Introduction To Pig

Alan Gates

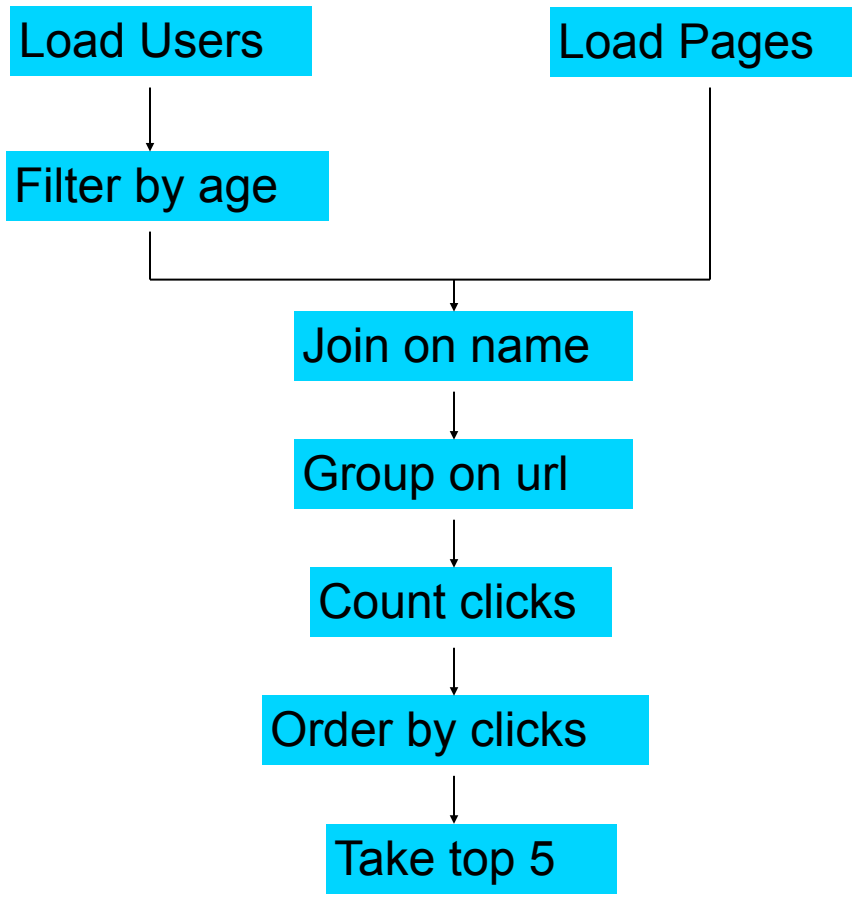
Yahoo!

About Me

- Pig Committer
- Member of Hadoop PMC
- Architect for Pig development group at Yahoo!

Why is Pig Interesting?

- Data
  - User records
  - Pages served
- Question: the 5 pages most visited by users aged 18 - 25.



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.a pache.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.JobC   ontrol;
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.sub  string(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 w   e 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.to  String();
                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,
                OutputColle  ctor<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', first   Comma);
            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 ke y,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            wh ile (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.se tJobName("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.s etJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.add  InputPath(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(IdentityMap   per.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.se  tOutputPath(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(MRE  xample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFi  leOutputFormat.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(SequenceFileOutputF   ormat.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();
    }
}
```

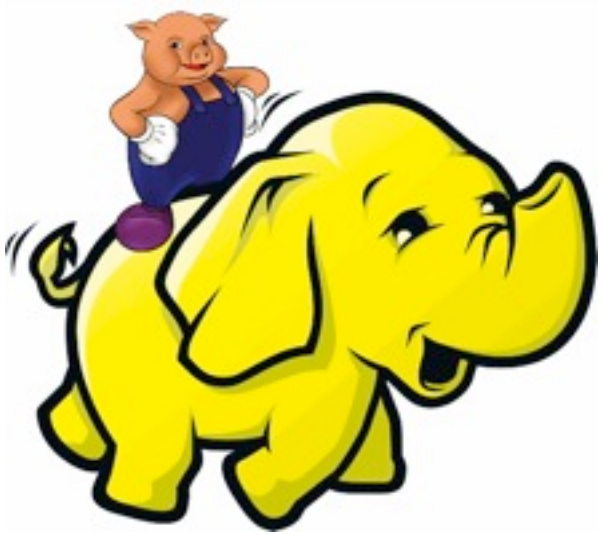

In Pig Latin

```
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';
```


What is Pig?

Pig Latin, a high level data processing language.

An engine that executes Pig Latin locally or on a Hadoop cluster.



Apache SOFTWARE FOUNDATION

A subproject of Apache Hadoop

Comparing Pig and Map Reduce

- Pig
  - radically reduces development time
  - includes standard relational operations (join, sort, etc.) so you don't have to write them yourself
  - copes with Hadoop version and API changes
- While preserving the power of Hadoop
  - user code can be plugged in at almost any point
  - works with structured, semi-structured, and unstructured data
  - no requirement up front to define data schema
  - explicit data flow control

Why not SQL?

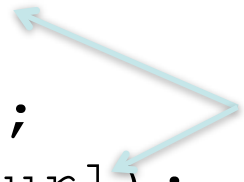
- SQL is spoken almost everywhere
- But
  - SQL requires schema definition up front
  - SQL lets you say what you want, but not how you want it done
  - SQL works on tables, but HDFS has files
  - Writing long involved data processes in SQL is painful (subqueries anyone?)
- Pig Latin seeks to be the native language for data processing in Hadoop

Detailed Example

```
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';
```


Detailed Example

```
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';
```



Detailed Example

```
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';
```


Detailed Example

```
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';
```

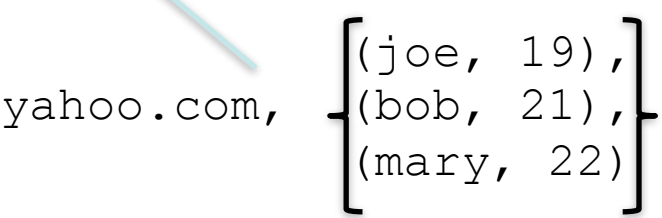
Dynamic typing

Detailed Example

```
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';
```


Detailed Example

```
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';
```



Plugging in Your Code

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


Plugging in Your Code

User can define how data is parsed


```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```

Plugging in Your Code

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


Plugging in Your Code

Loader can provide schema

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


Plugging in Your Code

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


Plugging in Your Code

Column transformation function

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```

Filter function

Aggregate

Plugging in Your Code

```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


Plugging in Your Code

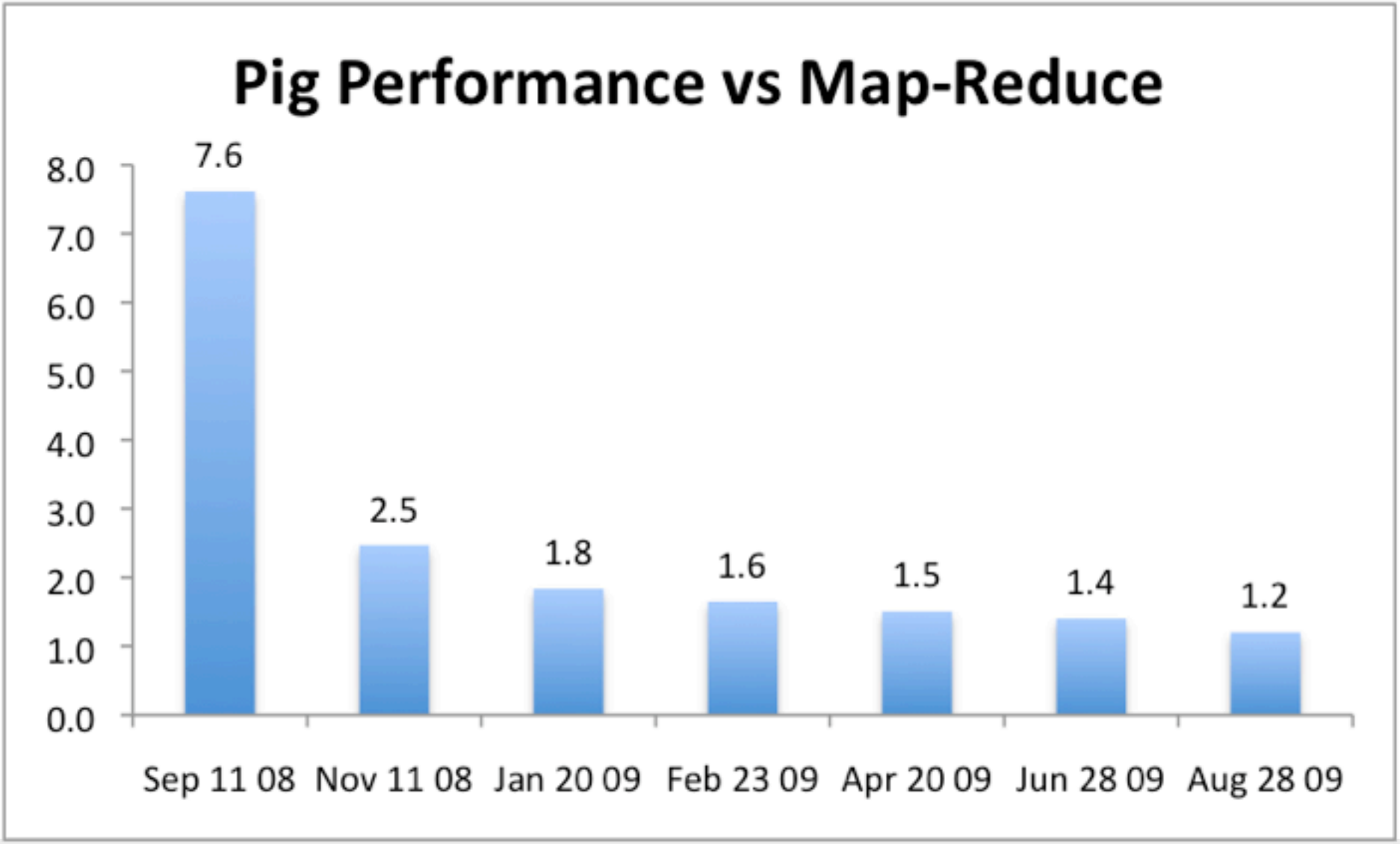
```
A = load 'webpages' using JsonLoader();
B = foreach B generate Canonical(url), content;
C = filter B by NotABot(url);
C = group B by url;
D = foreach C generate group, PageRank(D1);
store D into 'rankedpages' using AvroStorage();
```


User can define how data is stored

Branching Data Flows

```
A = load 'users' as (name, age, gender,
      city, state);
B = filter A by name is not null;
C = group B by age, gender;
D = foreach C generate group, COUNT(B);
store D into 'by_demographic';
gamma = group B by state;
delta = foreach gamma generate group, COUNT(B);
store delta into 'by_state';
```

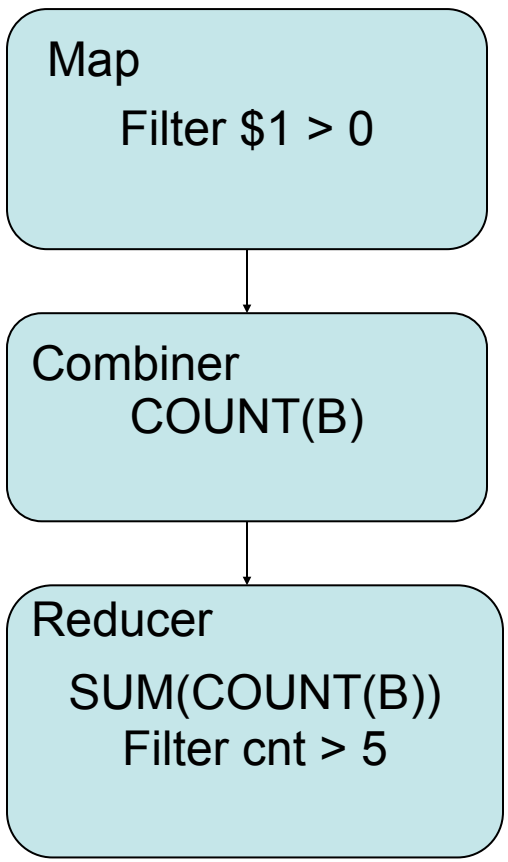

Performance



How it Works

Pig Latin script is translated to a set of operators which are placed in one or more MR jobs and executed.

```
A = load 'myfile';
B = filter A by $1 > 0;
C = group B by $0;
D = foreach C generate
    group, COUNT(B) as cnt;
E = filter D by cnt > 5;
dump E;
```



Components

Job executes on cluster

Pig resides on user machine

User machine

Hadoop Cluster

No need to install anything extra on your Hadoop cluster.

What Users Are Doing With Pig

- Inside Yahoo (based on user interviews)
  - 60% of ad hoc and 40% of production MR jobs
  - Production
    - Examples: search infrastructure, ad relevance
    - Attraction: fast development, extensibility via custom code, protection against Hadoop changes, debugability
  - Ad hoc
    - Examples: user intent analysis
    - Attraction: easy to learn, compact readable code, fast iteration when trying new algorithms, easy for collaboration

What Users Are Doing With Pig

- Outside Yahoo (based on mailing list responses)
  - Processing search engine query logs
    "Pig programs are easier to maintain, and less error-prone than native java programs. It is an excellent piece of work."
  - Image recommendations
    "I am using it as a rapid-prototyping language to test some algorithms on huge amounts of data."
  - Adsorption Algorithm (video recommendations)
  - Hoffman's PLSI implementation
    "The E/M login was implemented in pig in 30-35 lines of pig-latin statements. Took a lot less compared to what it took in implementing the algorithm in mapreduce java. Exactly that's the reason I wanted to try it out in Pig. It took ~ 3-4 days for me to write it, starting from learning pig."

Up and Coming Features

- UDFs in scripting languages that compile to Java byte code (JRuby, Jython, Groovy)
- Making Pig bilingual: SQL
- Performance: faster, faster, faster
- Better integration with other Hadoop subprojects

Getting Your Own Pig

- You can download Pig from http://hadoop.apache.org/pig/
- Pig is included as part of the Cloudera distribution http://www.cloudera.com/hadoop
- Amazon Elastic Map Reduce service allows you to use Hadoop with no hardware investment, and supports Pig: http://aws.amazon.com/elasticmapreduce/

Next Steps

- Read the online documentation: http://hadoop.apache.org/pig/
- On line tutorials
  - From Yahoo, http://developer.yahoo.com/hadoop/tutorial/
  - From Cloudera, http://www.cloudera.com/hadoop-training
- A couple of Hadoop books available that include chapters on Pig, search at your favorite bookstore
- Join the mailing lists:
  - pig-user@hadoop.apache.org for user questions
  - pig-dev@hadoop.apache.com for developer issues

Questions