MapReduce

Drew Dolgert
Cornell Center for Advanced Computing
December 8, 2010

Map and Reduce in Functional Programming

- Map
  f(x) = x^2
  map(f, [1,2,3,4]) => 1, 4, 9, 16
- Reduce or fold
  f(x,y) = x+y
  reduce(f, [1,2,3,4]) => (((1+2)+3)+4)
- Any kind of function or argument.
- Not necessarily associative.

Reverse Index

For each set of HTML files:
Extract all links and the files they link from.
Gather all (link,file) for the same link.
For each link:
Make a list of files with that link

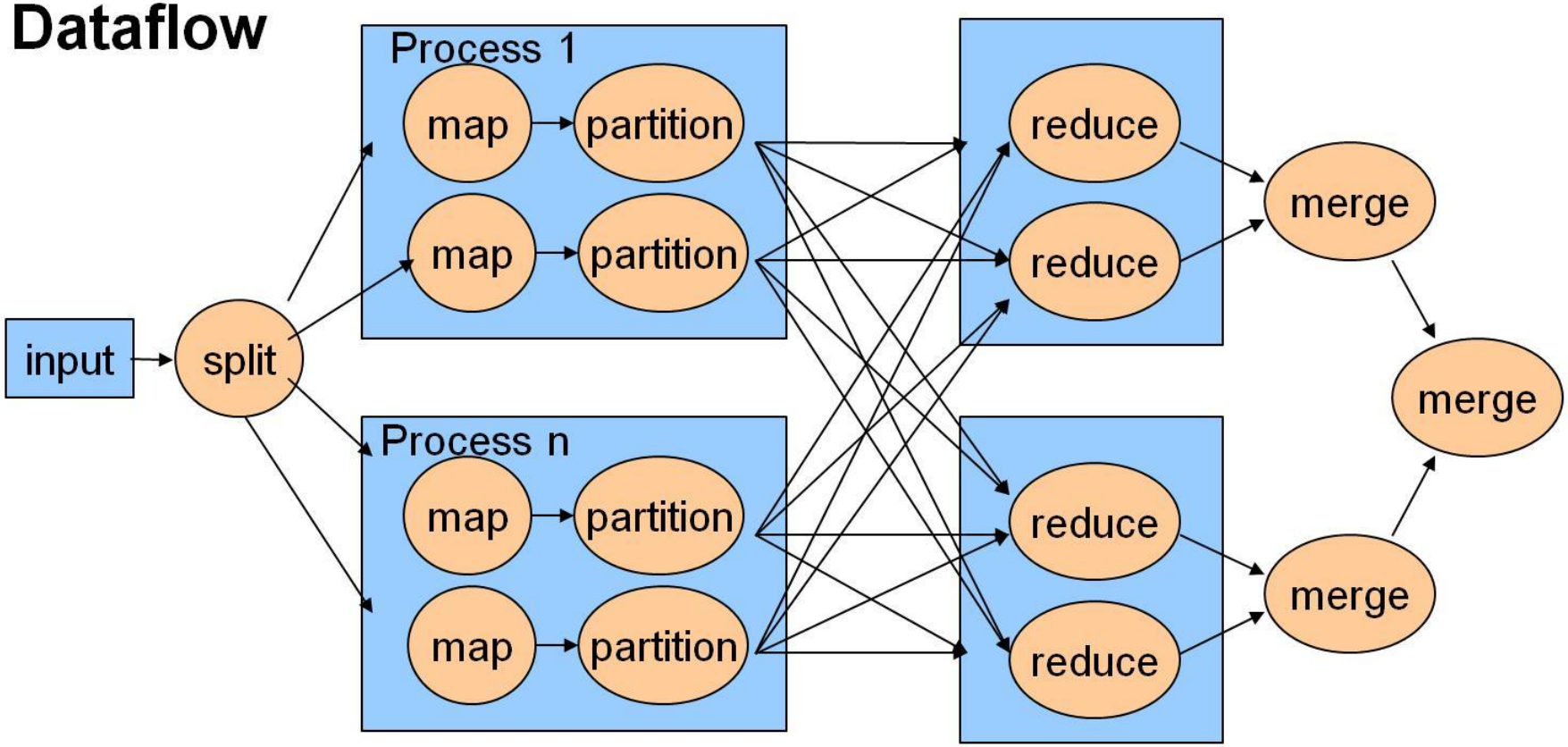
- All about the gather step.
- Typically a hashtable.

Reverse Index in Key-Value Pairs

For each set of HTML files:
Extract all links and the files they link from.
Save array of (link,file)
Gather all (link,file) for the same link.
For each (link, array of files):
Make a list of files with that link

- Standard is to pass key-value pairs.
- void* in C, typed in Java

Dataflow



After Colby Ranger, et al, on Evaluating MapReduce for Multi-core

Hadoop

- Hadoop is a software platform for running applications that process large sets of data.
- Hadoop is basically two components:
  - MapReduce – a programming paradigm for processing and generating data sets composed of a Map function followed by a Reduce funciton
    - Map –function that runs on all data pieces to generate a new data chunk
    - Reduce – function to merge data chunks from map step
  - Hadoop Distributed File System (HDFS) - creates multiple copies of the data and stores it in a distributed file system.
    - HDFS is designed to be reliable and to run on commodity hardware (that may fail)
- These work in concert because Map Reduce runs the appropriate Map function where the data lives (rather then sending the data)

A Sense of Hadoop

- One master, a server on each node
- Data files split and distributed across the system in some HDFS way
- Ask it to run your jar file on all files in a given directory, output to another
- Or write a program that does multiple map-reduce loops to find its answer.

- Write Map and Reduce functions for WordCount
  - Map tokenizes each line and sets <word, 1> into the pairs
  - Reduce sums for all word

```
14 public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
15    private final static IntWritable one = new IntWritable(1);
16    private Text word = new Text();
17
18    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
19        String line = value.toString();
20        StringTokenizer tokenizer = new StringTokenizer(line);
21        while (tokenizer.hasMoreTokens()) {
22            word.set(tokenizer.nextToken());
23            output.collect(word, one);
24        }
25    }
26 }
27
28 public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
29
30    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
31        int sum = 0;
32        while (values.hasNext()) {
33            sum += values.next().get();
34        }
35        output.collect(key, new IntWritable(sum));
36    }
37 }
```

Hadoop

Map and Reduce functions are tied into a Job class, which makes them resuable.

```
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);
}
```

Hadoop Programming

- The previous example showed Java, but other options are supported
  - Python – via Jython to produce a jar
  - C and C++
    - Build a binary executable
    - Load the binary onto a location in HDFS
    - Execute the binary be using the built-in pipes algorithm and an XML configuration file.
- Using things like streaming hadoop, it's possible to use things like shell and perl scripts to execute code under hadoop.
- Additional tools allow non-java access to the HDFS-API, so applications can fetch and store data.

HOD

- Hadoop is designed primarily to run as the only thing running on a cluster, not an application executed on a general-purpose system.
- Hadoop on Demand (HOD) is a tool for starting Hadoop on a subset of nodes inside of a general-purpose HPC system. HDFS and MapReduce daemons are brought up as part of a normal job, which results in a temporary Hadoop system which jobs can be run on.
- We have created a patch to run HOD on Ranger and can work with individual clients to help them get this running for their needs (initial configuration help is probably needed as the patch has not yet been formally accepted into Hadoop core).

Hadoop on Demand

- ***Allocation*** – provisions a Hadoop system on a subset of the system and creates a needed config file to run jobs.

```
$ export HOD_PYTHON_HOME=/usr/local/bin/python2.5
$ export HOD_CONF_DIR=${HOD_HOME}/conf
$ cd ${HOD_HOME}/bin
$ ./hod allocate -d ~/hadoop/cluster -n 5 -t ~/hadoop/hadoop-0.18.3.tar.gz -A acctName -l 3600
INFO - Cluster Id 2938.scheduler.v4linux
INFO - HDFS UI at http://compute-3-42.v4linux:54072
INFO - Mapred UI at http://compute-3-43.v4linux:52010
INFO - hadoop-site.xml at /home/gfs01/naw47/hadoop/cluster
$ ./hod list -d ~/hadoop/cluster
INFO - alive 2938.scheduler.v4linux /home/gfs01/naw47/hadoop/cluster
```

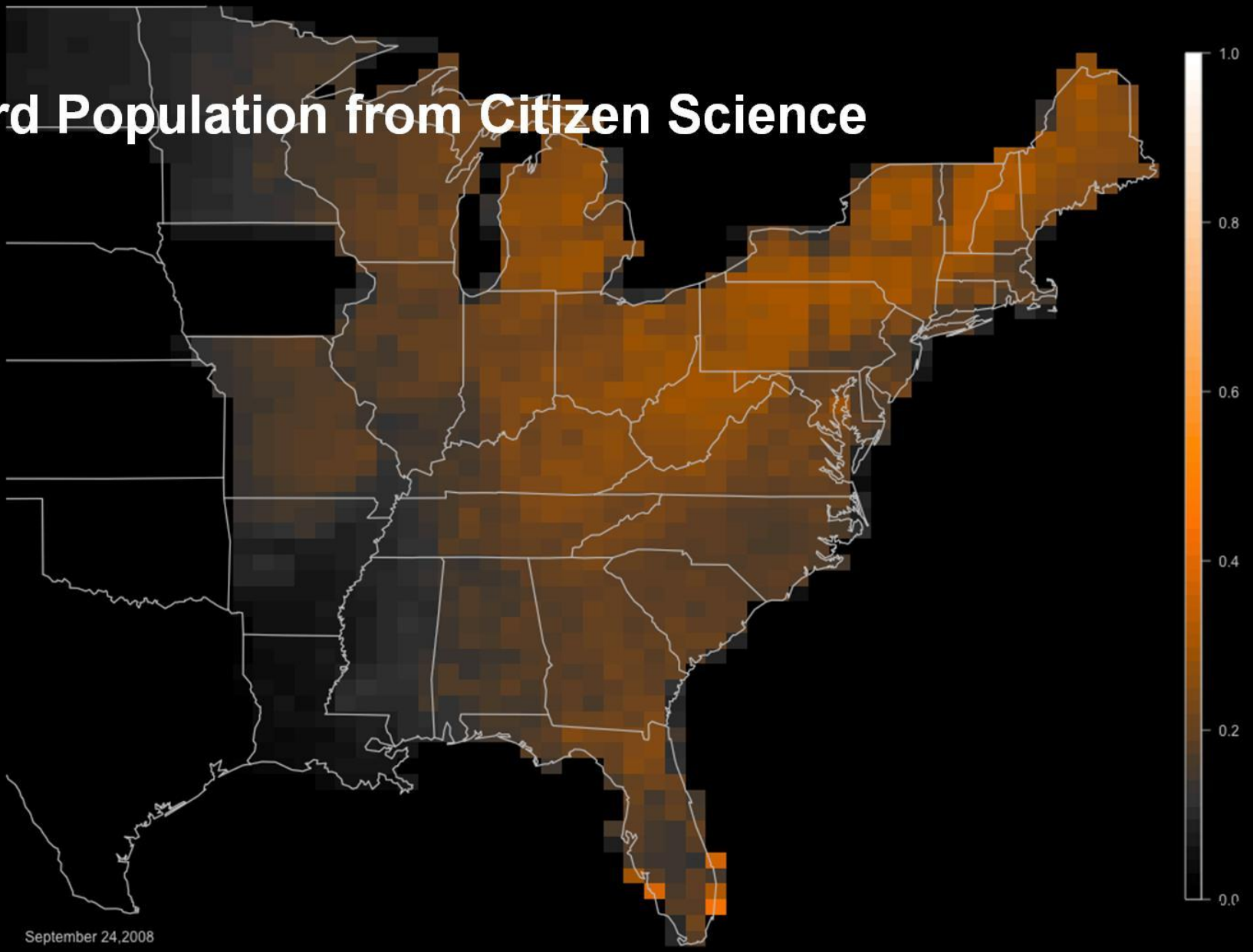
- ***Data is a problem*** – Using HOD this way means that data must be reloaded onto HDFS for every use, which means that a significant part of job time may be loading data.

```
$ export HADOOP_CONF_DIR=~/hadoop/cluster
$ cd hadoop-0.18.3/bin
$ ./hadoop fs -mkdir input
$ ./hadoop fs -mkdir output
$ ./hadoop fs -put ~/somefile input/words
$ ./hadooop jar ~/path/to/hadoop-0.18.3-examples.jar wordcount input/words output/wordcount
$ ./hadoop fs -get output/wordcount ~/wc_results
```

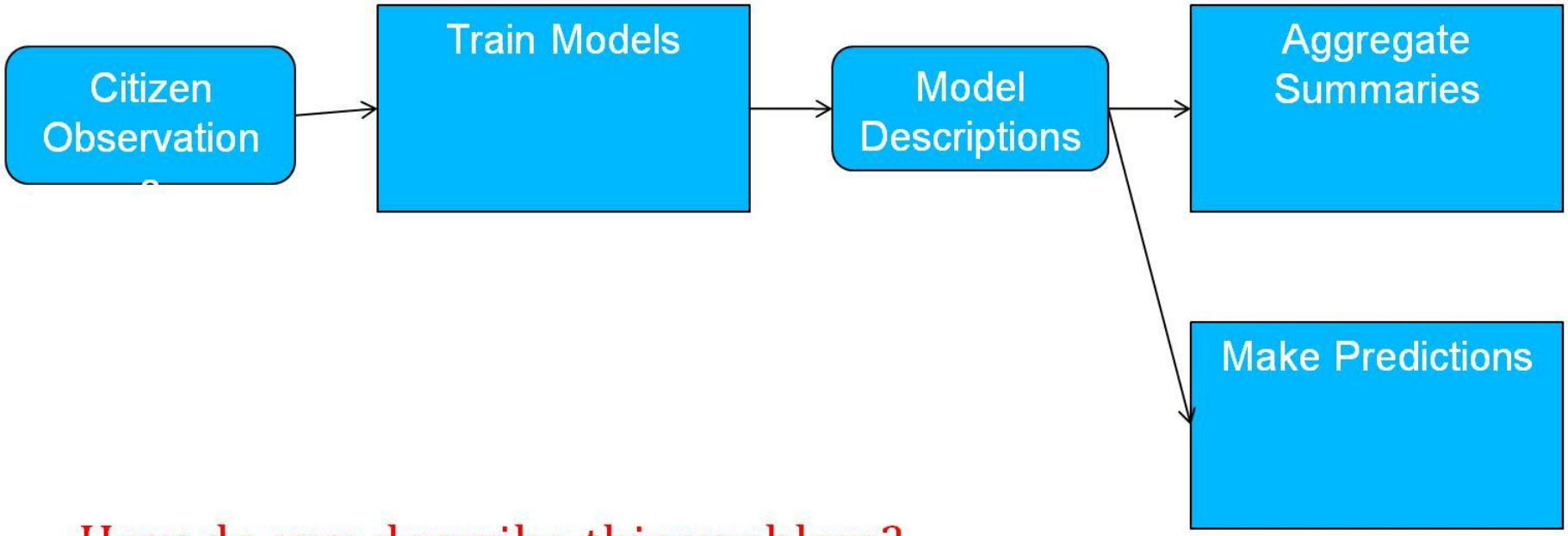
Why Use MapReduce

- Split between independent tasks and communication very explicit.
- Natural for some algorithms
  - Sorting
  - Reverse index
  - Data mining-ish things
- The framework figures out
  - How to bring computation to the data
  - Where to aggregate key-value pairs
  - If there was a failure

Bird Population from Citizen Science

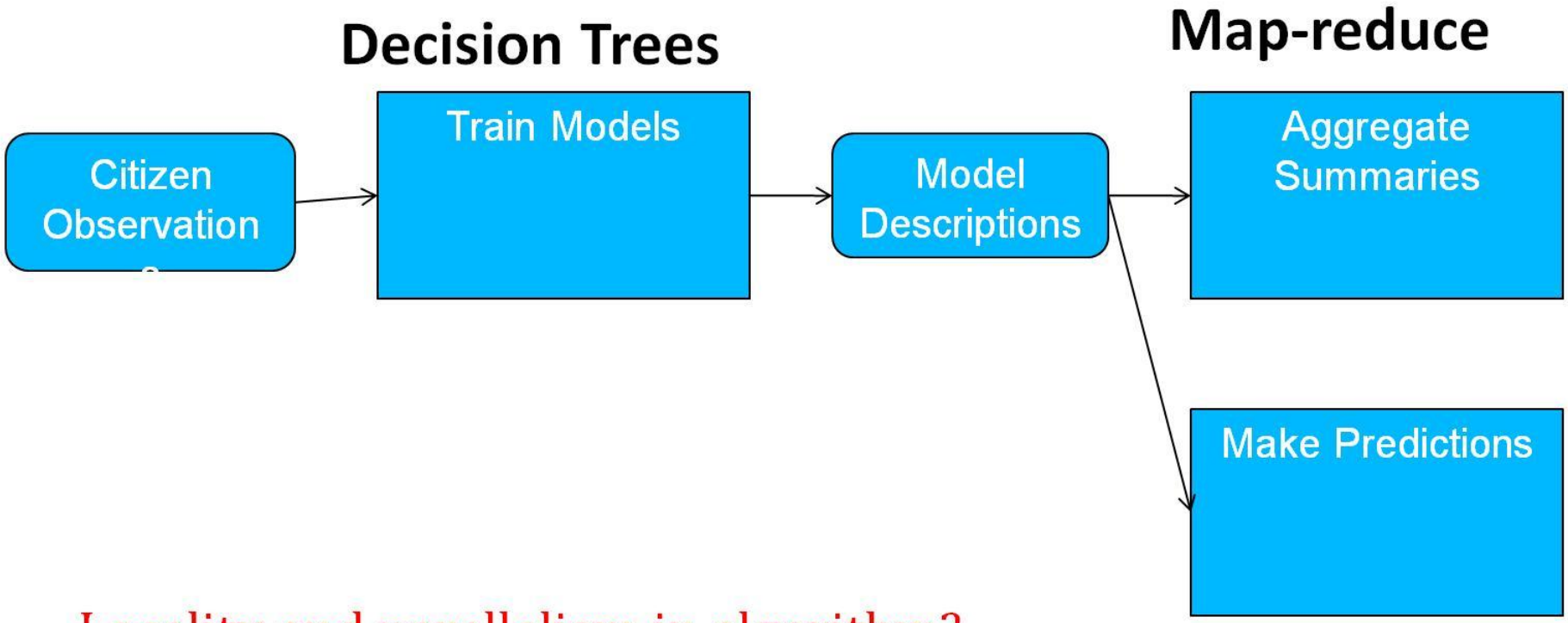


Architecture of Bird Models



How do you describe this problem?

Algorithm

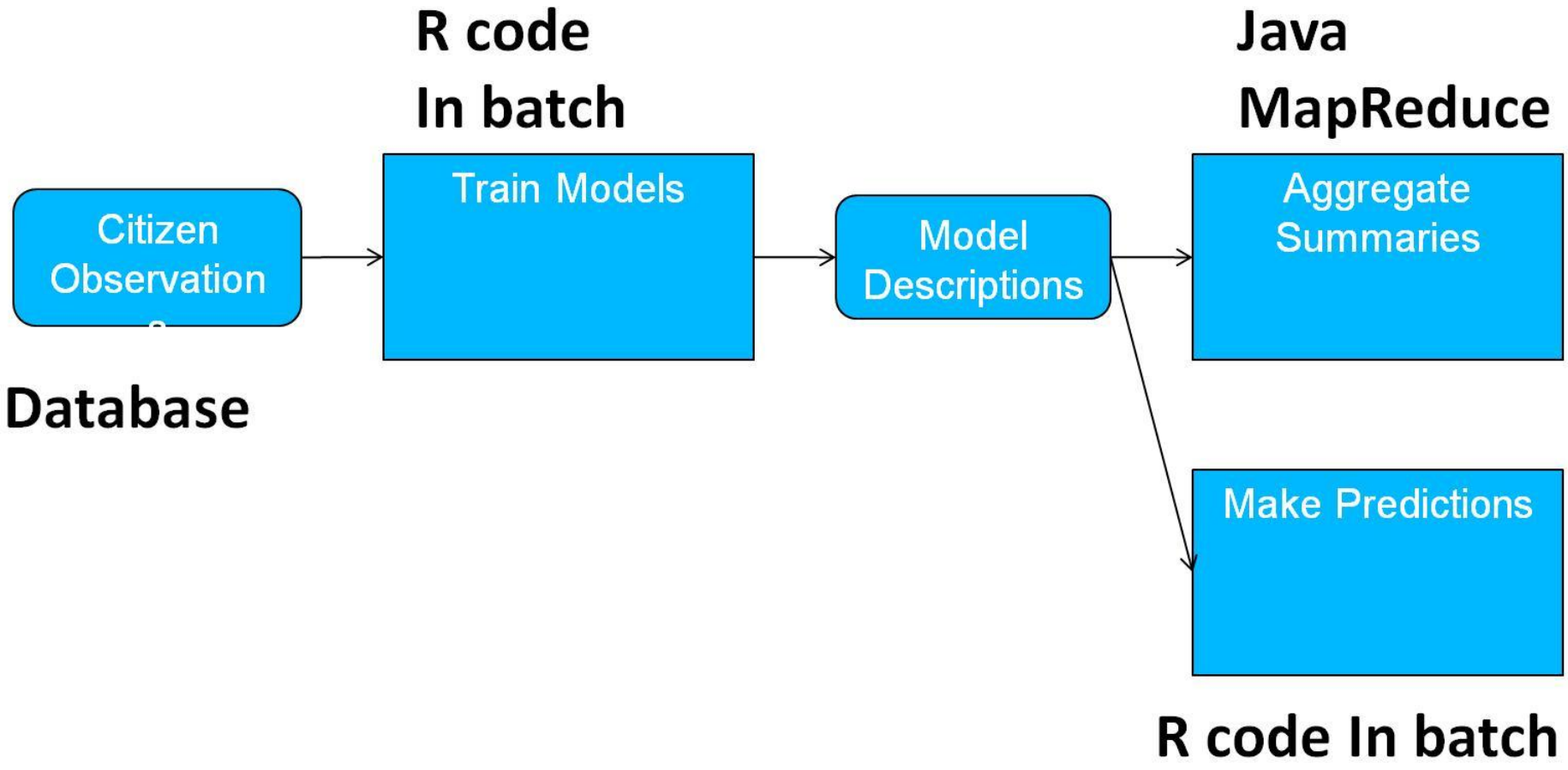


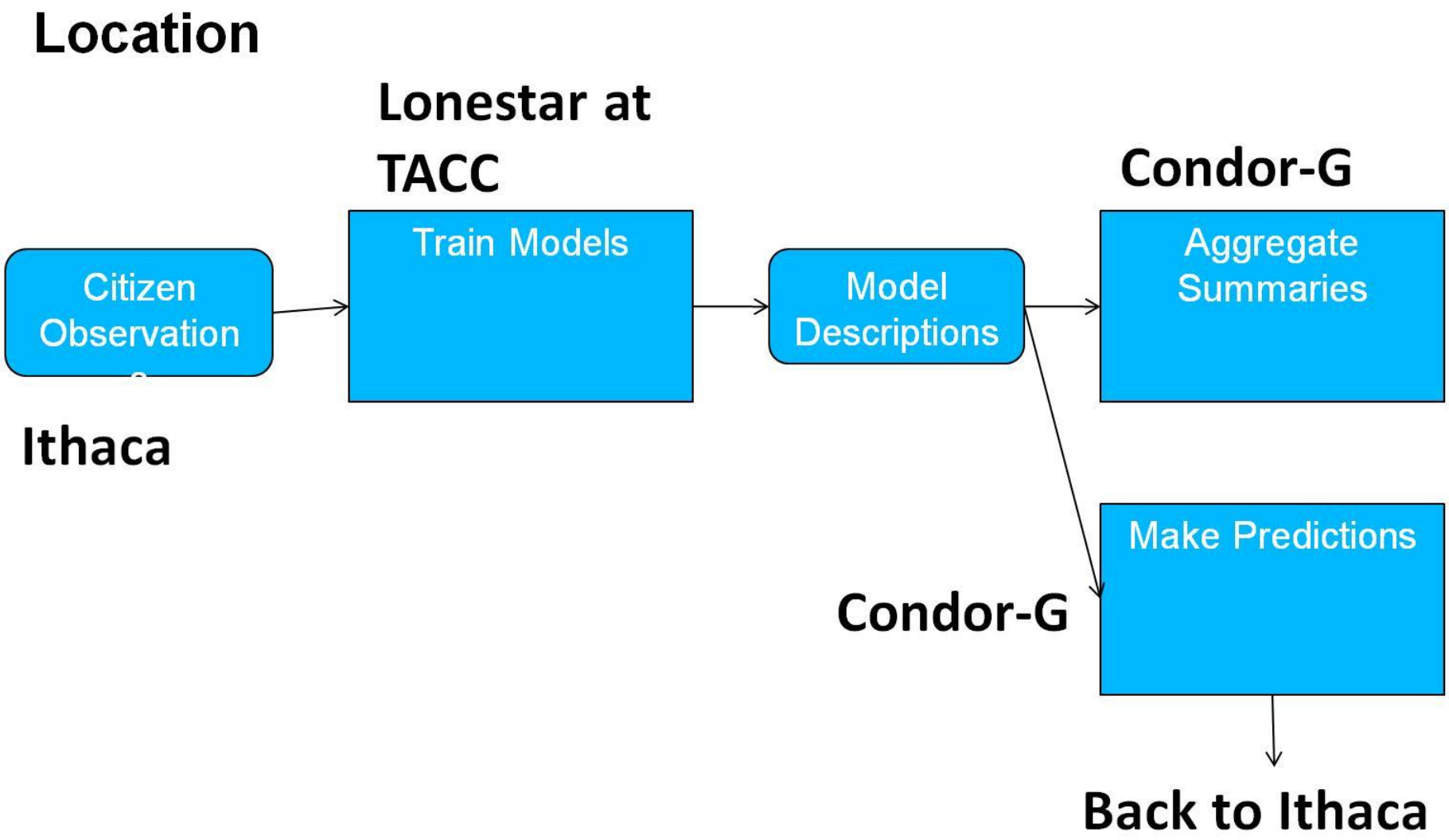
Locality and parallelism in algorithm?

Geometry

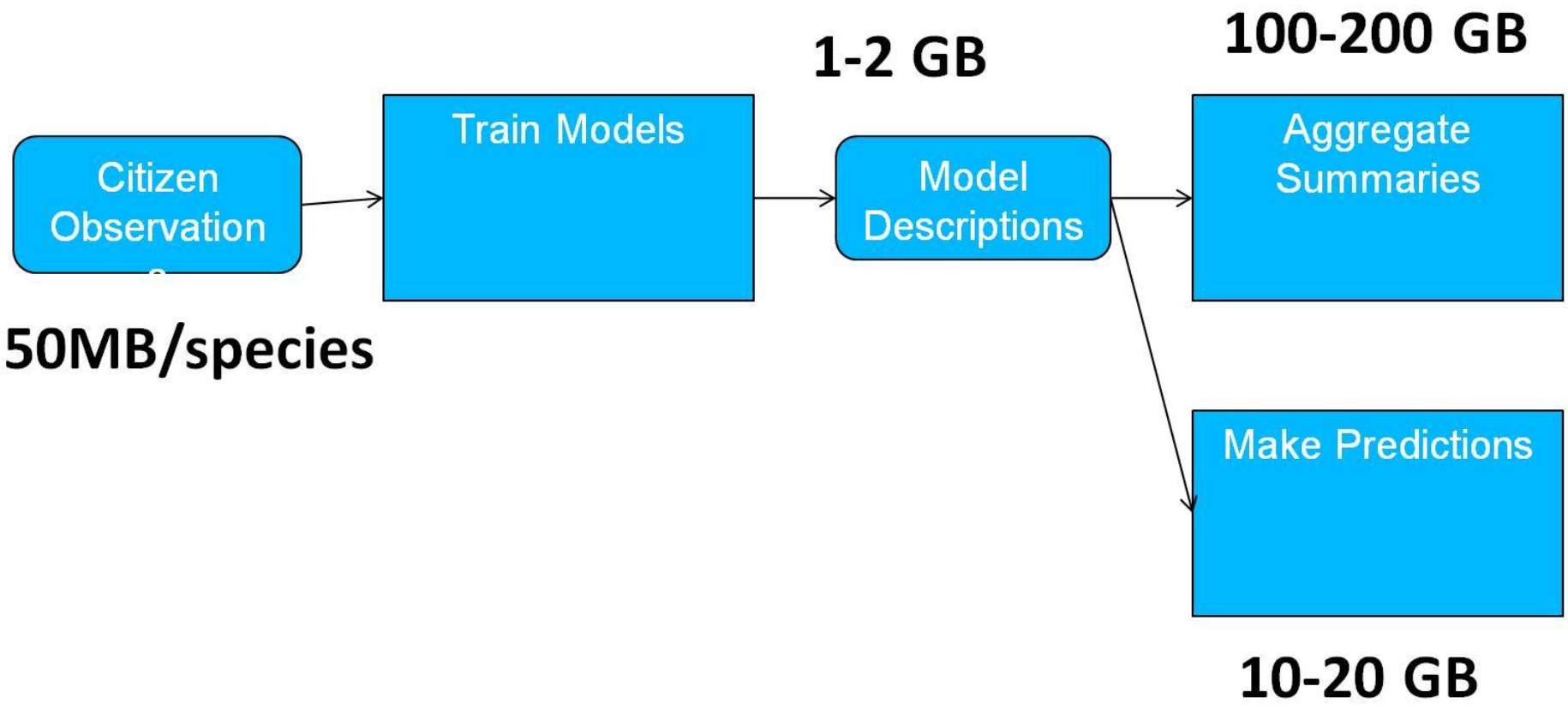
Intersection loop

Tools


R code
In batch
Java
MapReduce
Citizen Observation
Train Models
Model Descriptions
Aggregate Summaries
Database
Make Predictions
R code In batch


Location
Lonestar at TACC
Condor-G
Train Models
Citizen Observation
Model Descriptions
Aggregate Summaries
Ithaca
Make Predictions
Condor-G
Back to Ithaca

Data


1-2 GB
100-200 GB
Citizen Observation
Train Models
Model Descriptions
Aggregate Summaries
50MB/species
Make Predictions
10-20 GB

How Does Bird Population Depend on X?

- F(elevation, year, human population).
- Average over years and population densities.
  - F(elevation, year0, hpop0), F(elevation,year1,hpop1).

- Dataset for each bird type
- Summaries over each of many parameters.
- Visualizing at many points per summary.

It's a sum that can be done in parts.

Phoenix Multi-core MapReduce

- Ranger, Raghuraman, Penmetsa, Bradski, Kozyrakis at Stanford
- Set of C routines using Pthreads to implement MapReduce on a single CMP (CMP, SMP?). You just call it from C/C++.
- Load balancing thread library
- Compact MapReduce API
- Little data transfer overhead
- Handle transient faults
- Adjust granularity to cache availability

Implementation

- **Splitter** – yields chunks for mappers

1) **Map**

- **Partition** – aggregates key-value pairs

1) **Reduce**

- One C file, or maybe three, but same diff.
- Call it with work, and it spawns threads, allocates memory, cleans up at the end of the call.
- Tunable parameters for how to split, what hash to partition, numbers of tasks.

Code Size Ratio to Serial Implementation

Algorithm	Pthreads	Phoenix
Word count	1.8	0.9
Matrix multiply	1.8	2.2
Reverse index	1.5	0.9
Kmeans	1.2	1.7
String match	1.8	1.5
PCA	1.7	2.5
Histogram	2.4	2.2
Linear regression	1.7	1.6

Phoenix uses Pthreads, but it's a matter of *static* versus *dynamic* allocation.

How Would You Implement?

- Matix Multiply
- Principal Component Analysis
- What are the main concerns?

Matrix Multiply

- Split matrix into rows
- Map multiplies the rows, puts result into key=(i,j) of destination
- Reduce does nothing
- A little faster than naïve pthreads

Principal Component Analysis (PCA)

- Find the mean vector, split among rows
- Reduce task does nothing
- Second round, each map computes some of covariance matrix
- Reduce task does nothing.
- Much slower than Pthreads version.
- "Suffers from saturation of the bus that interconnects the processors."

Phoenix Performance

- Surprisingly good, even on matrix-multiply.
- It's about cache use.
- Concurrent multi-processors (CMPs) not Symmetric Multi-processors (SMPs)
- Ranger is CMP. Memory allocated near process, thread on another processor picks it up.

Just One Way to Use Multicore Systems

- Pthreads – cross-platform threading library
- OpenMP – directives in C / Fortran
- Languages: CILK, Chapel, UPC and X10
- Tool support: Parallel Matlab, Mathematica, IPython
- Libraries: PETSc, Intel MKL

Ways to Use It

- As part of an algorithm for a job, with other code
- Interactively, because those daemons are running
  - Pig and Pig Latin
  - Your own command-line

Conclusion

- Novel tools and interfaces
  - For data mining
  - For interactive use of large resources