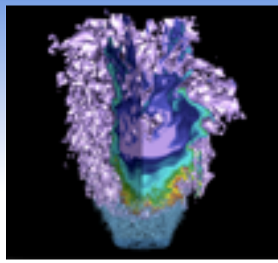
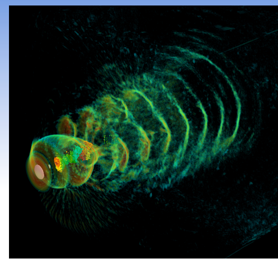
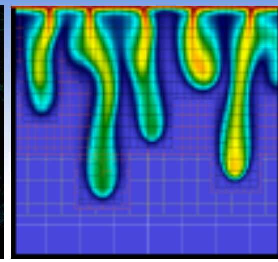
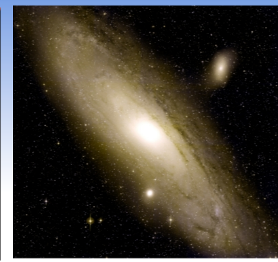
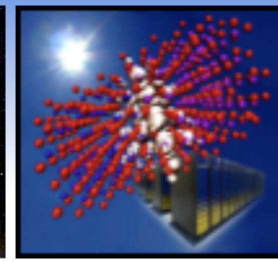
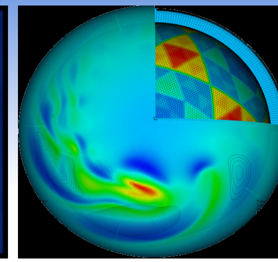
Outline of Tutorial

- Hadoop and Pig Overview
- Hands-on

Hadoop and Pig Overview

Lavanya Ramakrishnan
Shane Canon

Lawrence Berkeley National Lab

October 2011

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

Overview

- Concepts & Background
  - MapReduce and Hadoop
- Hadoop Ecosystem
  - Tools on top of Hadoop
- Hadoop for Science
  - Examples, Challenges
- Programming in Hadoop
  - Building blocks, Streaming, C-HDFS API

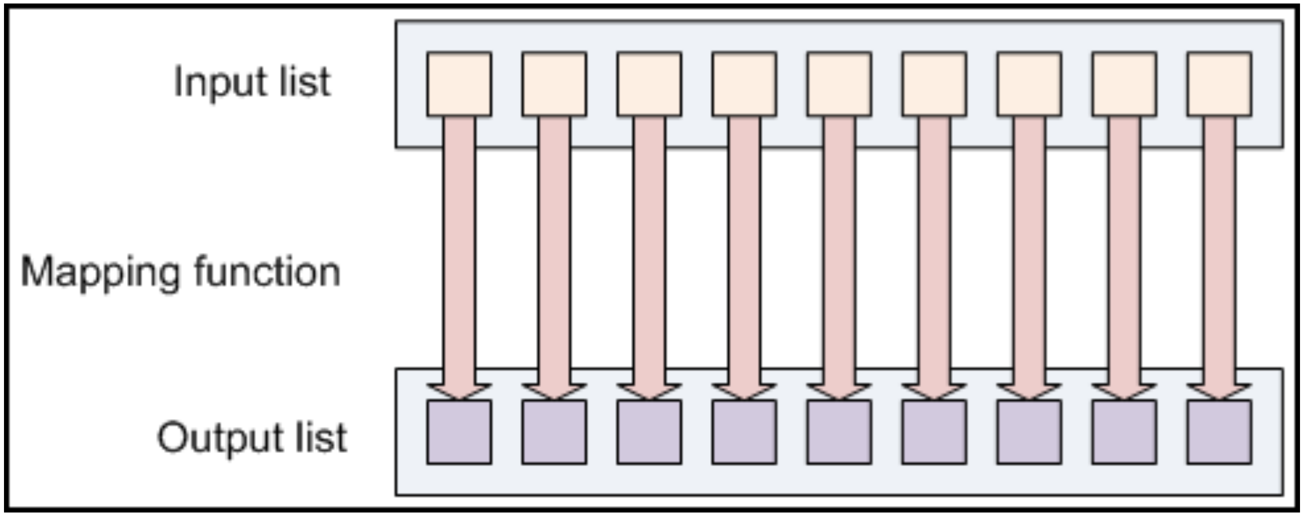
Processing Big Data

- Internet scale generates BigData
  - Terabytes of data/day
  - just reading 100 TB can be overwhelming
    - using clusters of standard commodity computers for linear scalability
- Timeline
  - Nutch open source search project (2002-2004)
  - MapReduce & DFS implementation and Hadoop splits out of Nutch (2004-2006)

MapReduce

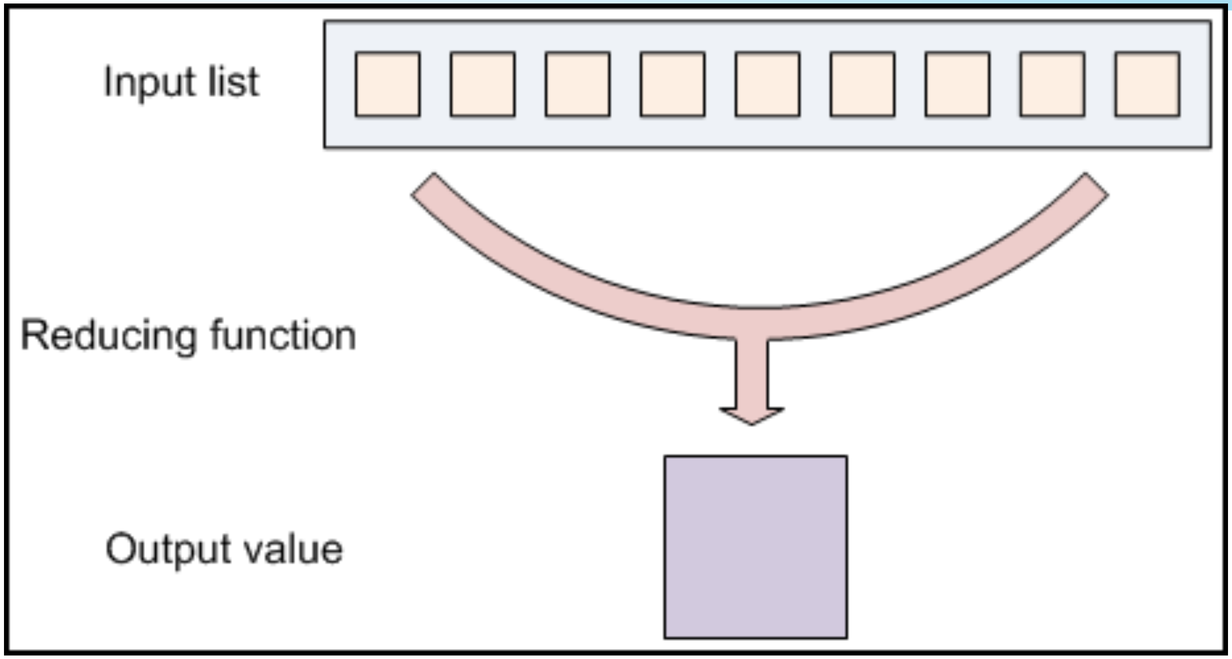
- Computation performed on large volumes of data in parallel
  - divide workload across large number of machines
  - need a good data management scheme to handle scalability and consistency
- Functional programming concepts
  - map
  - reduce

Mapping



- **Map input to an output using some function**
- **Example**
  - **string manipulation**

Reduces



- **Aggregate values together to provide summary data**
- **Example**
  - **addition of the list of numbers**

Google File System

- **Distributed File System**
  - accounts for component failure
  - multi-GB files and billions of objects
- **Design**
  - single master with multiple chunkservers per master
  - file represented as fixed-sized chunks
  - 3-way mirrored across chunkservers

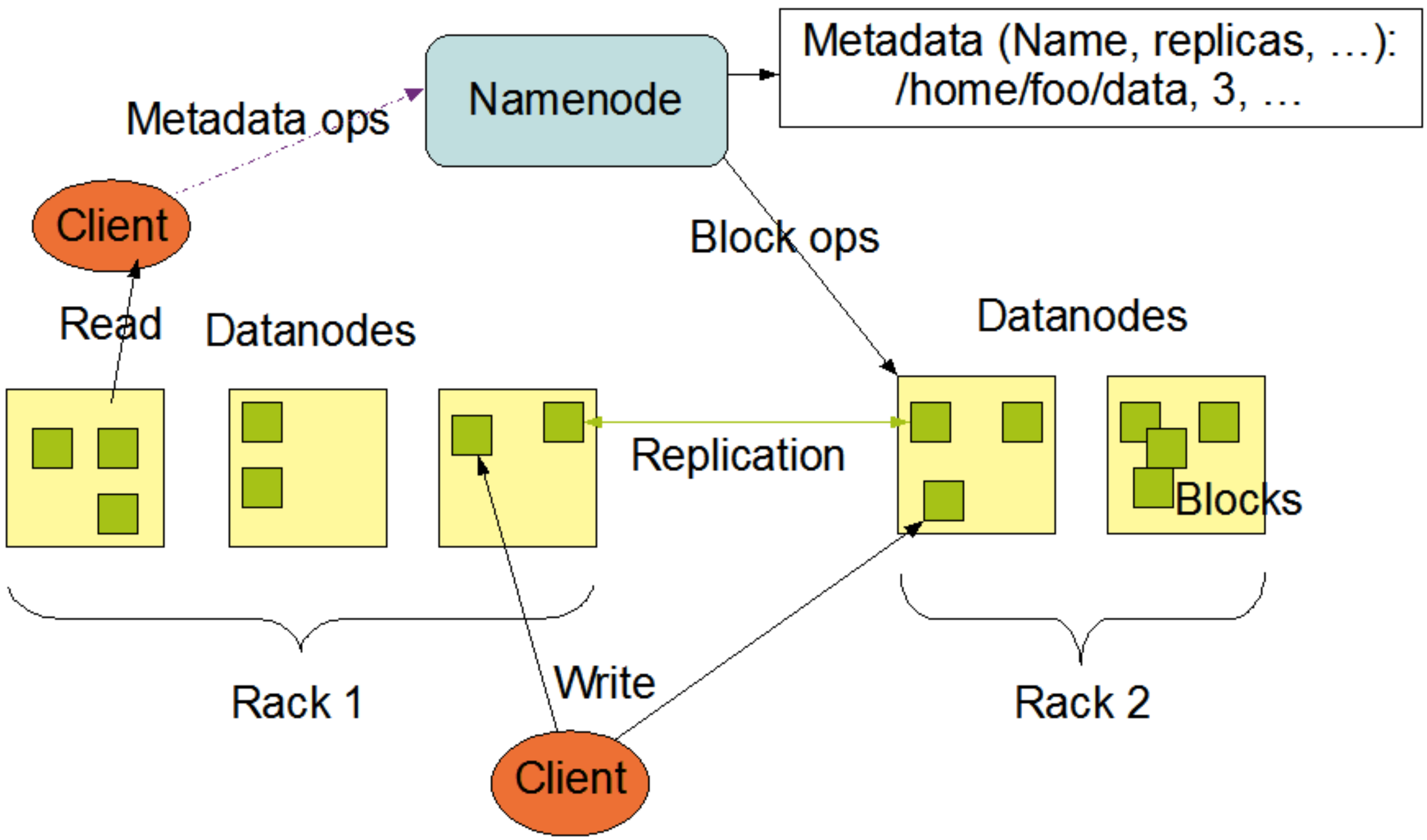
Hadoop

- Open source reliable, scalable distributed computing platform
  - implementation of MapReduce
  - Hadoop Distributed File System (HDFS)
  - runs on commodity hardware
- Fault Tolerance
  - restarting tasks
  - data replication
- Speculative execution
  - handles stragglers

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

HDFS Architecture



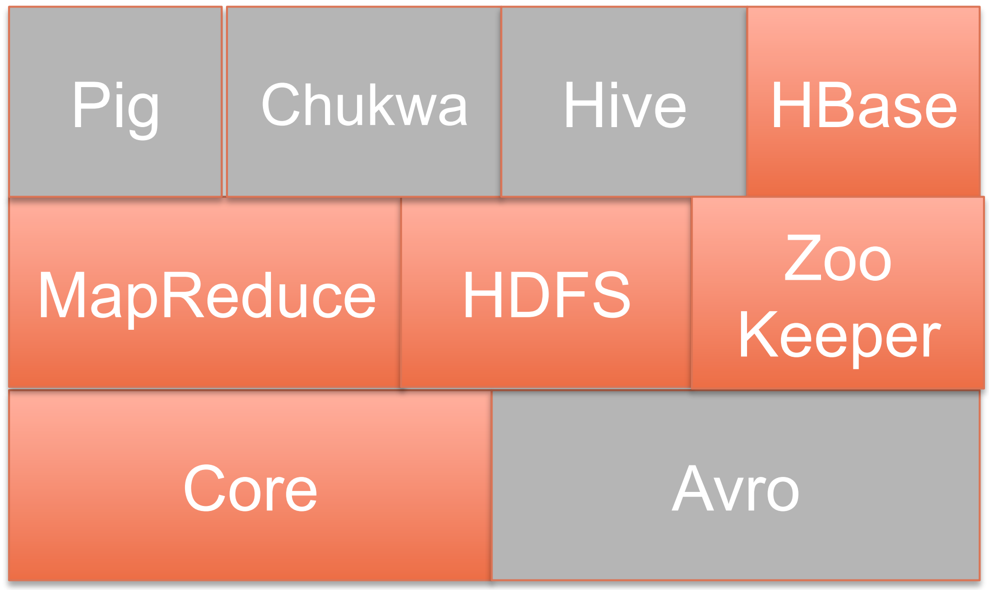
HDFS and other Parallel Filesystems

	HDFS	GPFS and Lustre
Typical Replication	3	1
Storage Location	Compute Node	Servers
Access Model	Custom (except with Fuse)	POSIX
Stripe Size	64 MB	1 MB
Concurrent Writes	No	Yes
Scales with	# of Compute Nodes	# of Servers
Scale of Largest Systems	O(10k) Nodes	O(100) Servers
User/Kernel Space	User	Kernel

Who is using Hadoop?

- A9.com
- Amazon
- Adobe
- AOL
- Baidu
- Cooliris
- Facebook
- NSF-Google university initiative
- IBM
- LinkedIn
- Ning
- PARC
- Rackspace
- StumbleUpon
- Twitter
- Yahoo!

Hadoop Stack



Source: Hadoop: The Definitive Guide

Constantly evolving!

Google Vs Hadoop

Google	Hadoop
MapReduce	Hadoop MapReduce
GFS	HDFS
Sawzall	Pig, Hive
BigTable	Hbase
Chubby	Zookeeper
Pregel	Hama, Giraph

Pig

- Platform for analyzing large data sets
- Data-flow oriented language "Pig Latin"
  - data transformation functions
  - datatypes include sets, associative arrays, tuples
  - high-level language for marshalling data
- Developed at Yahoo!

Hive

- SQL-based data warehousing application
  - features similar to Pig
  - more strictly SQL-type
- Supports SELECT, JOIN, GROUP BY, etc
- Analyzing very large data sets
  - log processing, text mining, document indexing
- Developed at Facebook

HBase

- **Persistent, distributed, sorted, multidimensional, sparse map**
  - **based on Google BigTable**
  - **provides interactive access to information**
- **Holds extremely large datasets (multi-TB)**
- **High-speed lookup of individual (row, column)**

ZooKeeper

- **Distributed consensus engine**
  - runs on a set of servers and maintains state consistency
- **Concurrent access semantics**
  - leader election
  - service discovery
  - distributed locking/mutual exclusion
  - message board/mailboxes
  - producer/consumer queues, priority queues and multi-phase commit operations

Other Related Projects [1/2]

- Chukwa – Hadoop log aggregation
- Scribe – more general log aggregation
- Mahout – machine learning library
- Cassandra – column store database on a P2P backend
- Dumbo – Python library for streaming
- Spark – in memory cluster for interactive and iterative
- Hadoop on Amazon – Elastic MapReduce

Other Related Projects [2/2]

- **Sqoop – import SQL-based data to Hadoop**
- **Jaql – JSON (JavaScript Object Notation) based semi-structured query processing**
- **Oozie – Hadoop workflows**
- **Giraph – Large scale graph processing on Hadoop**
- **Hcatlog – relational view of HDFS**
- **Fuse-DS – POSIX interface to HDFS**

Hadoop for Science

Magellan and Hadoop

- **DOE funded project to determine appropriate role of cloud computing for DOE/SC midrange workloads**
- **Co-located at Argonne Leadership Computing Facility (ALCF) and National Energy Research Scientific Center (NERSC)**
- **Hadoop/Magellan research questions**
  - **Are the new cloud programming models useful for scientific computing?**

–

Data Intensive Science

- **Evaluating hardware and software choices for supporting next generation data problems**
- **Evaluation of Hadoop**
  - **using mix of synthetic benchmarks and scientific applications**
  - **understanding application characteristics that can leverage the model**
    - **data operations: filter, merge, reorganization**
    - **compute-data ratio**

(collaboration w/ Shane Canon, Nick Wright, Zacharia Fadika)

MapReduce and HPC

- Applications that can benefit from MapReduce/Hadoop
  - Large amounts of data processing
  - Science that is scaling up from the desktop
  - Query-type workloads
- Data from Exascale needs new technologies
  - Hadoop On Demand lets one run Hadoop through a batch queue

Hadoop for Science

- Advantages of Hadoop
  - transparent data replication, data locality aware scheduling
  - fault tolerance capabilities
- Hadoop Streaming
  - allows users to plug any binary as maps and reduces
  - input comes on standard input

BioPig

- **Analytics toolkit for Next-Generation Sequence Data**
- **User defined functions (UDF) for common bioinformatics programs**
  - BLAST, Velvet
  - readers and writers for FASTA and FASTQ
  - pack/unpack for space conservation with DNA sequencesâ

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

Application Examples

- **Bioinformatics applications (BLAST)**
  - parallel search of input sequences
  - Managing input data format
- **Tropical storm detection**
  - binary file formats can't be handled in streaming
- **Atmospheric River Detection**
  - maps are differentiated on file and parameter

“Bring your application” Hadoop workshop

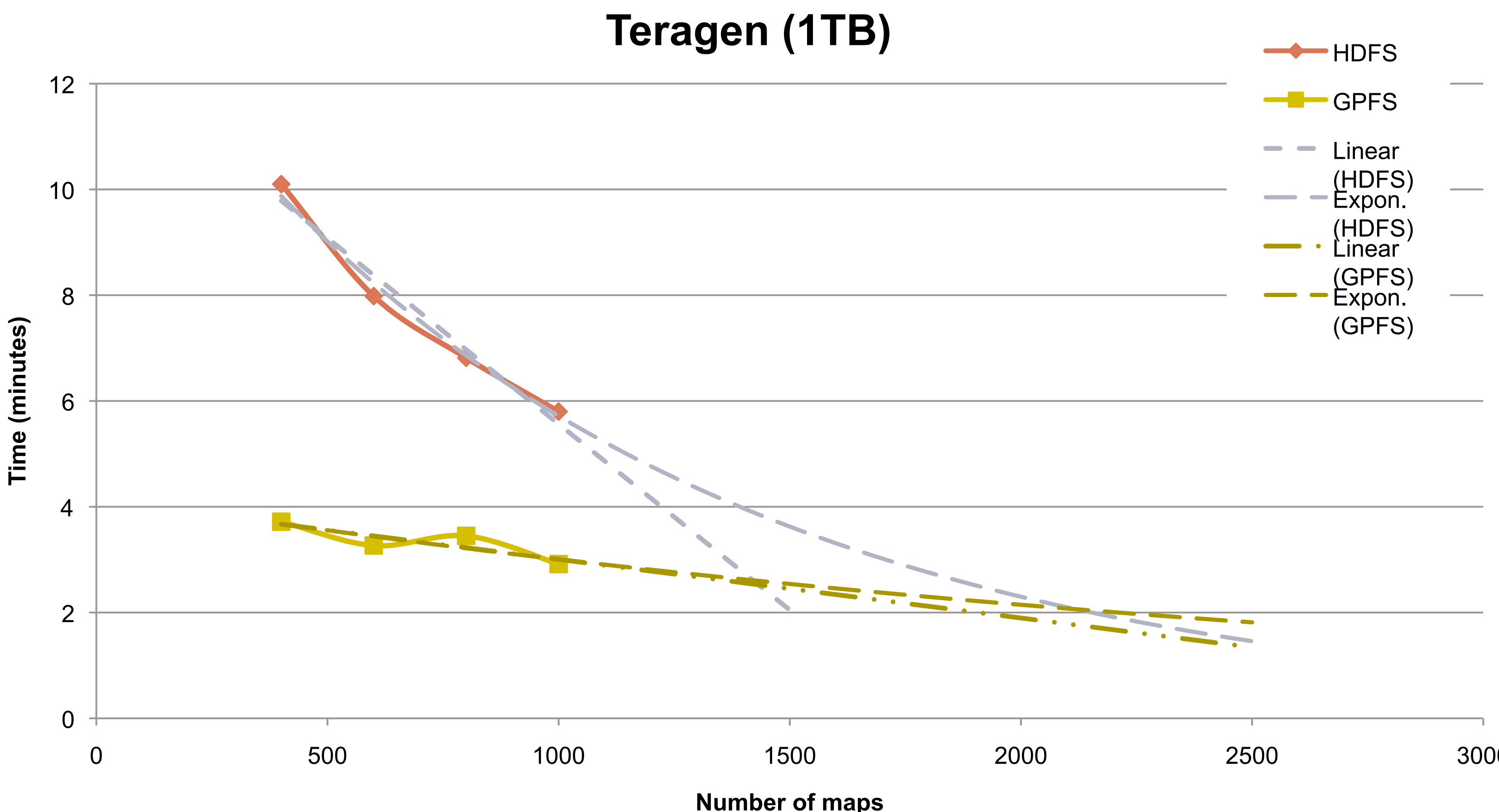
- **When: TBD**
- **Send us email if you are interested**
  - LRamakrishnan@lbl.gov
  - Scanon@lbl.gov

- **Include a brief description of your application.**

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

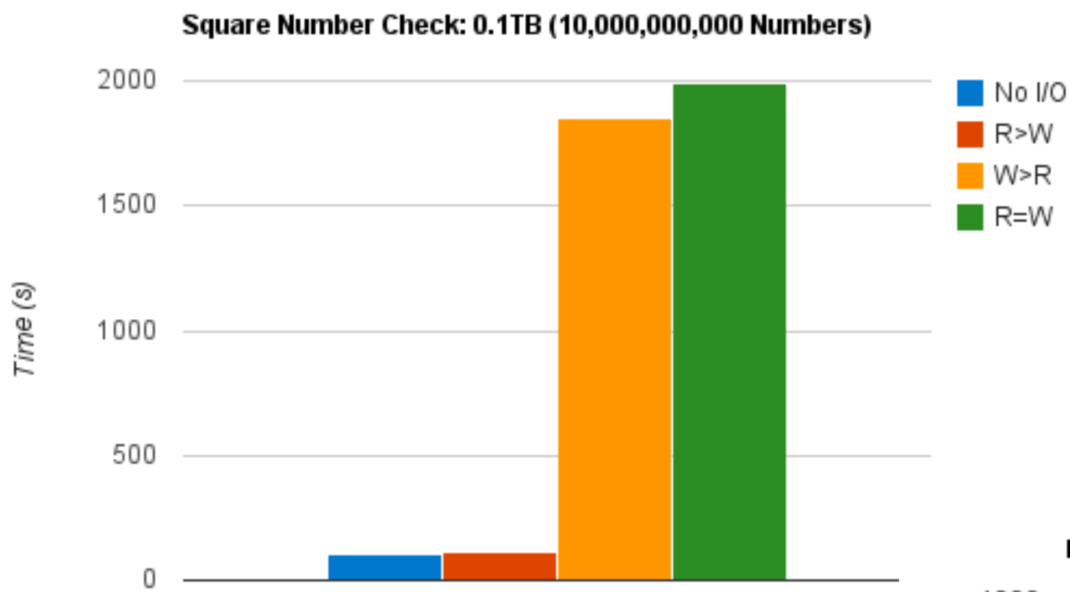
HDFS vs GPFS (Time)



U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

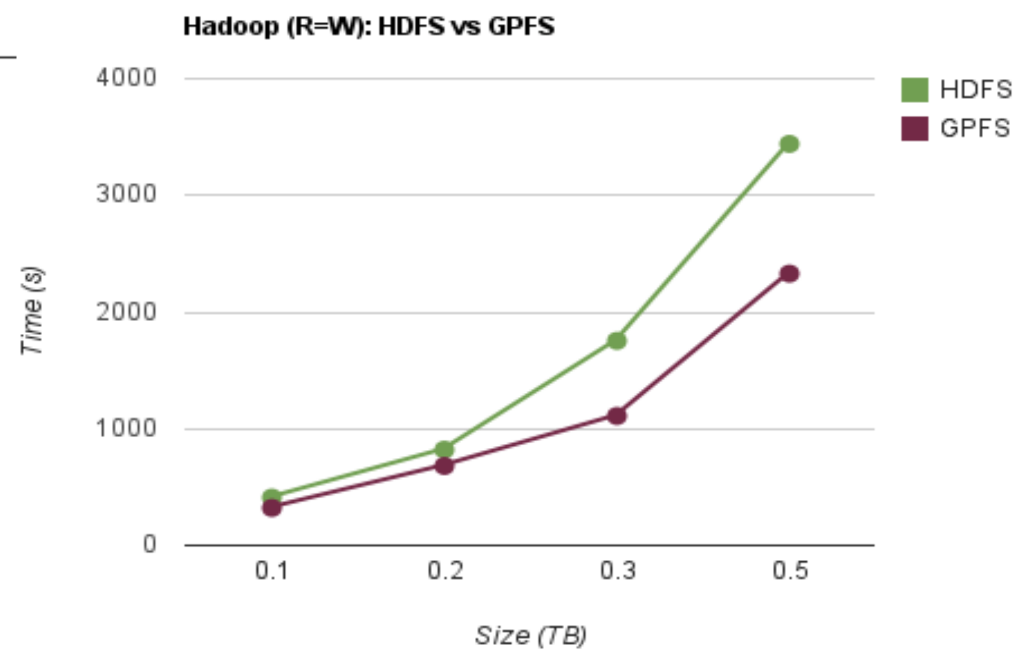
Application Characteristic Affect Choices



Wikipedia data set

On ~ 75 nodes, GPFS performs better with large nodes

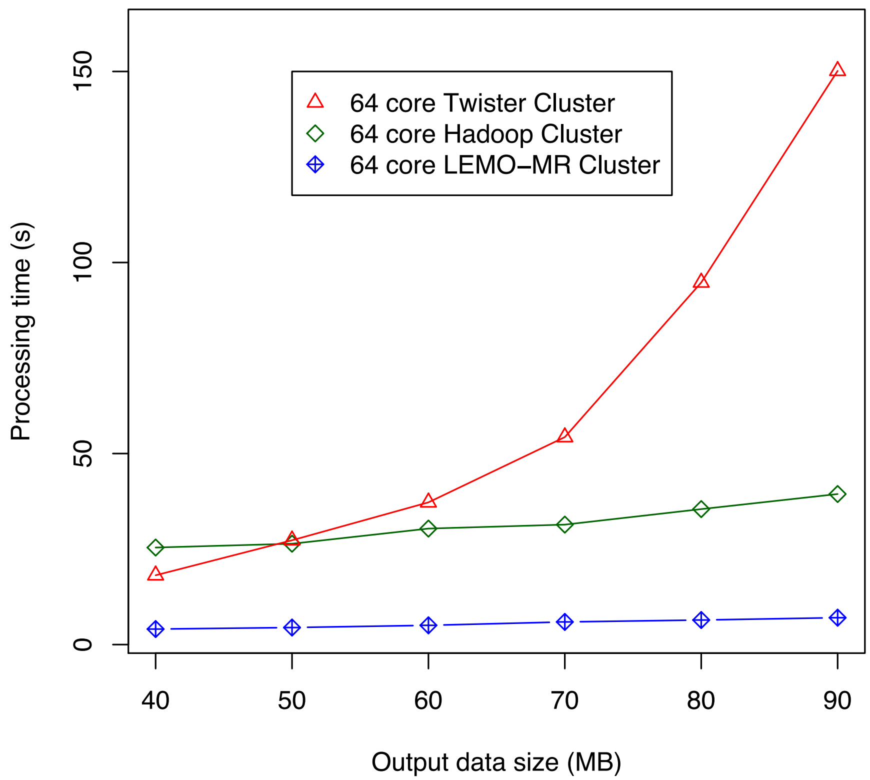
- Identical data loads and processing load
- Amount of writing in application affects performance



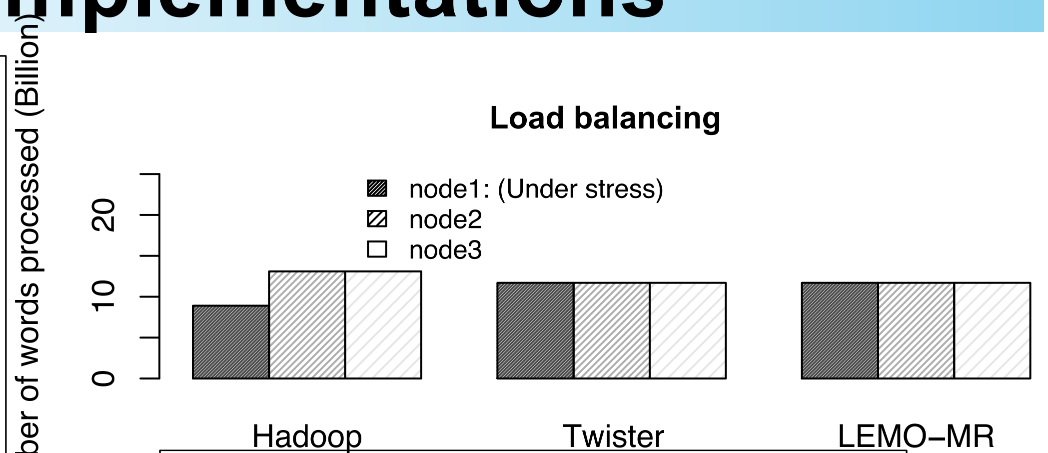
Hadoop: Challenges

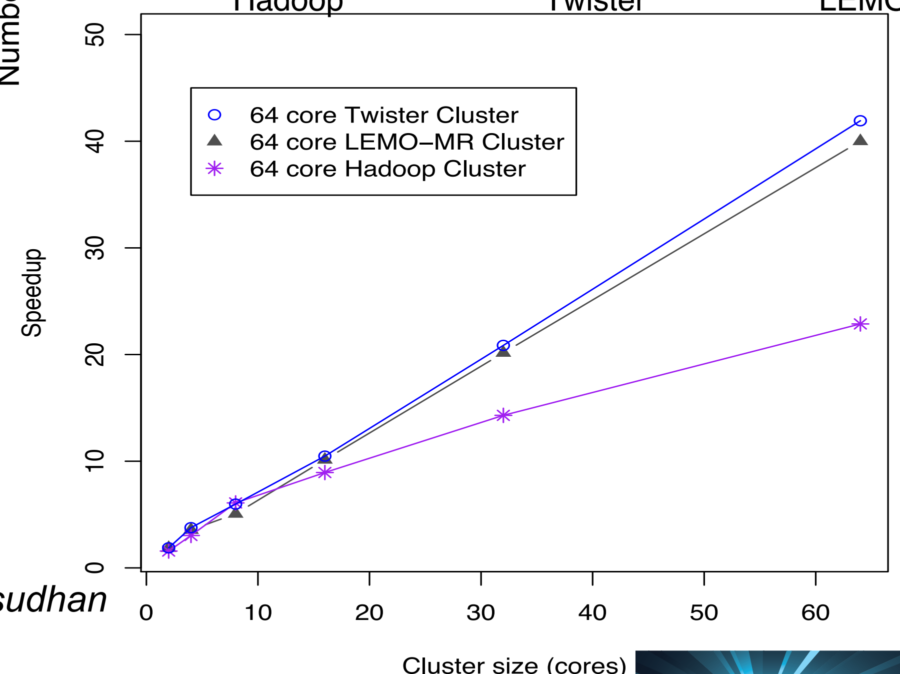
- **Deployment**
  - all jobs run as user “hadoop” affecting file permissions
  - less control on how many nodes are used - affects allocation policies
- **Programming: No turn-key solution**
  - using existing code bases, managing input formats and data
- **Additional benchmarking, tuning needed, Plug-ins for Science**

Comparison of MapReduce Implementations



Producing random floating point numbers





Processing 5 million 33 x 33 matrices

Collaboration w/ Zacharia Fadika, Elif Dede, Madhusudhan Govindaraju, SUNY Binghamton

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

Programming Hadoop

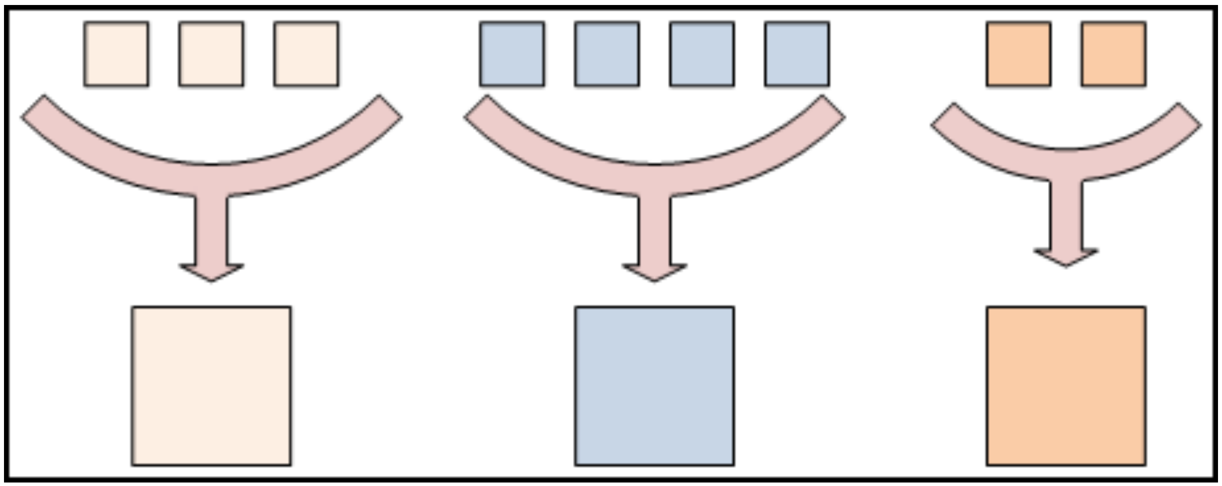
Programming with Hadoop

- Map and reduce as Java programs using Hadoop API
- Pipes and Streaming can help with existing applications in other languages
- C- HDFS API
- Higher-level languages such as Pig might help with some applications

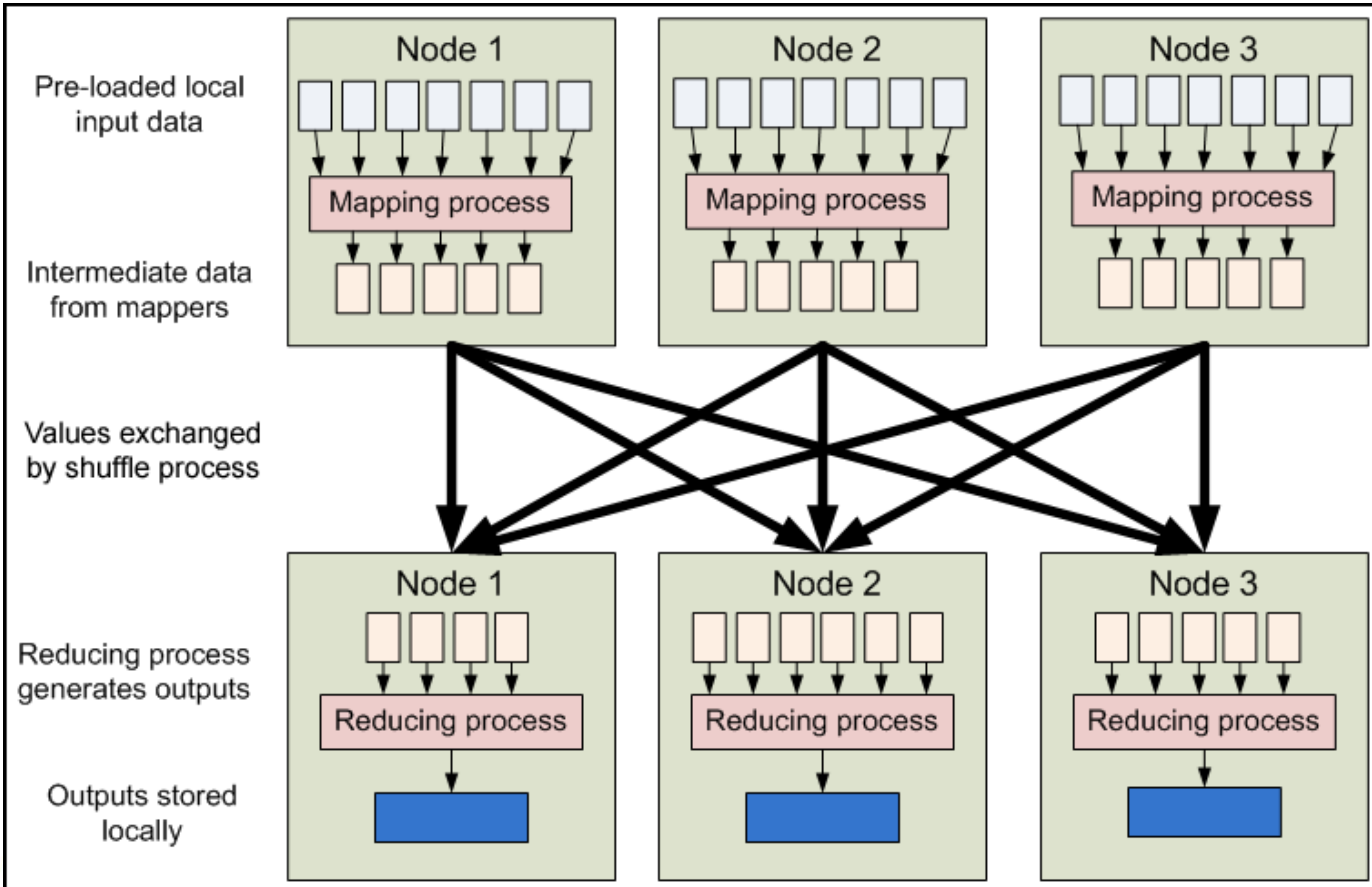
Keys and Values

- **Maps and reduces produce key-value pairs**
  - arbitrary number of values can be output
  - may map one input to 0,1, ….100 outputs
  - reducer may emit one or more outputs
- **Example: Temperature recordings**
  - 94089 8:00 am, 59
  - 27704 6:30 am, 70
  - 94089 12:45 pm, 80
  - 47401   1 pm, 90

Keys divide the reduce space



Data Flow


Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Mapping process
Mapping process
Node 1
Node 2
Node 3
Reducing process
Reducing process
Reducing process

Mechanics[1/2]

- Input files
  - large 10s of GB or more, typically in HDFS
  - line-based, binary, multi-line, etc.
- InputFormat
  - function defines how input files are split up and read
  - TextInputFormat (default), KeyValueInputFormat, SequenceFileInputFormat
- InputSplits
  - unit of work that comprises a single map task
  - FileInputFormat divides it into 64MB chunks

Mechanics [2/2]

- RecordReader
  - loads data and converts to key value pair
- Sort & Partiton & Shuffle
  - intermediate data from map to reducer
- Combiner
  - reduce data on a single machine
- Mapper & Reducer
- OutputFormat, RecordWriter

Word Count Mapper

```
public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

Word Count Reducer

```
public static class IntSumReducer
   extends Reducer<Text,IntWritable,Text,IntWritable> {
   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
     int sum = 0;
     for (IntWritable val : values) {
       sum += val.get();
     }
     result.set(sum);
     context.write(key, result);
   }
 }
```

Word Count Example

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
   ….
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);

}
```

Pipes

- Allows C++ code to be used for Mapper and Reducer
- Both key and value inputs to pipes programs are provided as std::string
- $ hadoop pipes

C-HDFS API

- **Limited C API to read and write from HDFS**

```
#include "hdfs.h"
int main(int argc, char **argv)
{
  hdfsFS fs = hdfsConnect("default", 0);
  hdfsFile writeFile = hdfsOpenFile(fs, writePath, O_WRONLY|O_CREAT, 0, 0, 0);
  tSize num_written_bytes = hdfsWrite(fs, writeFile, (void*)buffer, strlen(buffer)+1);
  hdfsCloseFile(fs, writeFile);
}
```

Hadoop Streaming

- Generic API that allows programs in any language to be used as Hadoop Mapper and Reducer implementations
- Inputs written to stdin as strings with tab character separating
- Output to stdout as key \t value \n
- $ hadoop jar contrib/streaming/hadoop-[version]-streaming.jar

Debugging

- Test core functionality separate
- Use Job Tracker
- Run “local” in Hadoop
- Run job on a small data set on a single node
- Hadoop can save files from failed tasks

Pig – Basic Operations

- LOAD – loads data into a relational form
- FOREACH..GENERATE – Adds or removes fields (columns)
- GROUP – Group data on a field
- JOIN – Join two relations
- DUMP/STORE – Dump query to terminal or file

There are others, but these will be used for the exercises today

Pig Example

Find the number of gene hits for each model in an hmmsearch (>100GB of output, 3 Billion Lines)

bash# cat * |cut –f 2|sort|uniq -c

```
> hits = LOAD '/data/bio/*' USING PigStorage() AS
  (id:chararray,model:chararray, value:float);
> amodels = FOREACH hits GENERATE model;
> models = GROUP amodels BY model;
> counts = FOREACH models GENERATE group,COUNT
  (amodels) as count;
> STORE counts INTO 'tcounts' USING PigStorage();
```

Pig - LOAD

Example:

```
hits = LOAD 'load4/*' USING PigStorage() AS
  (id:chararray, model:chararray,value:float);
```

- **Pig has several built-in data types (chararray, float, integer)**
- **PigStorage can parse standard line oriented text files.**
- **Pig can be extended with custom load types written in Java.**
- **Pig doesn't read any data until triggered by a DUMP or STORE**

Pig – FOREACH..GENERATE, GROUP

Example:

```
amodel = FOREACH model GENERATE hits;
models = GROUP amodels BY model;
counts = FOREACH models GENERATE group,COUNT
   (amodels) as count;
```

- Use FOREACH..GENERATE to pick of specific fields or generate new fields. Also referred to as a projection
- GROUP will create a new record with the group name and a "bag" of the tuples in each group
- You can reference a specific field in a bag with <bag>.field (i.e. amodels.model)
- You can use aggregate functions like COUNT, MAX, etc on a bag

Pig – Important Points

- Nothing really happens until a DUMP or STORE is performed.
- Use FILTER and FOREACH early to remove unneeded columns or rows to reduce temporary output
- Use PARALLEL keyword on GROUP operations to run more reduce tasks

Questions?

- **Shane Canon**
  - Scanon@lbl.gov
- **Lavanya Ramakrishnan**
  - LRamakrishnan@lbl.gov