

PIG

High Level Data Flow Language

Agenda

- Map-Reduce and the need for Pig Latin
- Pig Latin
- Compilation into Map-Reduce
- Implementation
- Comparison with Map-Reduce
- Optimization in Pig



Scale

Scalable due to simpler design

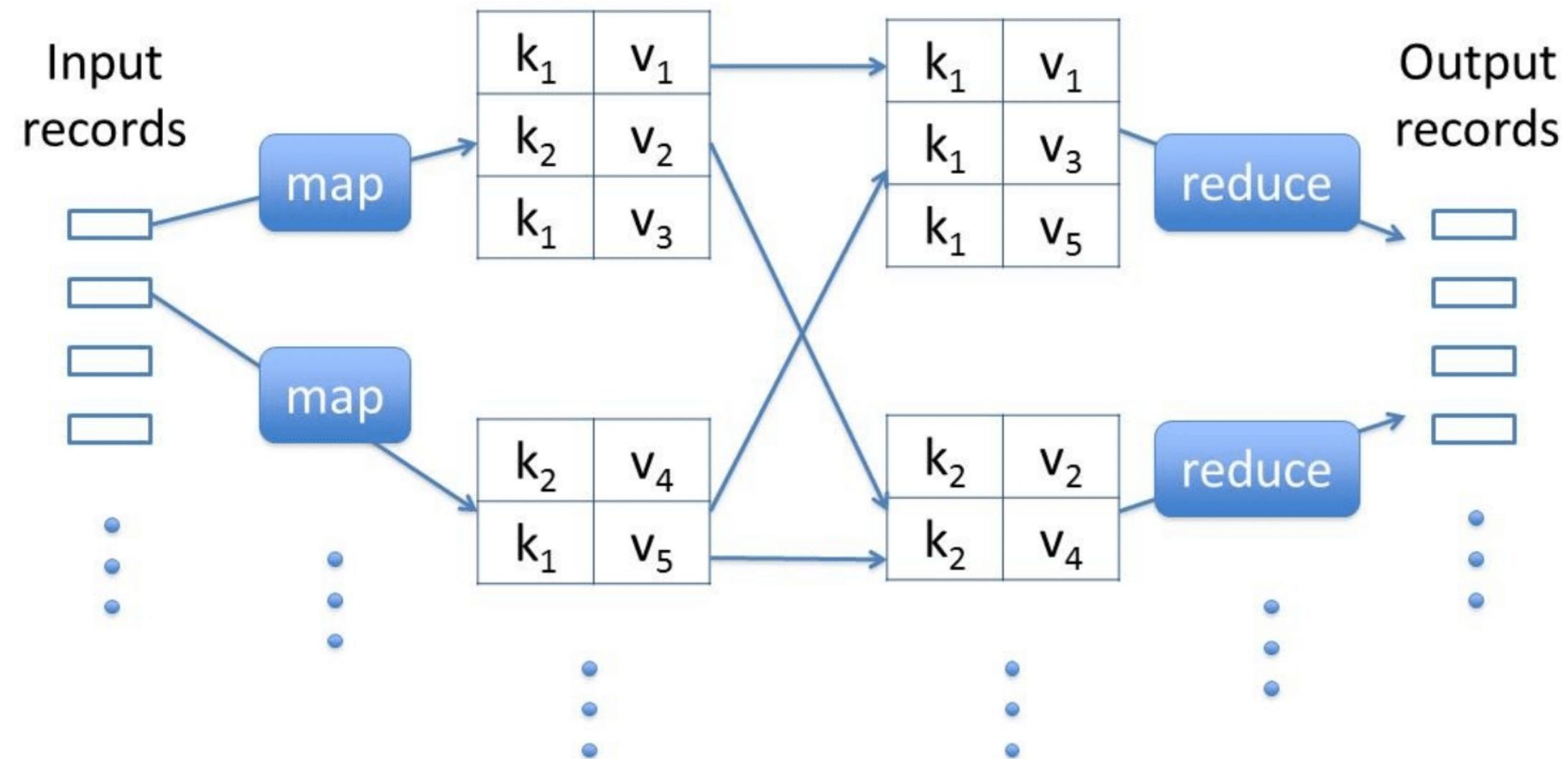
- Only parallelizable operations
- No transactions

\$

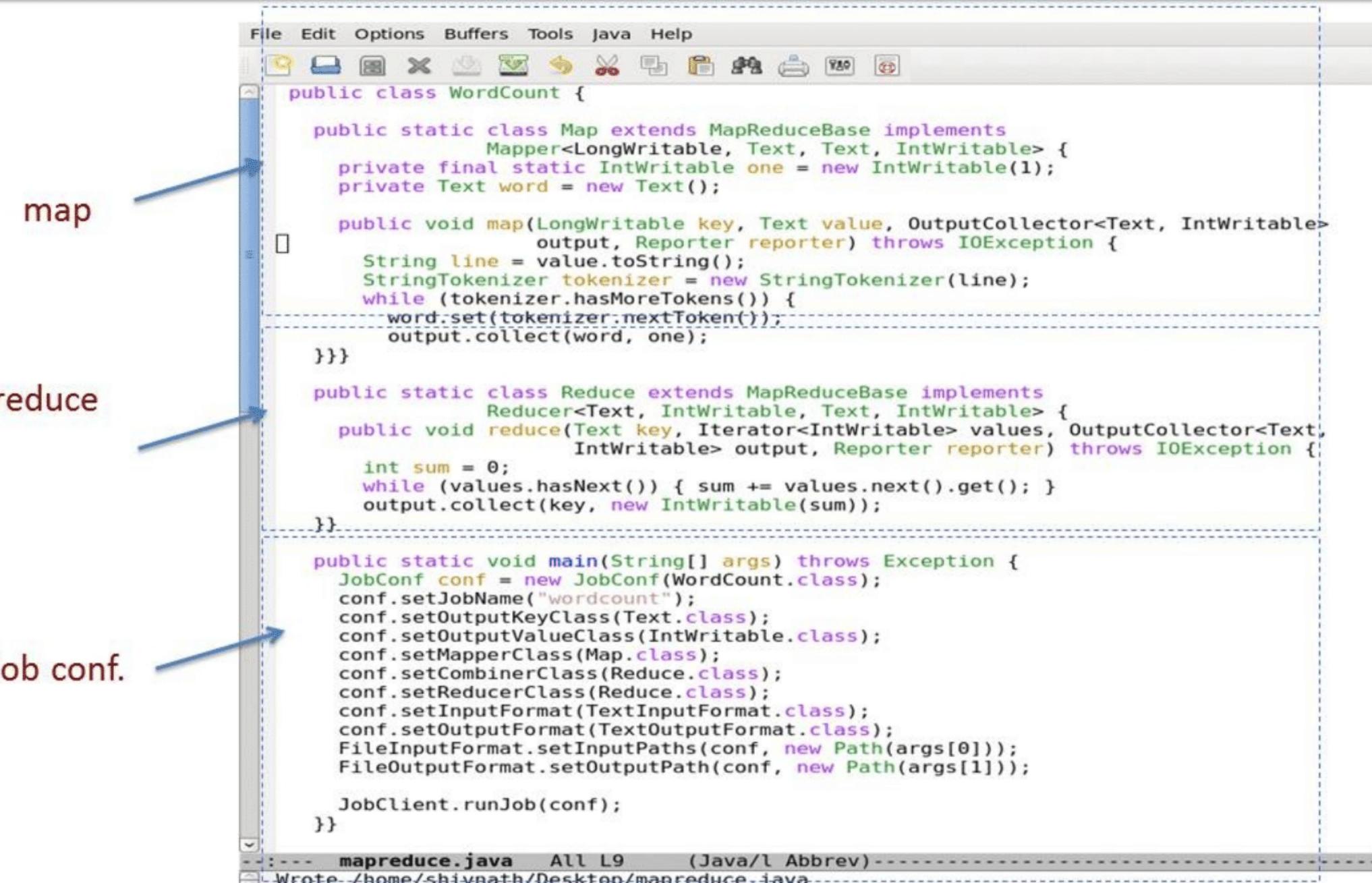
Runs on cheap commodity hardware

~~SQL~~

Procedural Control- a processing “pipe”

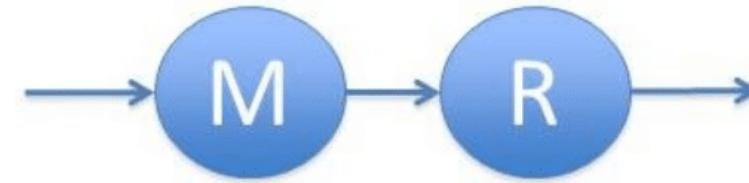


Java Example



```
File Edit Options Buffers Tools Java Help
public class WordCount {
    public static class Map extends MapReduceBase implements
        Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
            output, Reporter reporter) throws IOException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
        public static class Reduce extends MapReduceBase implements
            Reducer<Text, IntWritable, Text, IntWritable> {
            public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
                IntWritable> output, Reporter reporter) throws IOException {
                int sum = 0;
                while (values.hasNext()) { sum += values.next().get(); }
                output.collect(key, new IntWritable(sum));
            }
        }
        public static void main(String[] args) throws Exception {
            JobConf conf = new JobConf(WordCount.class);
            conf.setJobName("wordcount");
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            conf.setMapperClass(Map.class);
            conf.setCombinerClass(Reduce.class);
            conf.setReducerClass(Reduce.class);
            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            JobClient.runJob(conf);
        }
    }
}
----- mapreduce.java All L9 (Java/l Abbrev) -----
Wrote- /home/shivnath/Desktop/mapreduce.java-----
```

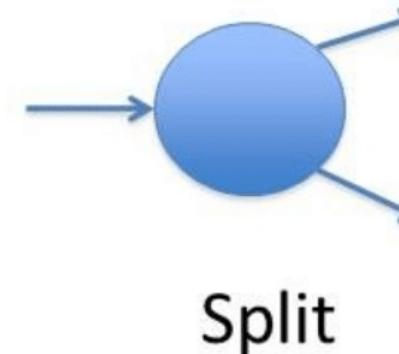
1. Extremely rigid data flow



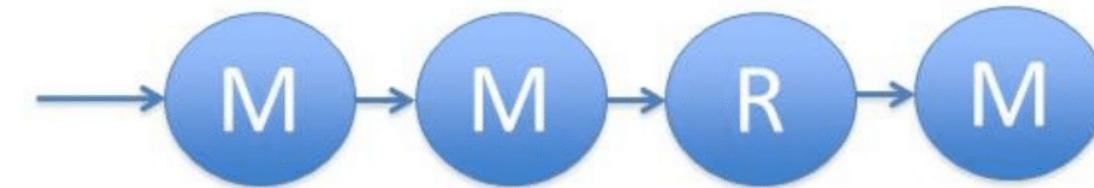
Other flows constantly hacked in



Join, Union



Split



Chains

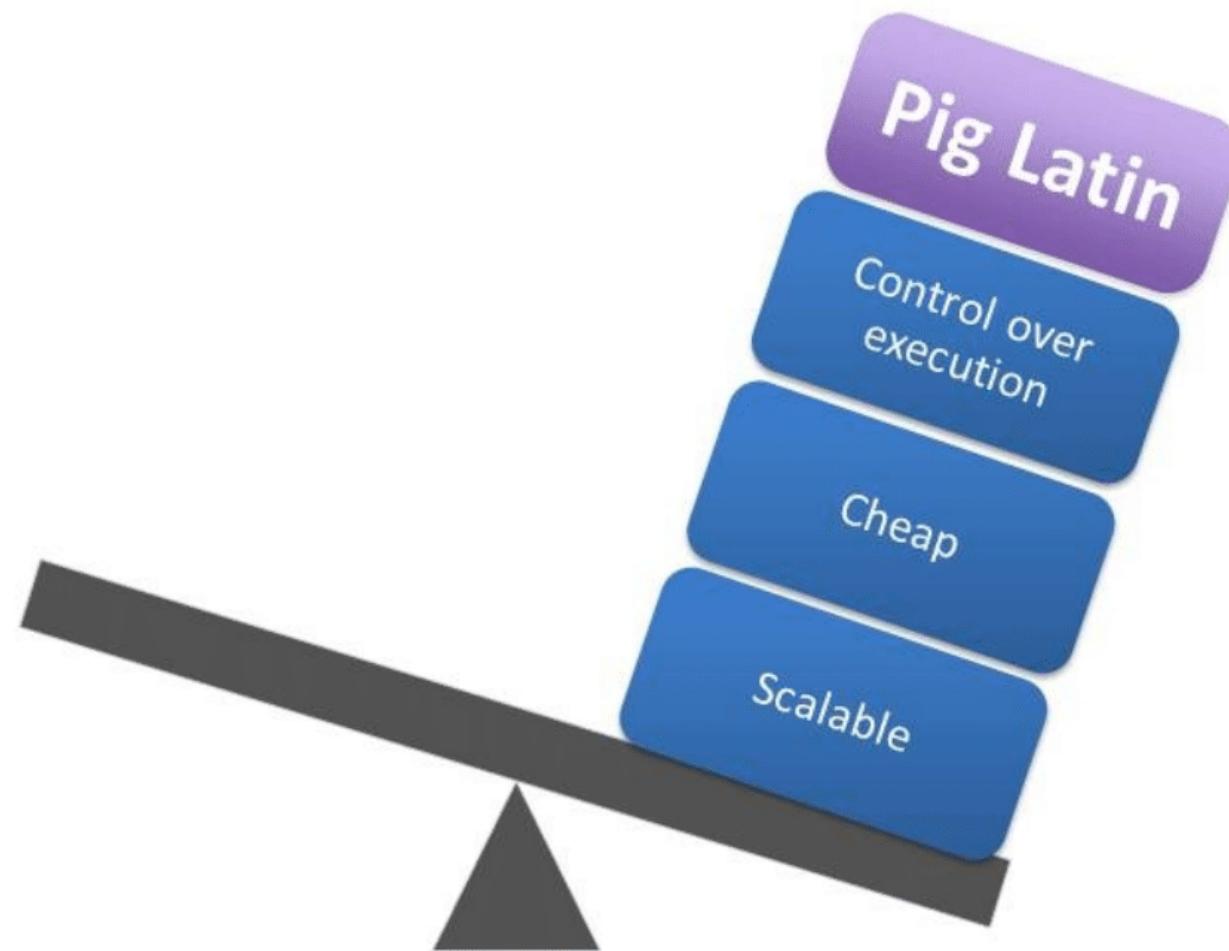
2. Common operations must be coded by hand

- Join, filter, projection, aggregates, sorting, distinct

Need a high-level, general data flow language

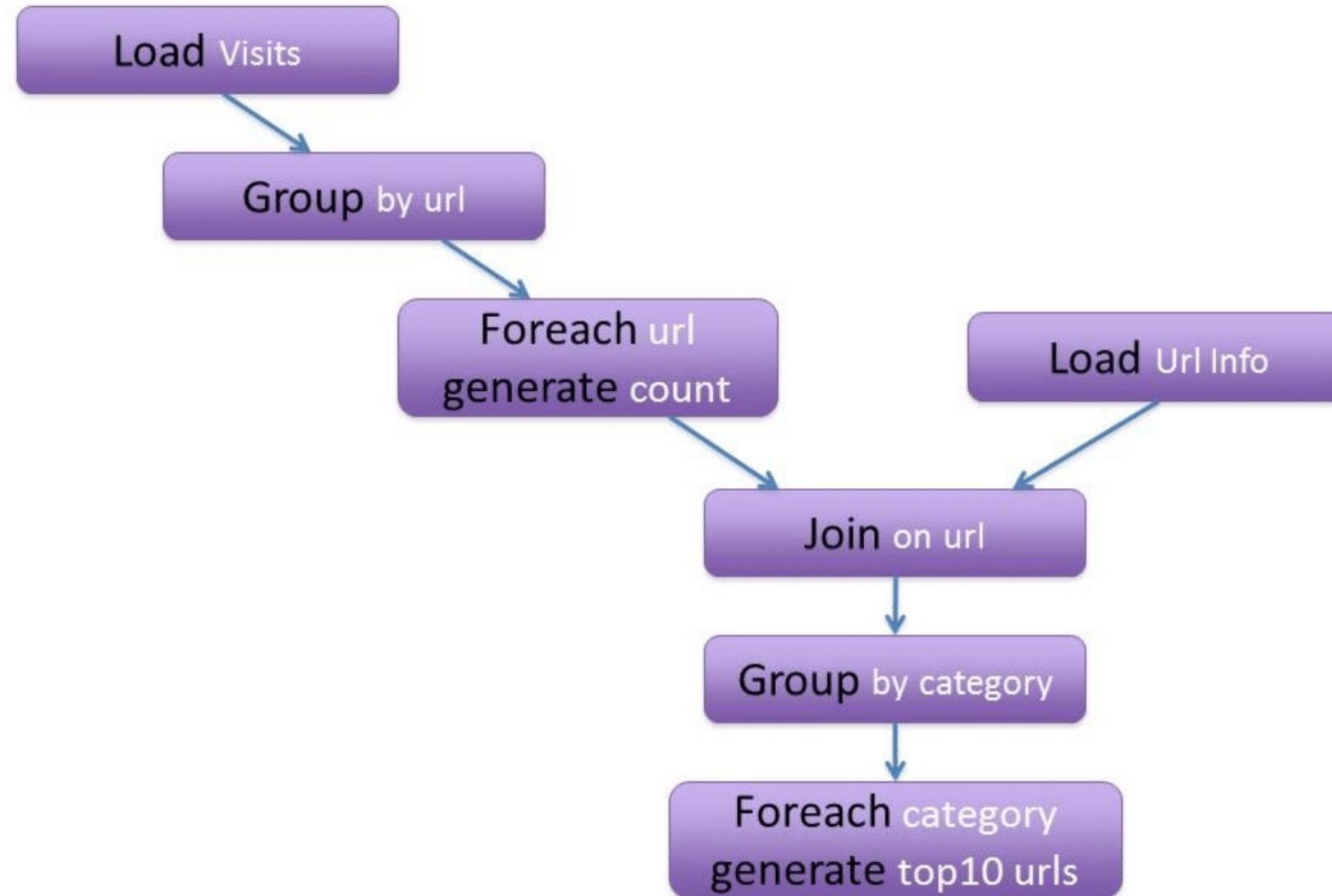


Need a high-level, general data flow language



- A platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs.
- Compiles down to Map Reduce jobs
- Developed by Yahoo!
- Open-source language





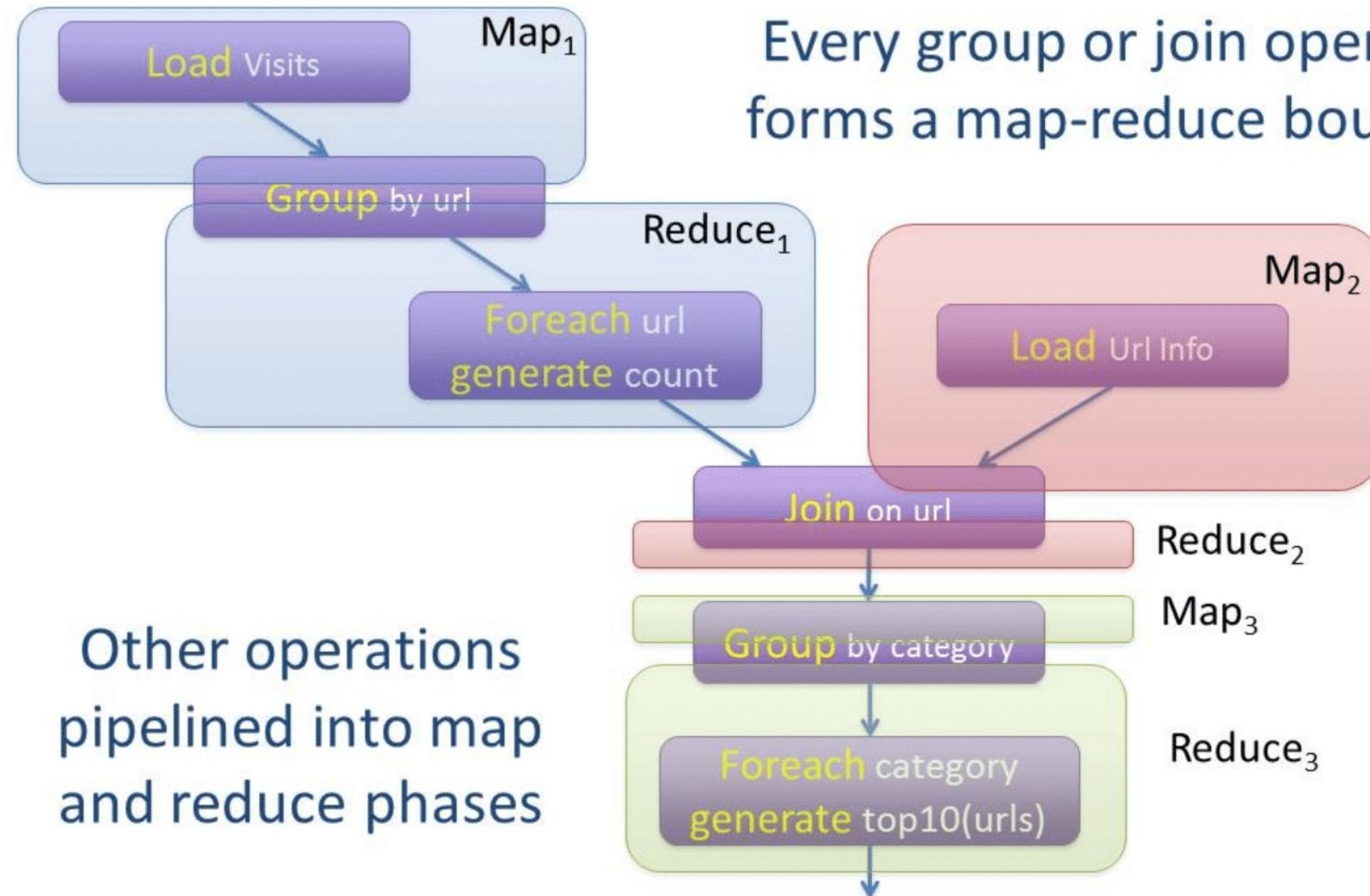
```
visits      = load '/data/visits' as (user, url, time);
gVisits     = group visits by url;
visitCounts = foreach gVisits generate url, count(visits);

urlInfo     = load '/data/urlInfo' as (url, category, pRank);
visitCounts = join visitCounts by url, urlInfo by url;

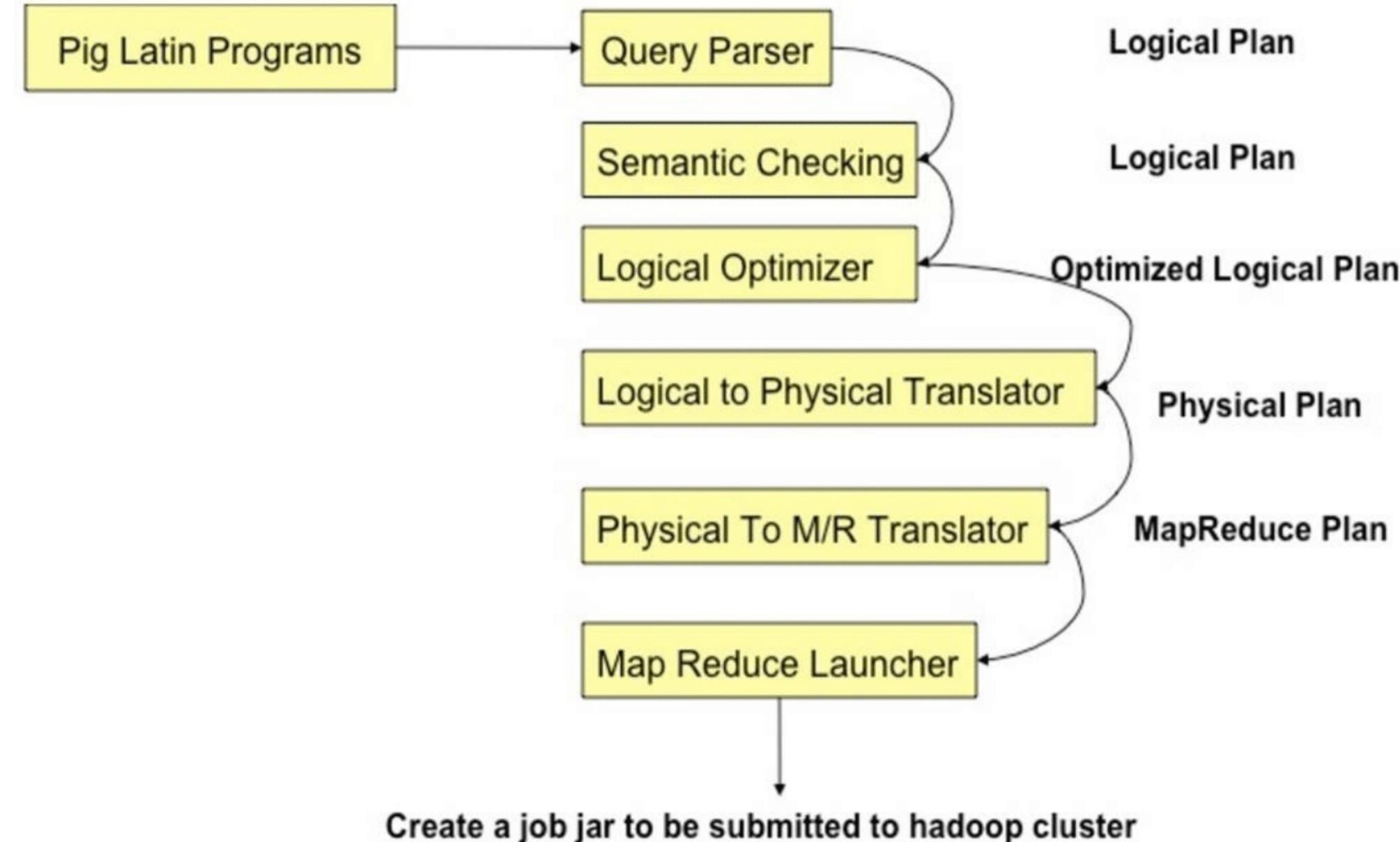
gCategories = group visitCounts by category;
topUrls = foreach gCategories generate top(visitCounts,10);

store topUrls into '/data/topUrls';
```

Contd..



Pig Compilation



Implementation

