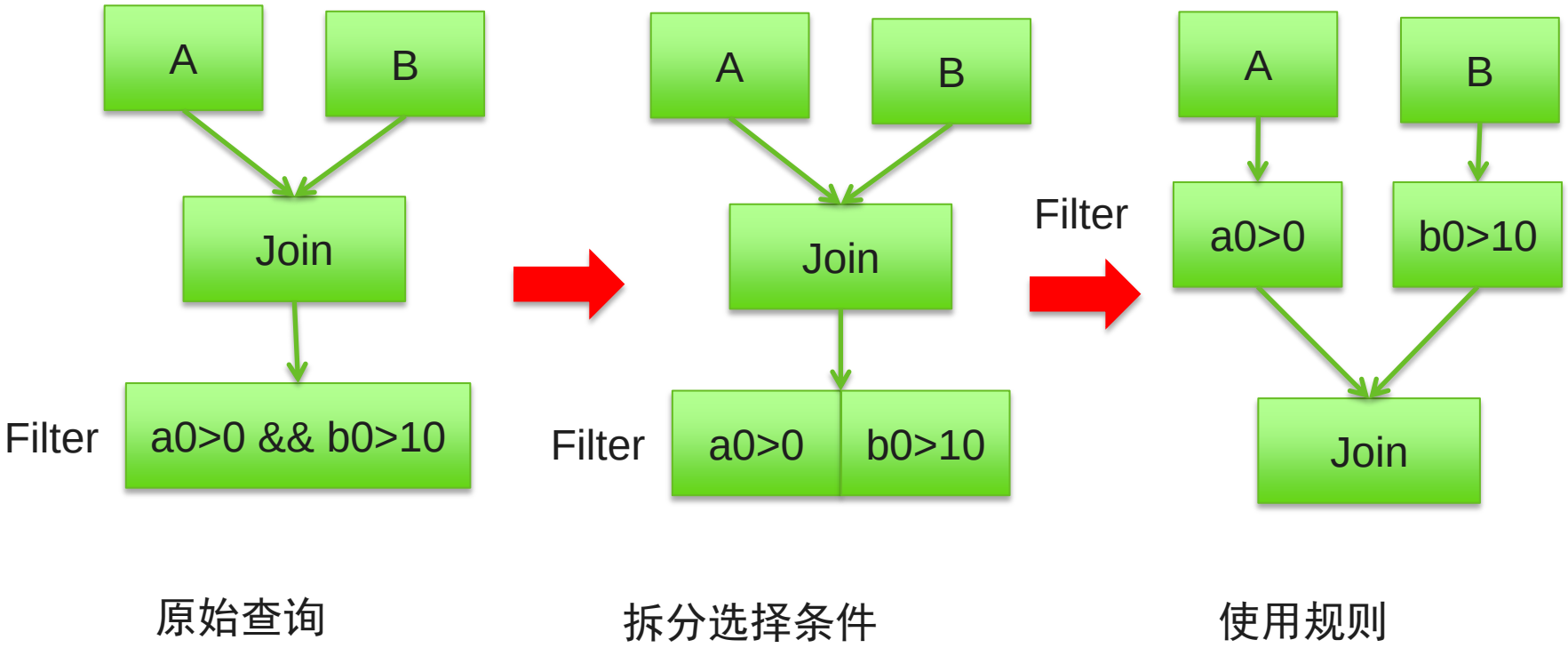
- Pig自动Prune Column

```
A = load ‘input’ as (a0, a1, a2);
B = foreach A generate a0+a1;
C = order B by $0;
Store C into ‘output’;
```

- 某些情况下Pig无法完成自动Column Prune
  - Load语句没有Schema
  - Group by之后有未用到的列
  - 用户可以自行用foreach语句略去不用的列

Push up filter

- Pig使用规则前先进行选择条件拆分

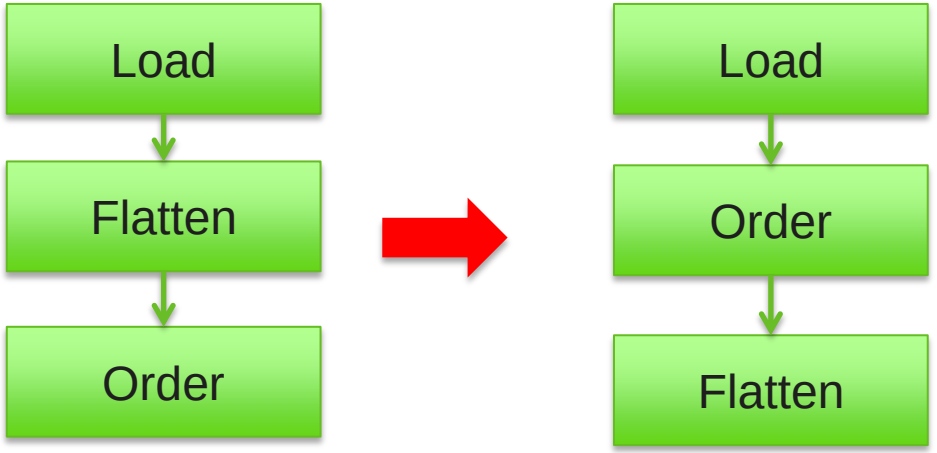


原始查询　　拆分选择条件　　使用规则

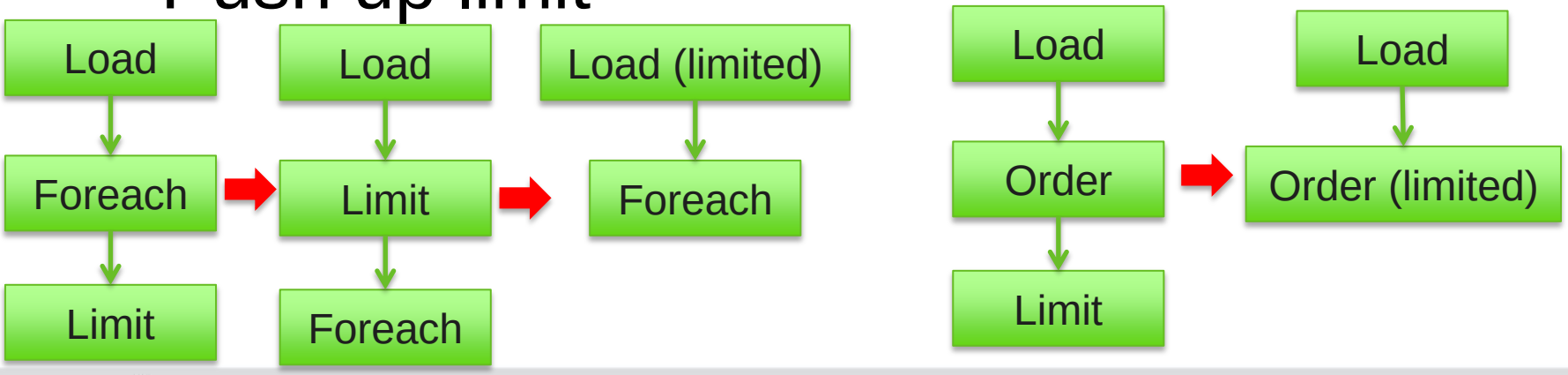
其他push up/down

- Push down flatten

```
A = load ‘input’ as (a0:bag, a1);
B = foreach A generate
flattten(a0), a1;
C = order B by a1;
Store C into ‘output’;
```

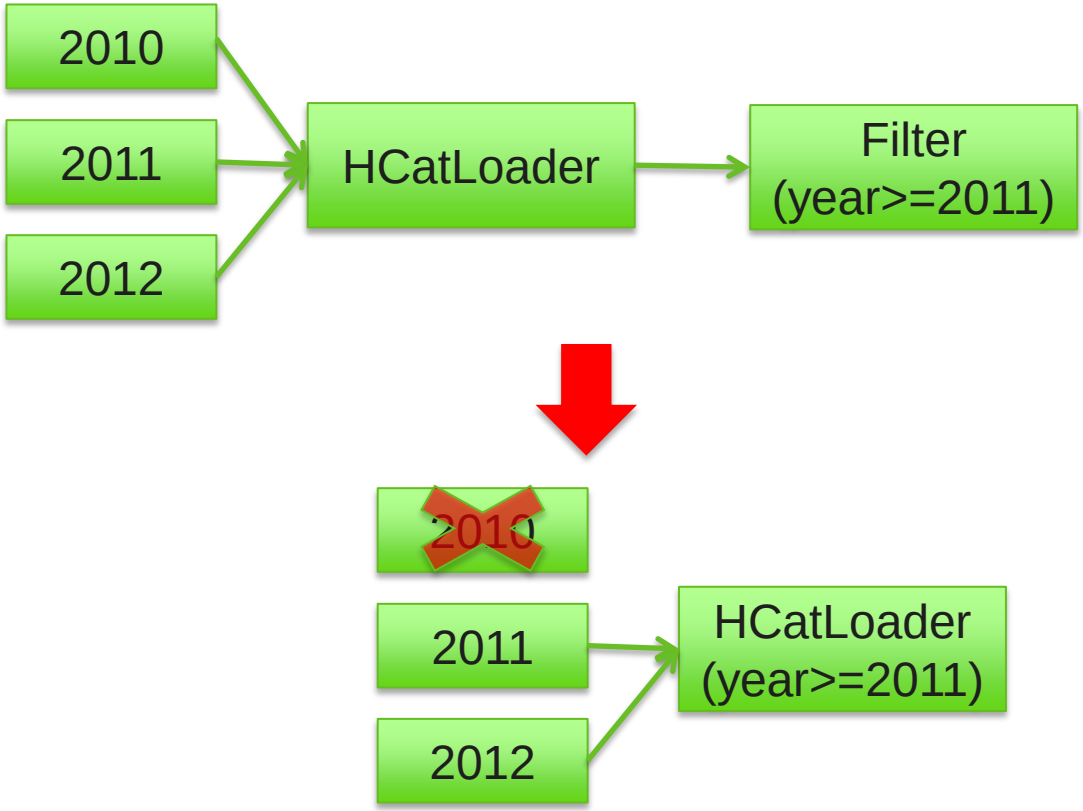


- Push up limit

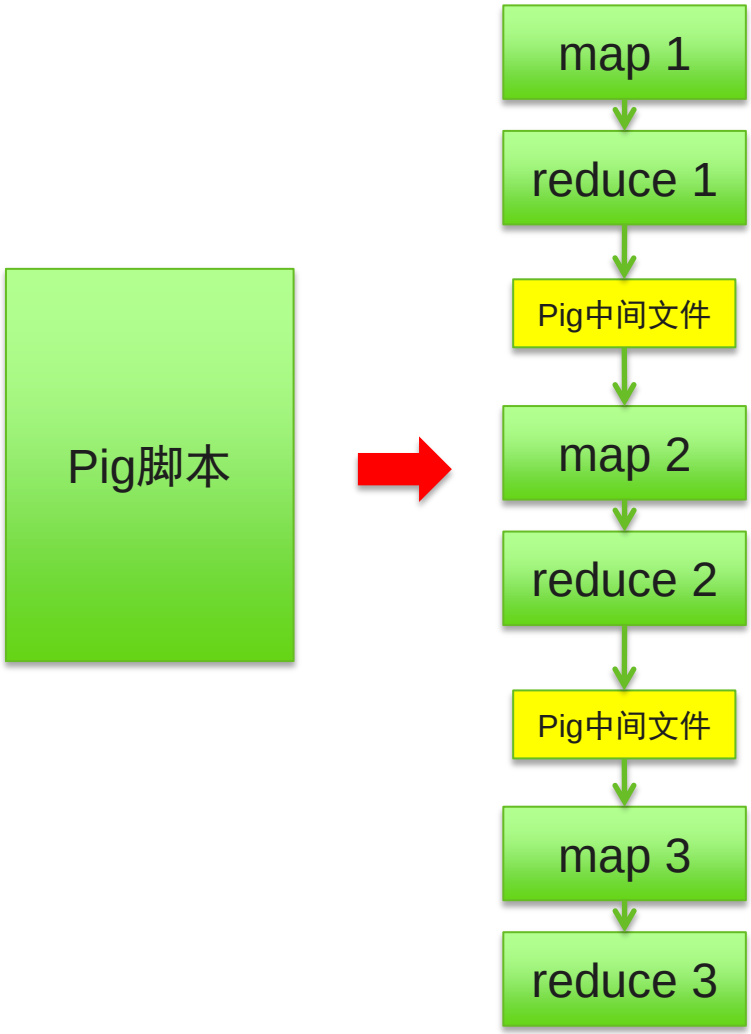


分区pruning

- 略去整个不需要的分区
  - HCatLoader



压缩中间文件



- Map和reduce之间的中间文件
  - Snappy
- 不同mapreduce作业间的中间文件
  - 缺省没有压缩

定制Pig中间文件压缩

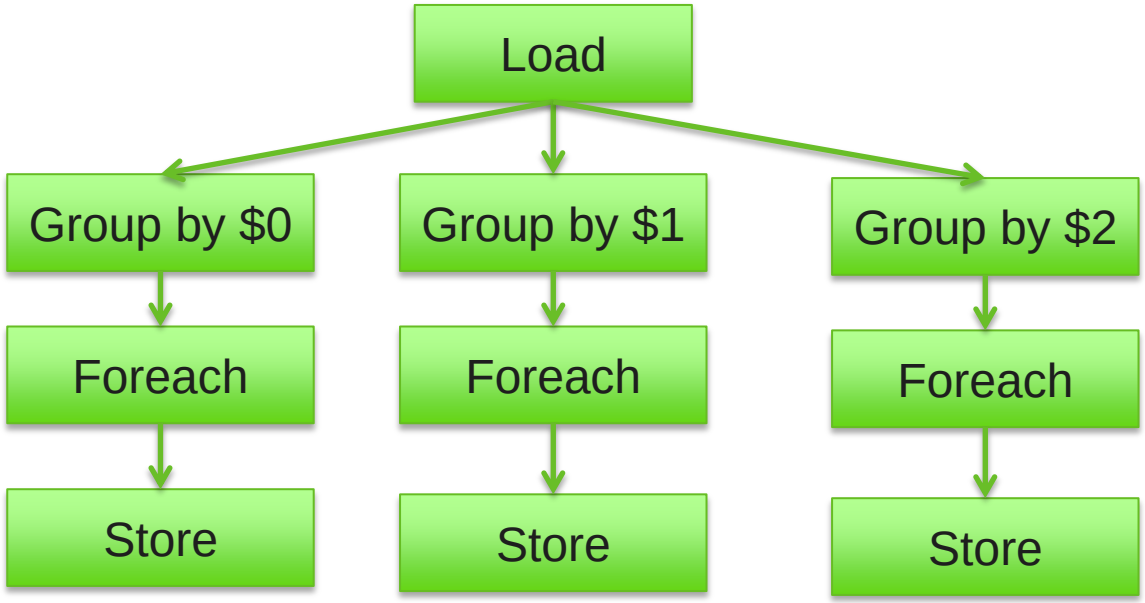
- Pig中间文件缺省不压缩
  - snappy的问题(HADOOP-7990)
  - LZO: 没有Apache许可
- 开启LZO compression
  - 在Hadoop上安装LZO
  - 修改conf/pig.properties

```
pig.tmpfilecompression = true
pig.tmpfilecompression.codec  = lzo
```

  - 开启LZO，我们观察到高达90%的磁盘空间节省和4倍的查询速度提升

合并MapReduce作业

- 合并两个以上的MapReduce作业



  - Pig自动进行作业合并
  - 某些情况下我们需要控制合并粒度：Pig合并了过多的作业

控制合并粒度

- 取消multiquery
  - 命令行参数: -M
- 用“exec”标明作业边界

```
A = load ‘input’;
B0 = group A by $0;
C0 = foreach B0 generate group, COUNT(A);
Store C0 into ‘output0’;
B1 = group A by $1;
C1 = foreach B1 generate group, COUNT(A);
Store C1 into ‘output1’;
exec
B2 = group A by $2;
C2 = foreach B2 generate group, COUNT(A);
Store C2 into ‘output2’;
```

合并输入文件

- 一系列小输入文件
  - Hadoop: 每个输入文件一个map
  - 太多map作业
- Pig自动合并小输入文件

Pig 0.11新特性

- Cube
- Rank
- 新数据类型: Datetime
- JRuby UDF
- 性能优化
  - SchemaTuple优化
  - Local mode优化

Pig的未来

- Low latency查询
  - 利用YARN的新特性
- 性能优化
  - Cost based optimizer
  - 基于编译的后端
  - Hbase的Join优化
- Visualization, 监控
  - 图形界面
  - 更好的Execution Plan显示
  - 监控系统 (Hortonworks Sandbox, Ambari)

参与Pig开发

- 订阅邮件列表
  - user@pig.apache.org, dev@pig.apache.org
- 贡献Patch
  - 从newbie Jira开始
- Pig Committer
  - 在Pig邮件列表和Jira上活跃6个月以上
  - 贡献若干Patch
  - Review其他开发者的Patch

Hortonworks

?