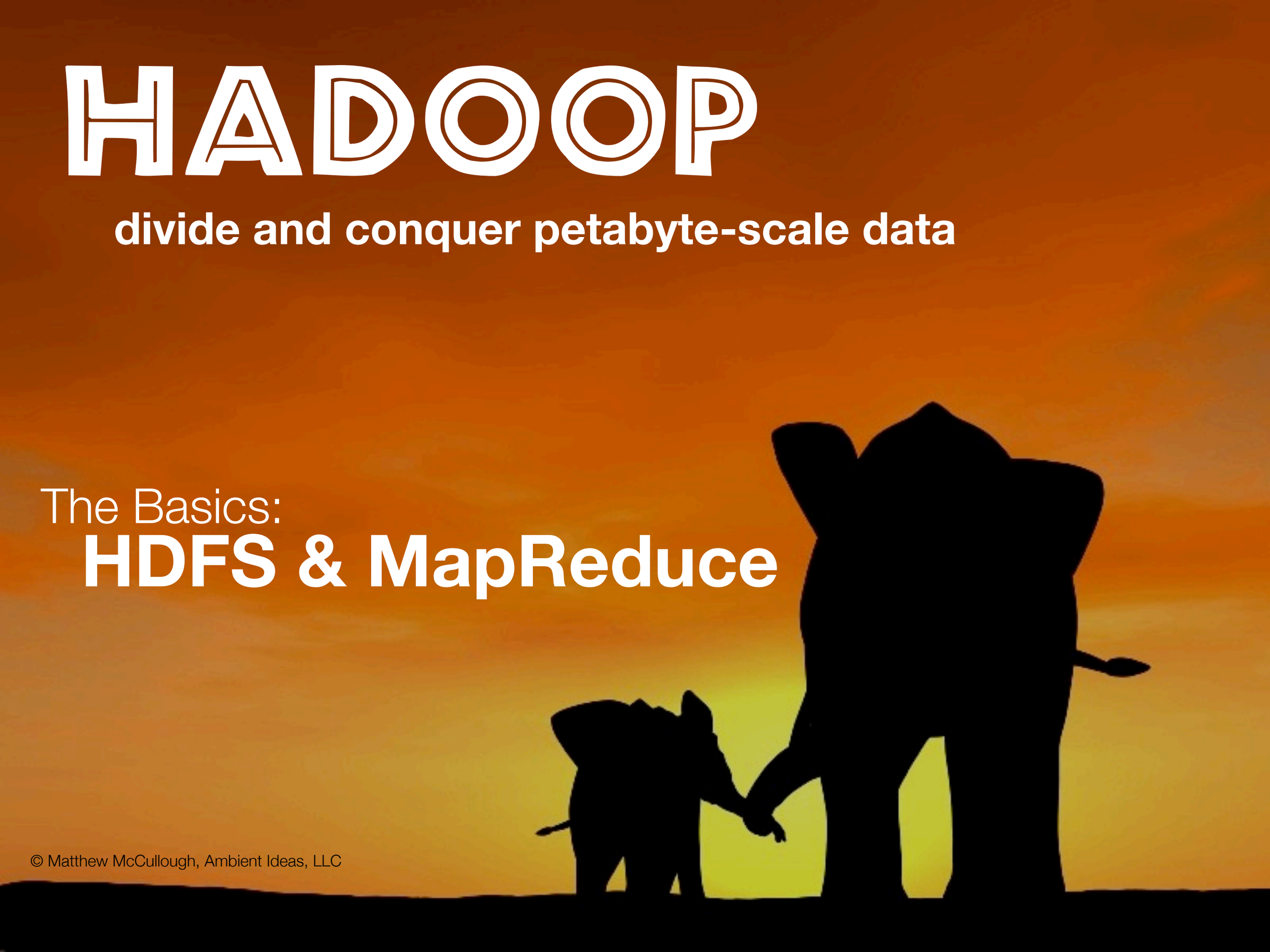
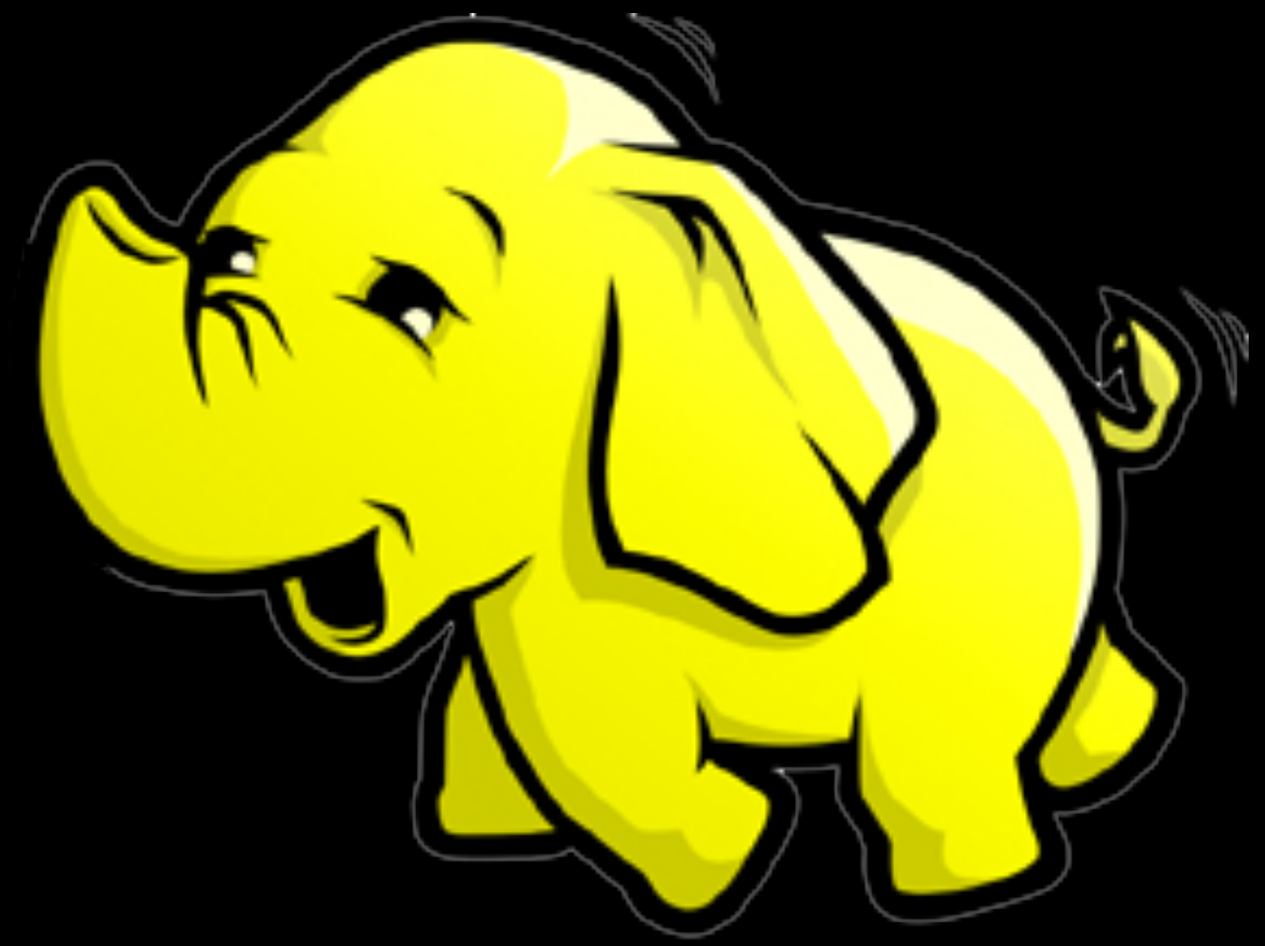
HADOOP

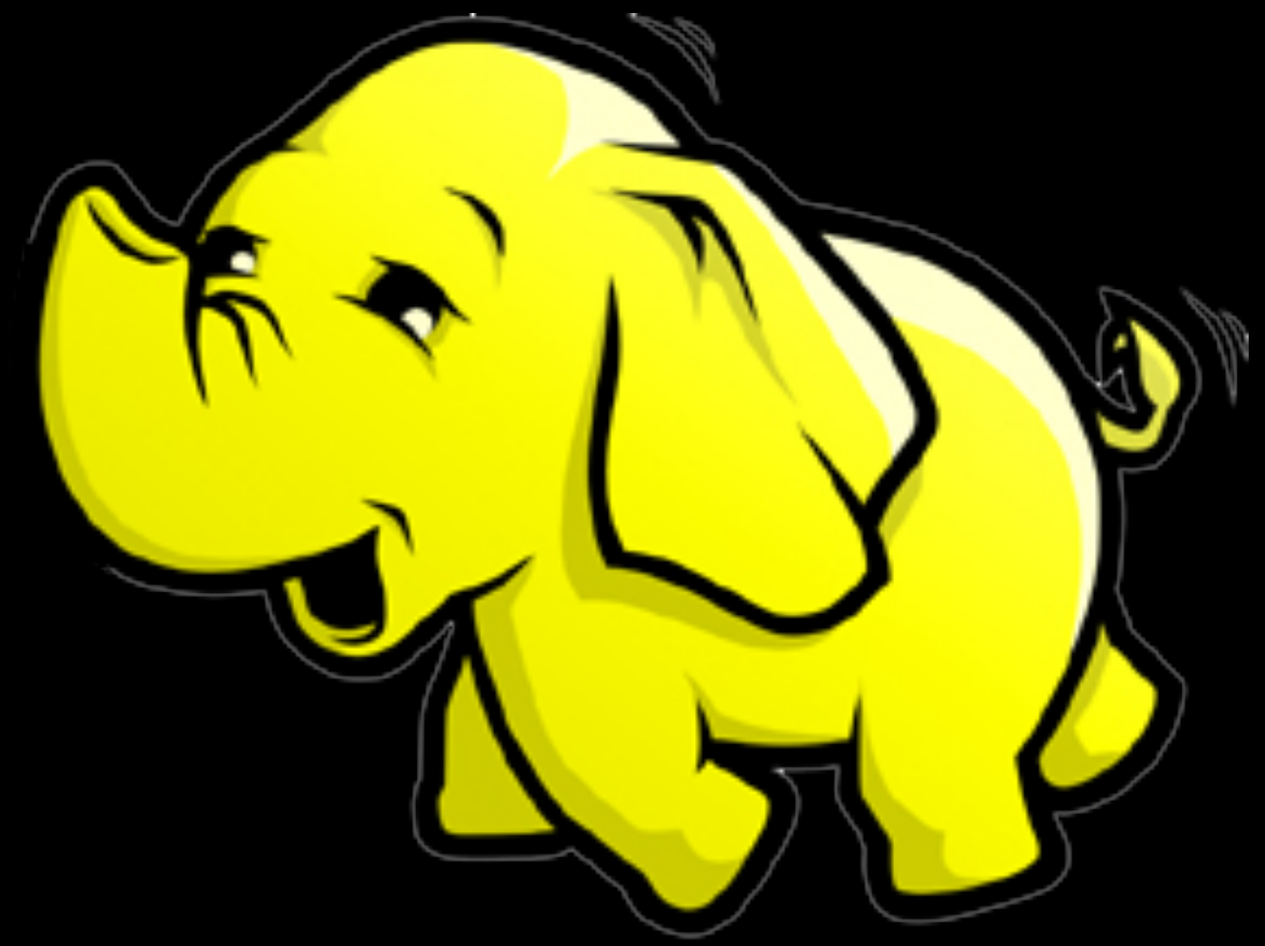
divide and conquer petabyte-scale data



The Basics:

HDFS & MapReduce






hadoop

METADATA

Matthew McCullough
Ambient Ideas, LLC
matthewm@ambientideas.com
http://ambientideas.com/blog
@matthewmccull

Code

http://github.com/matthewmccullough/hadoop-intro

Matthew McCullough's hadoop Bookmarks

See all hadoop bookmarks in **Popular**, **Recent**, or your **Network**.

matthew.mccullough | hadoop ✖ | Type another tag | Bookmarks 30

Display options

27 FEB 10 **Hadoop PigLatin** 27
EDIT | DELETE — commandline, reference, opensource, scalability, grid, hadoop

Hadoop Hive Language Manual, CLI 4
EDIT | DELETE — commandline, reference, opensource, scalability, grid, hadoop

24 FEB 10 **Facebook's Petabyte Scale Data Warehouse using Hive and Hadoop** 58
EDIT | DELETE — facebook, scalability, cloud, programming, mapreduce, hadoop

23 FEB 10 **Sort Benchmark, TeraSort Home Page** 40
Results of
EDIT | DELETE — programming, scalability, mapreduce, Sorting, competition, hadoop

25 JAN 10 **MyNoSQL - Software Patents, MapReduce and Open Source**
EDIT | DELETE — mapreduce, patent, google, hadoop

19 JAN 10 **United States Patent: 7650331 - MapReduce** 10
EDIT | DELETE — mapreduce, google, scalability, patent, hadoop

06 JAN 10 **Notes on Distributed Programming and CAP « MyNoSQL** 8
EDIT | DELETE — distributed, cloud, slides, opensource, programming, scalability, hadoop

20 DEC 09 **Hadoop Training: Virtual Machine | Cloudera** 41
EDIT | DELETE — google, tutorial, mapreduce, programming, virtualmachine, hadoop

Distributed Systems - Google Code University 356
google, tutorial, mapreduce, programming, hadoop

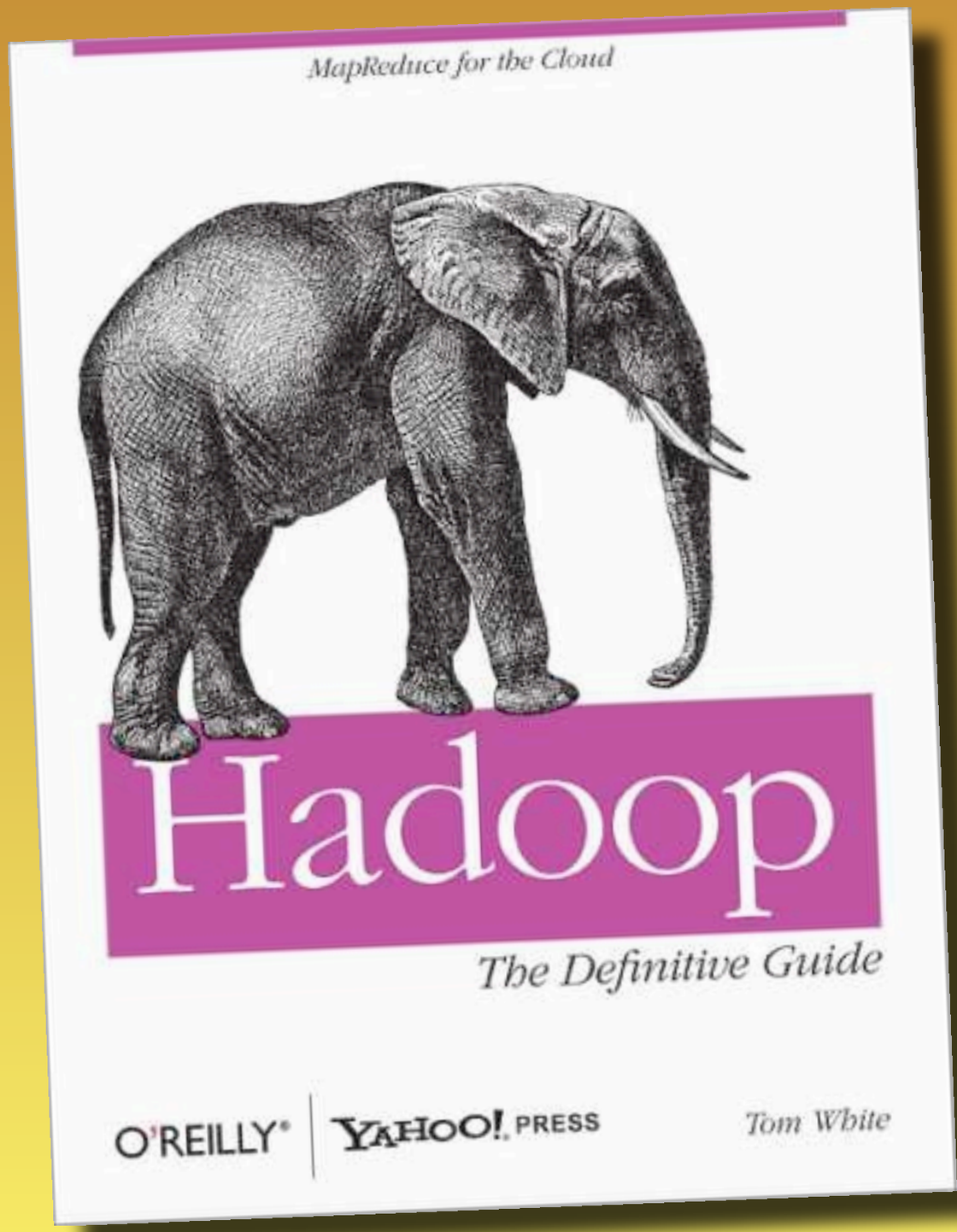
University of Washington: Problem Solving on Large Scale Clusters - Google Code University 38
EDIT | DELETE — google, tutorial, mapreduce, programming, education, hadoop

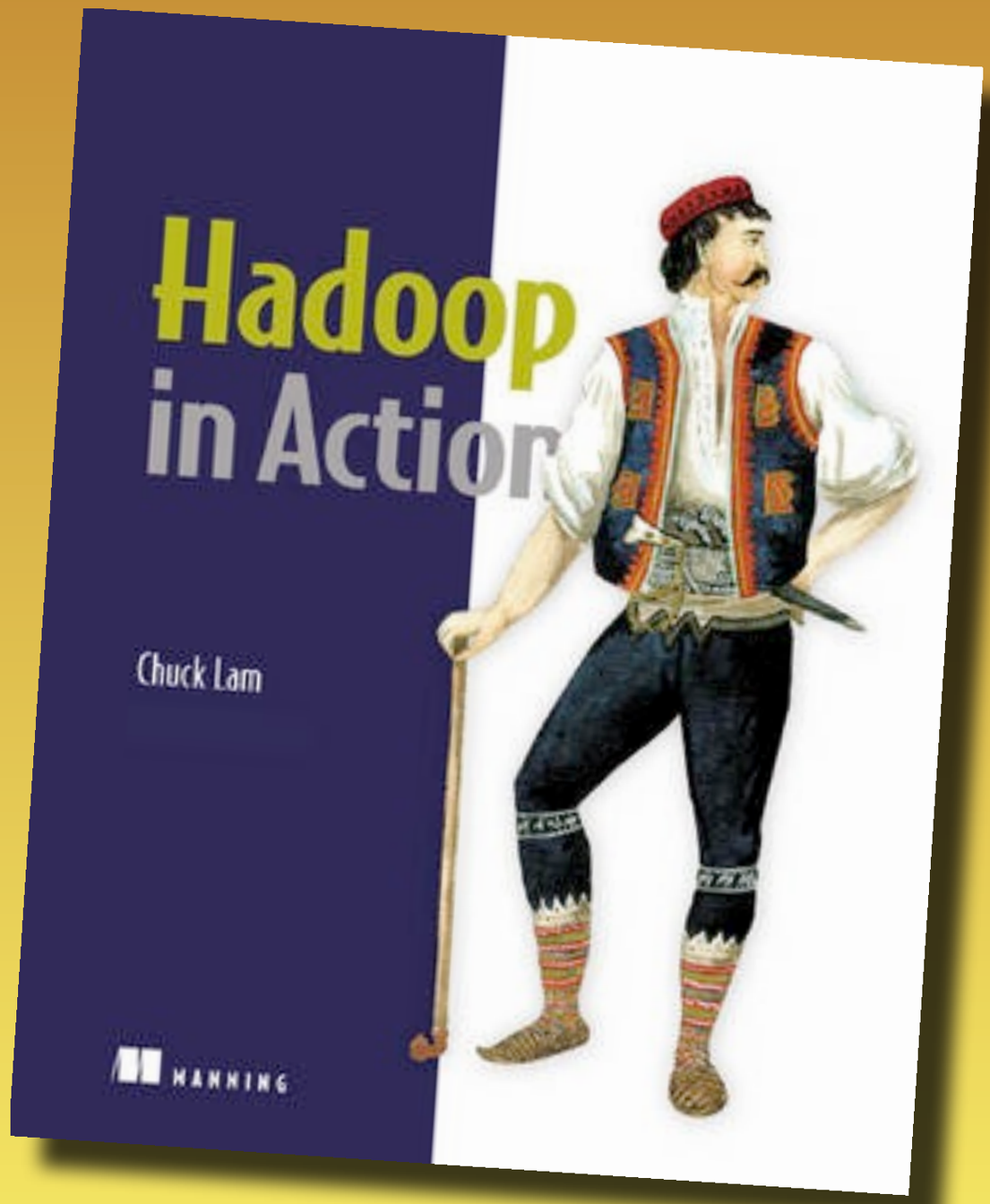
Tags | People | Options

Related Tags 29

+amazon	1
+bigtable	1
+clojure	1
+cloud	12
+commandline	2
+competition	1
+database	4
+distributed	1
+education	1
+facebook	1
+gae	3
+google	10
+grid	3
+java	2
+jdo	1
+mapreduce	23
+news	1
+opensource	6
+patent	2
+presentation	7
+programming	15
+reference	2
+research	1
+scalability	16
+slides	1
+tutorial	6
+video	1
+virtualmachine	1

http://delicious.com/matthew.mccullough/hadoop


MapReduce for the Cloud
Hadoop
The Definitive Guide
O'REILLY
YAHOO! PRESS
Tom White


Hadoop
in Actio
Chuck Lam
MANNING

Why Hadoop?

HDFS

MapReduce

YOUR BIG DATA

- Data type?
- Business need?
- Processing?
- How large?
- Growth projections?
- What if you saved everything?

Why Hadoop?

HDFS

MapReduce

Why Hadoop?

I use Hadoop often.

I use Hadoop often.

That's the sound my Tivo makes every time I skip a commercial.

-**Brian Goetz**, author of Java Concurrency in Practice


BIG DATA

NO SQL

NO SQL

NO SQL

Not SQL

Not SQL

Not Only SQL

NOSQL

- Applies to data
  - No strong schemas
  - No foreign keys
- Applies to processing
  - No SQL-99 standard
  - No execution plan

The computer you are using
right now
may very well have the
fastest GHz processor
you`ll ever own

Scale up?


Scale out

WEB CRAWLING

Doug Cutting

open source

MARKETING!

NAME

RESEARCH!

DAUGHTER'S STUFFED TOY

Lucene

Lucene

↓

Nutch

Lucene

↓

Nutch

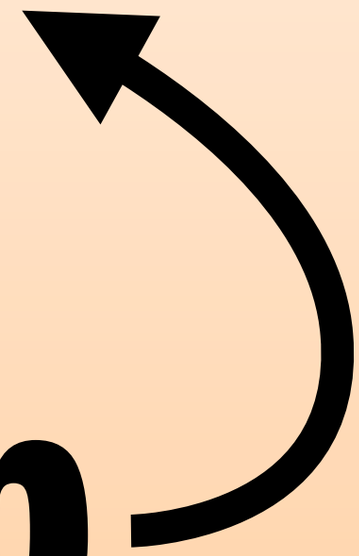
↓

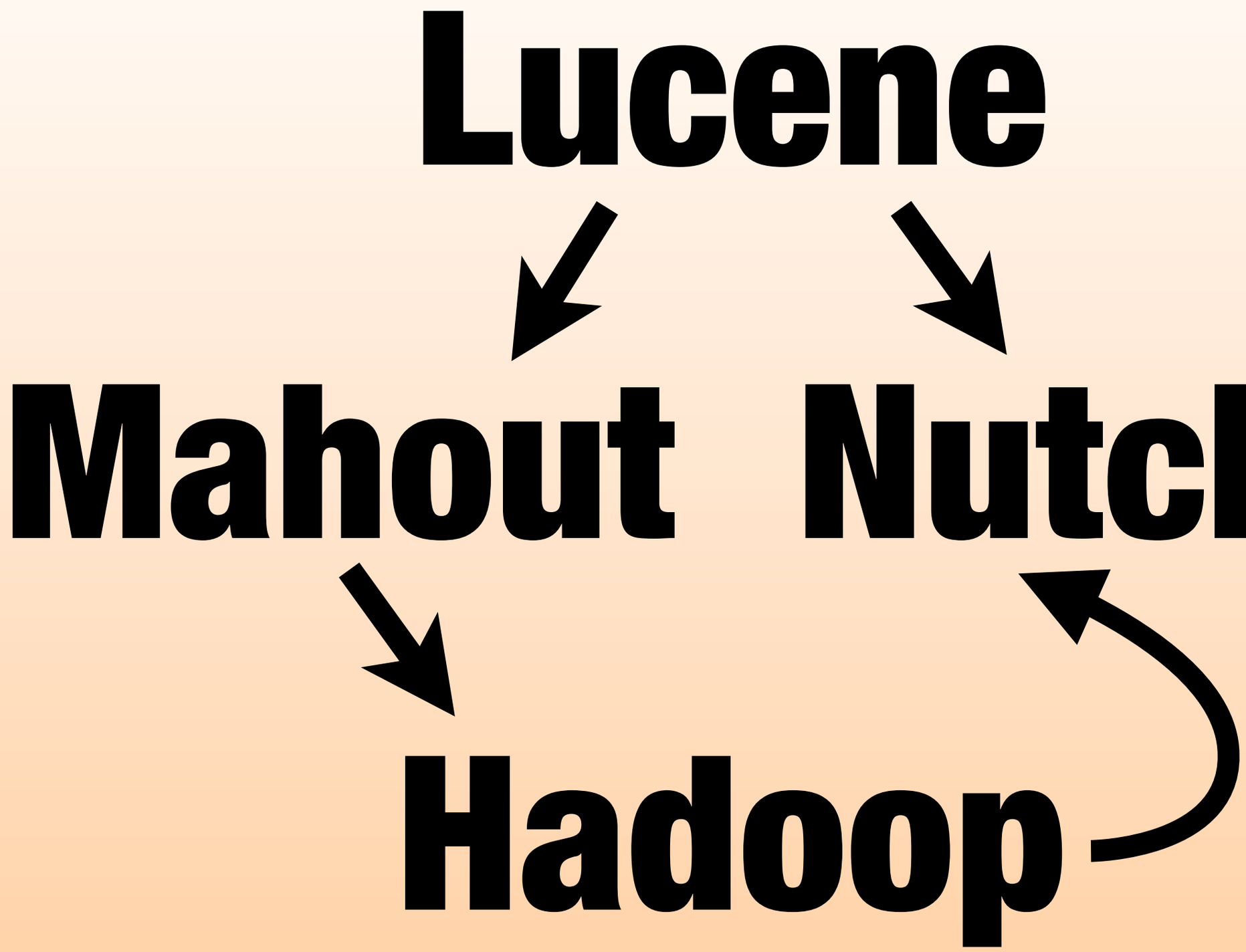
Hadoop

Lucene

Nutch

Hadoop

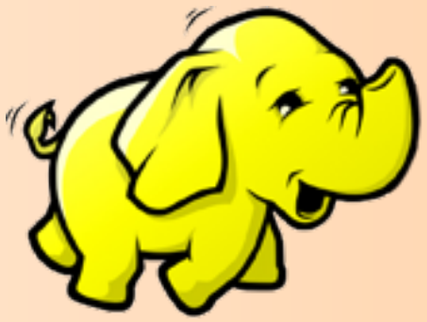



Lucene
Mahout
Nutch
Hadoop

YAHOO!®


TODAY

- **0.21.0** current version



- **Dozens** of companies contributing

- **Hundreds** of companies using

Applications and organizations using Hadoop include (alphabetically):

- A9.com - Amazon
  - We build Amazon's product search indices using the streaming API and pre-existing C++, Perl, and Python tools.
  - We process millions of sessions daily for analytics, using both the Java and streaming APIs.
  - Our clusters vary from 1 to 100 nodes.
- Adobe
  - We use Hadoop and HBase in several areas from social services to structured data storage and processing for internal use.
  - We currently have about 30 nodes running HDFS, Hadoop and HBase in clusters ranging from 5 to 14 nodes on both production and development. We plan a deployment on an 80 nodes cluster.
  - We constantly write data to HBase and run MapReduce jobs to process then store it back to HBase or external systems.
  - Our production cluster has been running since Oct 2008.
- Able Grape - Vertical search engine for trustworthy wine information

- We have one of the world's smaller Hadoop clusters (2 nodes @ 8 CPUs/node)
- Hadoop and Nutch used to analyze and index textual information

- Adknowledge - Ad network
  - Hadoop used to build the recommender system for behavioral targeting, plus other clickstream analytics
  - We handle 500MM clickstream events per day
  - Our clusters vary from 50 to 200 nodes, mostly on EC2.
  - Investigating use of R clusters atop Hadoop for statistical analysis and modeling at scale.

- Alibaba
  - A 15-node cluster dedicated to processing sorts of business data dumped out of database and joining them together. These data will then be fed into iSearch, our vertical search engine.
  - Each node has 8 cores, 16G RAM and 1.4T storage.

- Amazon Web Services
  - We provide Amazon Elastic MapReduce. It's a web service that provides a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).
  - Our customers can instantly provision as much or as little capacity as they like to perform data-intensive tasks for applications such as web indexing, data mining, log file analysis, machine learning, financial analysis, scientific simulation, and bioinformatics research.

- AOL
  - We use hadoop for variety of things ranging from ETL style processing and statistics generation to running advanced algorithms for doing behavioral

We are one of six universities participating in IBM/Google's academic cloud computing initiative. Ongoing research and teaching efforts include projects in machine translation, language modeling, bioinformatics, email analysis, and image processing.

- University of Nebraska Lincoln, Research Computing Facility

  We currently run one medium-sized Hadoop cluster (200TB) to store and serve up physics data for the computing portion of the Compact Muon Solenoid (CMS) experiment. This requires a filesystem which can download data at multiple Gbps and process data at an even higher rate locally. Additionally, several of our students are involved in research projects on Hadoop.

- Veoh
  - We use a small Hadoop cluster to reduce usage data for internal metrics, for search indexing and for recommendation data.

- Visible Measures Corporation uses Hadoop as a component in our Scalable Data Pipeline, which ultimately powers VisibleSuite and other products. We use Hadoop to aggregate, store, and analyze data related to in-stream viewing behavior of Internet video audiences. Our current grid contains more than 128 CPU cores and in excess of 100 terabytes of storage, and we plan to grow that substantially during 2008.

- VK Solutions
  - We use a small Hadoop cluster in the scope of our general research activities at VK Labs to get a faster data access from web applications.
  - We also use Hadoop for filtering and indexing listing, processing log analysis, and for recommendation data.

- Vuelos baratos
  - We use a small Hadoop

- Vuelos baratos
  - We use a small Hadoop
- WorldLingo
  - Hardware: 44 servers (each server has: 2 dual core CPUs, 2TB storage, 8GB RAM)
  - Each server runs Xen with one Hadoop/HBase instance and another instance with web or application servers, giving us 88 usable virtual machines.
  - We run two separate Hadoop/HBase clusters with 22 nodes each.
  - Hadoop is primarily used to run HBase and Map/Reduce jobs scanning over the HBase tables to perform specific tasks.
  - HBase is used as a scalable and fast storage back end for millions of documents.
  - Currently we store 12million documents with a target of 450million in the near future.
- Yahoo!
  - More than 100,000 CPUs in >25,000 computers running Hadoop
  - Our biggest cluster: 4000 nodes (2*4cpu boxes w 4*1TB disk & 16GB RAM)
    - Used to support research for Ad Systems and Web Search
    - Also used to do scaling tests to support development of Hadoop on larger clusters
  - Our Blog - Learn more about how we use Hadoop.
  - >40% of Hadoop Jobs within Yahoo are Pig jobs.
- Zvents
  - 10 node cluster (Dual-Core AMD Opteron 2210, 4GB RAM, 1TB/node storage)
  - Run Naive Bayes classifiers in parallel over crawl data to discover event information

The New York Times
Expect the World®
YAHOO!
last.fm
rackspace
IBM
FOX
AOL
krugle
Sears
Google
facebook
cloudera
joost
hulu
LinkedIn

Bing

Why Hadoop?

HDFS

MapReduce

HDFS

VIRTUAL MACHINE

VM SOURCES

- **Yahoo**
  - True to the OSS distribution
- **Cloudera**
  - Desktop tools
- Both VMWare based

STARTING UP

Tailing the logs

Terminal — zsh — zsh

zsh | zsh | bash | zsh | zsh

```
Zeus /Applications/Dev/hadoop-family/hadoop-0.20.1/logs
24 logs %                                                    11:14:31
```

FILESYSTEM

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

Categories and Subject Descriptors

D [4]: 3—*Distributed file systems*

General Terms

Design, reliability, performance, measurement

Keywords

Fault tolerance, scalability, data storage, clustered storage

*The authors can be reached at the following addresses: {sanjay,hgobioff,shuntak}@google.com.

SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies. Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system.

Second, files are huge by traditional standards. Multi-GB files are common. Each file typically contains many application objects such as web documents. When we are regularly working with fast growing data sets of many TBs comprising billions of objects, it is unwieldy to manage billions of approximately KB-sized files even when the file system could support it. As a result, design assumptions and parameters such as I/O operation and block sizes have to be revisited.

Third, most files are mutated by appending new data rather than overwriting existing data. Random writes within a file are practically non-existent. Once written, the files are only read, and often only sequentially. A variety of data share these characteristics. Some may constitute large repositories that data analysis programs scan through. Some may be data streams continuously generated by running applications. Some may be archival data. Some may be intermediate results produced on one machine and processed on another, whether simultaneously or later in time. Given this access pattern on huge files, appending becomes the focus of performance optimization and atomicity guarantees, while caching data blocks in the client loses its appeal.

Fourth, co-designing the applications and the file system API benefits the overall system by increasing our flexibility.

"scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware"

HDFS BASICS

- Open Source implementation of **Google** BigTable
- **Replicated** data store
- Stored in 64MB blocks

HDFS

- Rack **location aware**
- Configurable redundancy factor
- Self-healing
- Looks almost like *NIX filesystem

WHY HDFS?

- Random reads
- Parallel reads
- Redundancy

DATA OVERLOAD

DATA OVERLOAD

overflow from traditional RDBMS or log files

destination?

DATA OVERLOAD

overflow from traditional RDBMS or log files

destination?

DATA OVERLOAD

overflow from traditional RDBMS or log files

destination?

DATA OVERLOAD

overflow from traditional RDBMS or log files

destination?

HDFS File System Shell Guide

TEST HDFS

- Upload a file
- List directories

Lab

HDFS UPLOAD

- Show contents of file in HDFS
- Show vocabulary of HDFS

Lab

HDFS CHALLENGES

- Writes are re-writes today
  - Append is planned
- Block size, alignment
- Small file inefficiencies
- NameNode SPOF

Why Hadoop?

HDFS

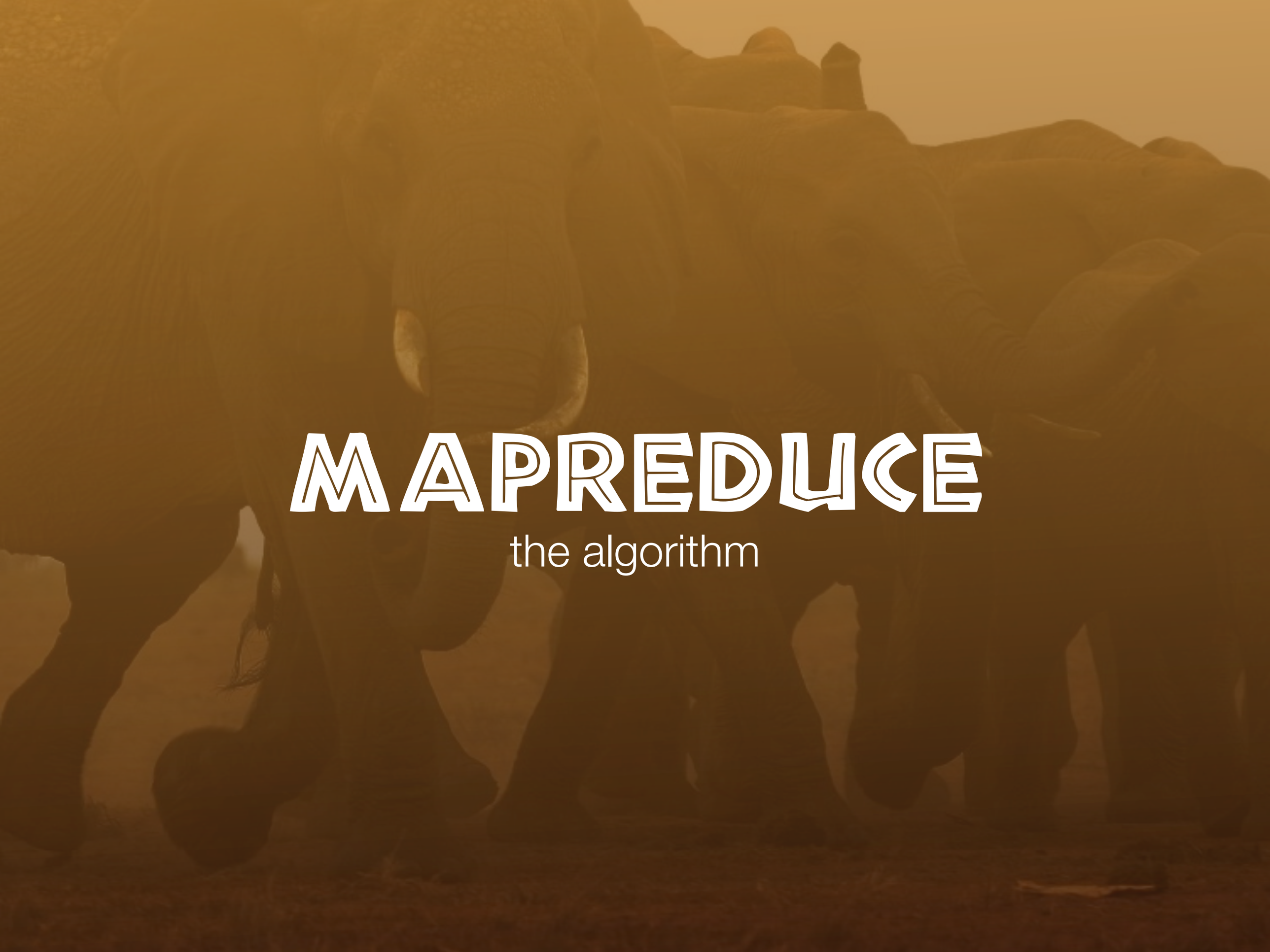
MapReduce

MapReduce

MapReduce is
functional programming
on a
distributed
processing platform

HADOOP'S PREMISE

- Geography-aware distribution of brute-force processing



MAPREDUCE

the algorithm

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapRe gramming model and an ated i essing and gene d function th sers spec e pair to generat interme s a nd a *reduce* functio rges value associated with the s me diate rld tasks are express Many aper. shown

ams written in this f style mati- llelized and exec arge clu m- chines. The r stem takes data, scheduling t of machines, handling ma- gram's e chine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

1 Introduction

Over the past five years, the authors and many others at Google have implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a

given day, etc. Most such computations are conceptu- straightforward. Howeve lly he computation hu housands sh in a reas ount The issues o par- allelize utati bute the data, failures e original sim u- tation wit mplex code ith these issu

As a re this co a new abstractio ows us to ex computa- tions w ing to perform but messy de- tail ization, fault-tolerance ution alancing in a li in- plied by the *map* and and many other fun guages. We reali most of our compu volved applying a *ma* eration to each log rd" in our input in o compute a set of int e key/value pairs. applying a *reduce* op d the same key, in order t ta ap- propriately. Our use of a fun del with user-specified map and reduce operations allows us to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance.

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.

Section 2 describes the basic programming model and gives several examples. Section 3 describes an implementation of the MapReduce interface tailored towards our cluster-based computing environment. Section 4 describes several refinements of the programming model that we have found useful. Section 5 has performance measurements of our implementation for a variety of tasks. Section 6 explores the use of MapReduce within Google including our experiences in using it as the basis

To appear in OSDI 2004

" A programming model and implementation for processing and generating large data sets"

VERIFY SETUP

- Launch "*cloudera-training*" VMWare instance
- Open a terminal window
- Verify hadoop

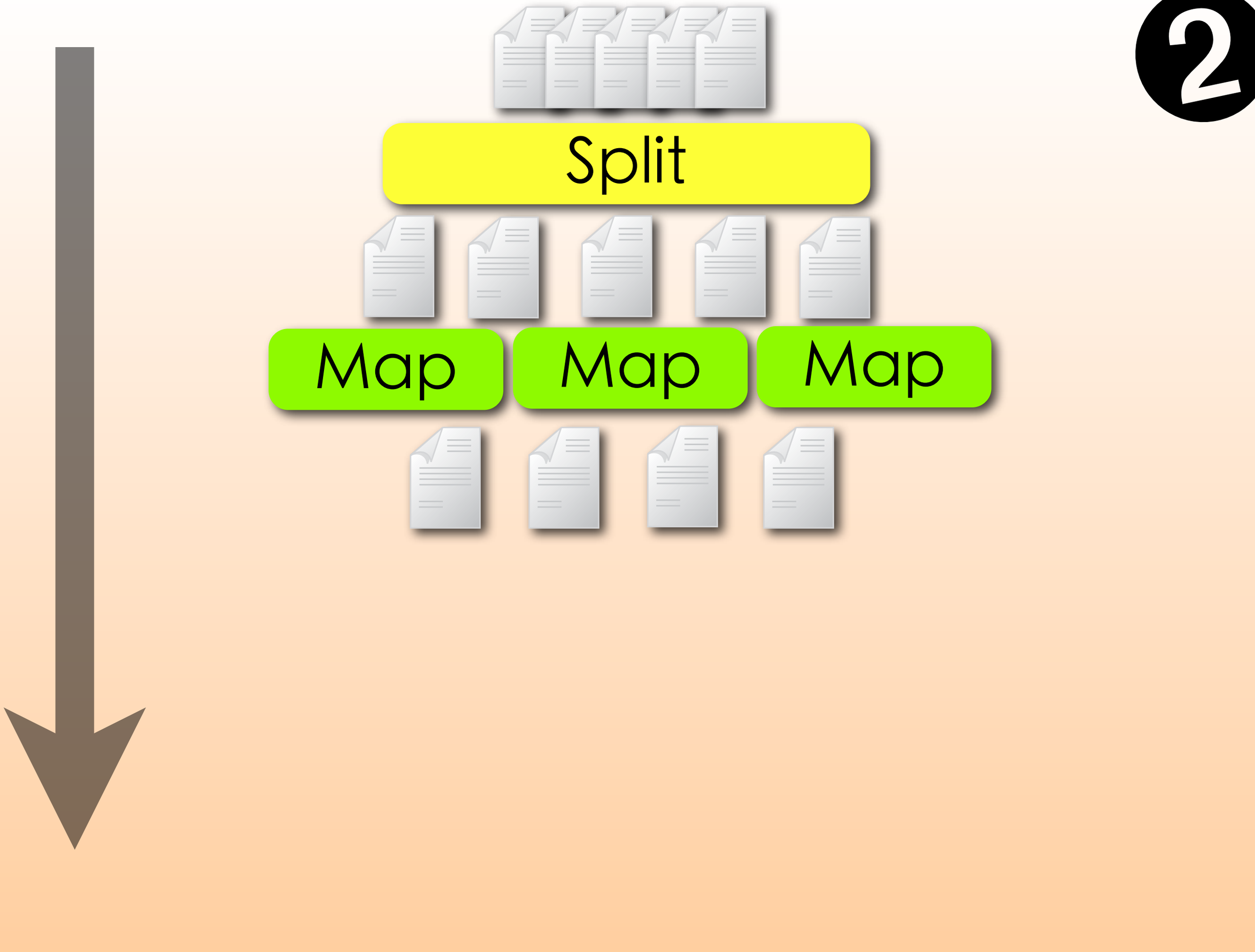
Lab

THE PROCESS

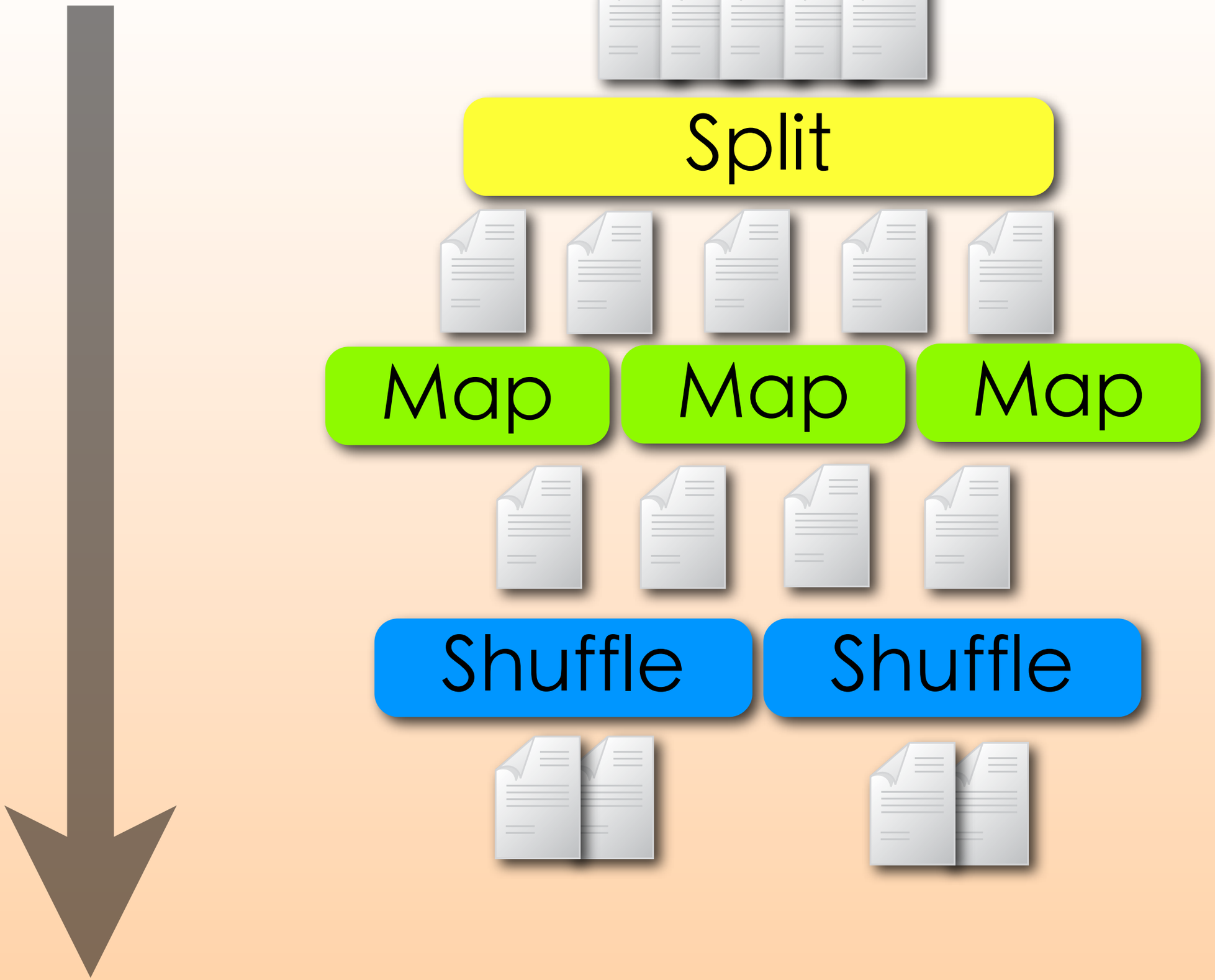
- **Map(k1,v1) -> list(k2,v2)**
  - Every item is parallel candidate for Map
- **Shuffle** (group) pairs from **all** lists by key
- **Reduce(k2, list (v2)) -> list(v3)**
  - Reduce in parallel on each group of keys

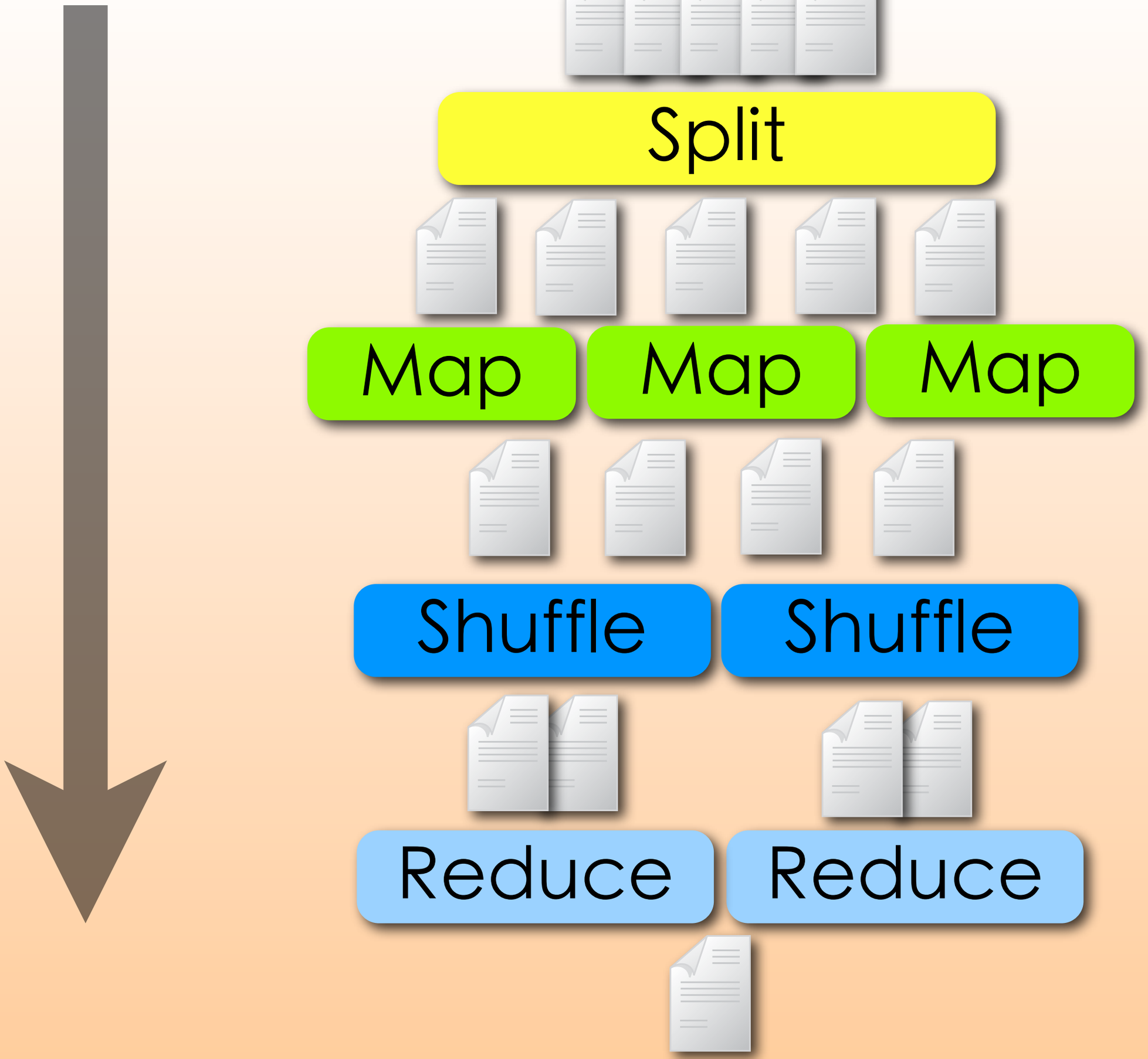
2

2


2
Split
Map
Map
Map

2




Split
Map
Map
Map
Shuffle
Shuffle
Reduce
Reduce

MAPREDUCE

- Run the grep Shakespeare job
- View the status consoles

Lab

MAPREDUCE

a word counting conceptual example

THE GOAL

Provide the **occurrence count** of each **distinct word** across **all documents**

RAW DATA

3

a folder of documents

mydoc1.txt

At four years old I acted out

mydoc2.txt

Glad I am not four years old

mydoc3.txt

Yet I still act like someone four

MAP

break documents into words

at	glad	yet
four	I	I
years	am	still
old	not	act
I	four	like
acted	years	someone
out	old	four

SHUFFLE

physically group (relocate) by key

at	glad	yet
four	I	not
four	I	still
four	I	act
old	am	like
old	years	someone
acted	years	out

REDUCE

count word occurrences

at = 1

four = 3

old = 2

acted = 1

glad = 1

I = 3

am = 1

years = 2

yet =1

not = 1

still = 1

act = 1

like = 1

someone = 1

out = 1

REDUCE AGAIN

sort occurrences alphabetically

act = 1

acted = 1

at = 1

am = 1

four = 3

glad = 1

I = 3

like = 1

not = 1

out = 1

old = 2

someone = 1

still = 1

years = 2

yet =1

GREP.JAVA

```
package org.apache.hadoop.examples;

import java.util.Random;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.*;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/* Extracts matching regexs from input files and counts them. */
public class Grep extends Configured implements Tool {
  private Grep() {}                               // singleton

  public int run(String[] args) throws Exception {
    if (args.length < 3) {
      System.out.println("Grep <inDir> <outDir> <regex>
[<group>]");
```

```
import org.apache.hadoop.util.ToolRunner;

/* Extracts matching regexs from input files and counts them. */
public class Grep extends Configured implements Tool {
  private Grep() {}                                  // singleton

  public int run(String[] args) throws Exception {
    if (args.length < 3) {
      System.out.println("Grep <inDir> <outDir> <regex>
[<group>]");
      ToolRunner.printGenericCommandUsage(System.out);
      return -1;
    }

    Path tempDir =
      new Path("grep-temp-"+
          Integer.toString(new Random().nextInt
(Integer.MAX_VALUE)));

    JobConf grepJob = new JobConf(getConf(), Grep.class);

    try {

      grepJob.setJobName("grep-search");

      FileInputFormat.setInputPaths(grepJob, args[0]);
```

```
    grepJob.setJobName("grep-search");

    FileInputFormat.setInputPaths(grepJob, args[0]);

    grepJob.setMapperClass(RegexMapper.class);
    grepJob.set("mapred.mapper.regex", args[2]);
    if (args.length == 4)
      grepJob.set("mapred.mapper.regex.group", args[3]);

    grepJob.setCombinerClass(LongSumReducer.class);
    grepJob.setReducerClass(LongSumReducer.class);

    FileOutputFormat.setOutputPath(grepJob, tempDir);
    grepJob.setOutputFormat(SequenceFileOutputFormat.class);
    grepJob.setOutputKeyClass(Text.class);
    grepJob.setOutputValueClass(LongWritable.class);

    JobClient.runJob(grepJob);

    JobConf sortJob = new JobConf(Grep.class);
    sortJob.setJobName("grep-sort");

    FileInputFormat.setInputPaths(sortJob, tempDir);
    sortJob.setInputFormat(SequenceFileInputFormat.class);
```

```
    FileOutputFormat.setOutputPath(grepJob, tempDir);
    grepJob.setOutputFormat(SequenceFileOutputFormat.class);
    grepJob.setOutputKeyClass(Text.class);
    grepJob.setOutputValueClass(LongWritable.class);

    JobClient.runJob(grepJob);

    JobConf sortJob = new JobConf(Grep.class);
    sortJob.setJobName("grep-sort");

    FileInputFormat.setInputPaths(sortJob, tempDir);
    sortJob.setInputFormat(SequenceFileInputFormat.class);
```

```
      sortJob.setMapperClass(InverseMapper.class);

      sortJob.setNumReduceTasks(1);                  // write a
single file
      FileOutputFormat.setOutputPath(sortJob, new Path(args
[1]));
      sortJob.setOutputKeyComparatorClass            // sort by
decreasing freq
      (LongWritable.DecreasingComparator.class);

      JobClient.runJob(sortJob);
    }
    finally {
```

```
      JobClient.runJob(grepJob);

      JobConf sortJob = new JobConf(Grep.class);
      sortJob.setJobName("grep-sort");

      FileInputFormat.setInputPaths(sortJob, tempDir);
      sortJob.setInputFormat(SequenceFileInputFormat.class);

      sortJob.setMapperClass(InverseMapper.class);

      sortJob.setNumReduceTasks(1);                 // write a
single file
      FileOutputFormat.setOutputPath(sortJob, new Path(args
[1]));
      sortJob.setOutputKeyComparatorClass           // sort by
decreasing freq
      (LongWritable.DecreasingComparator.class);

      JobClient.runJob(sortJob);
    }
    finally {
      FileSystem.get(grepJob).delete(tempDir, true);
    }
    return 0;
  }
```

REGEXMAPPER.JAVA

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.apache.hadoop.mapred.lib;

import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```java
/** A {@link Mapper} that extracts text matching a regular expression.
*/
public class RegexMapper<K> extends MapReduceBase
    implements Mapper<K, Text, Text, LongWritable> {

  private Pattern pattern;
  private int group;

  public void configure(JobConf job) {
    pattern = Pattern.compile(job.get("mapred.mapper.regex"));
    group = job.getInt("mapred.mapper.regex.group", 0);
  }

  public void map(K key, Text value,
                  OutputCollector<Text, LongWritable> output,
                  Reporter reporter)
    throws IOException {
    String text = value.toString();
    Matcher matcher = pattern.matcher(text);
    while (matcher.find()) {
      output.collect(new Text(matcher.group(group)), new LongWritable
(1));
    }
  }

}
```

MAPREDUCE

- View the Shakespeare output “part-XXXX” file
- Why “part-XXXX” naming?

HAVE CODE, WILL TRAVEL

- Code travels to the data
- Opposite of traditional systems

STREAMING

UNIX VIA MAPREDUCE

- Any UNIX command
- Any shell-invokable script
  - Perl
  - Python
  - Ruby

UNIX VIA MAPREDUCE

- Line at a time
- Tab separator

```
hadoop jar contrib/streaming/
hadoop-0.20.1-streaming.jar
 -input people.csv
-output outputstuff
-mapper 'cut -f 2 -d ,'
-reducer 'uniq'
```

STREAMING

- Run a streaming job

Lab

The Grid

DSLs

Tools

The Grid

MOTIVATIONS

1 Gigabyte

1 Terabyte

1 Petabyte

16 Petabytes

Near-Linear Hardware Scalability

APPLICATIONS

- Protein folding
  pharmaceutical research
- Search Engine Indexing
  walking billions of web pages
- Product Recommendations
  based on other customer purchases
- Sorting
  terabytes to petabyes in size
- Classification
  government intelligence

CONTEXTUAL ADS

CONTEXTUAL ADS

Shopper A

looking at

Product X

is told that 66% of other customers who bought X

Customer B

also bought Y

Product Y

Customer C

did not buy

Customer D

also bought Y

```
SELECT reccProd.name, reccProd.id
  FROM products reccProd
  WHERE purchases.customerId =
```

```
SELECT reccProd.name, reccProd.id
  FROM products reccProd
  WHERE purchases.customerId =

    (SELECT customerId
      FROM customers
      WHERE purchases.productId = thisProd)

LIMIT 5
```

GRID BENEFITS

SCALABLE

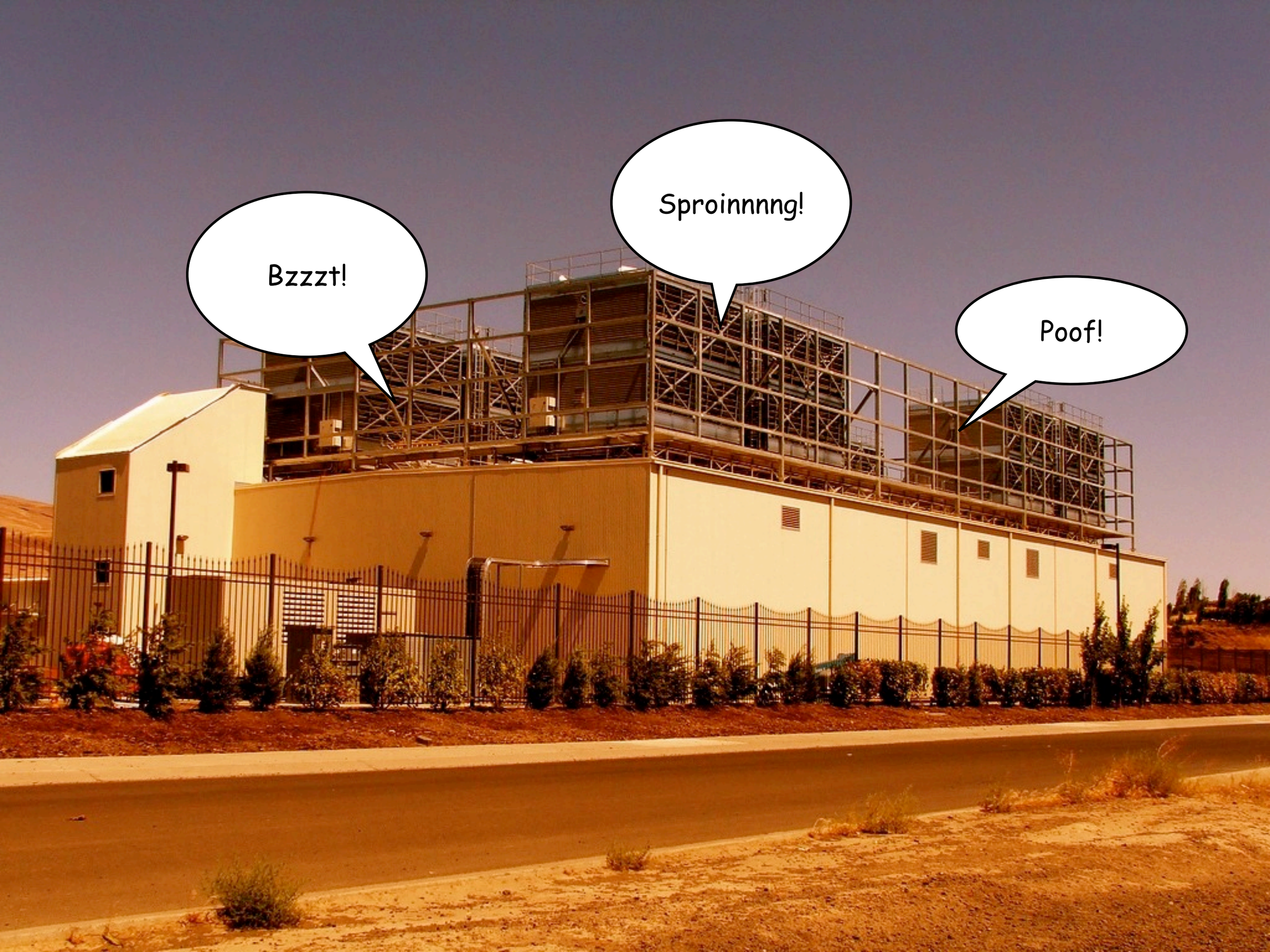
- Data storage is pipelined
- Code travels to data
- Near linear hardware scalability

OPTIMIZED

- Preemptive execution
- Maximizes use of faster hardware
- New jobs trump P.E.

FAULT TOLERANT

- Configurable data redundancy
- Minimizes hardware failure impact
- Automatic job retries
- Self healing filesystem


Bzzzt!
Sproinnnng!
Poof!

SERVER FUNERALS

- **No pagers** go off when machines die
- Report of dead machines **once a week**
  - Clean out the carcasses

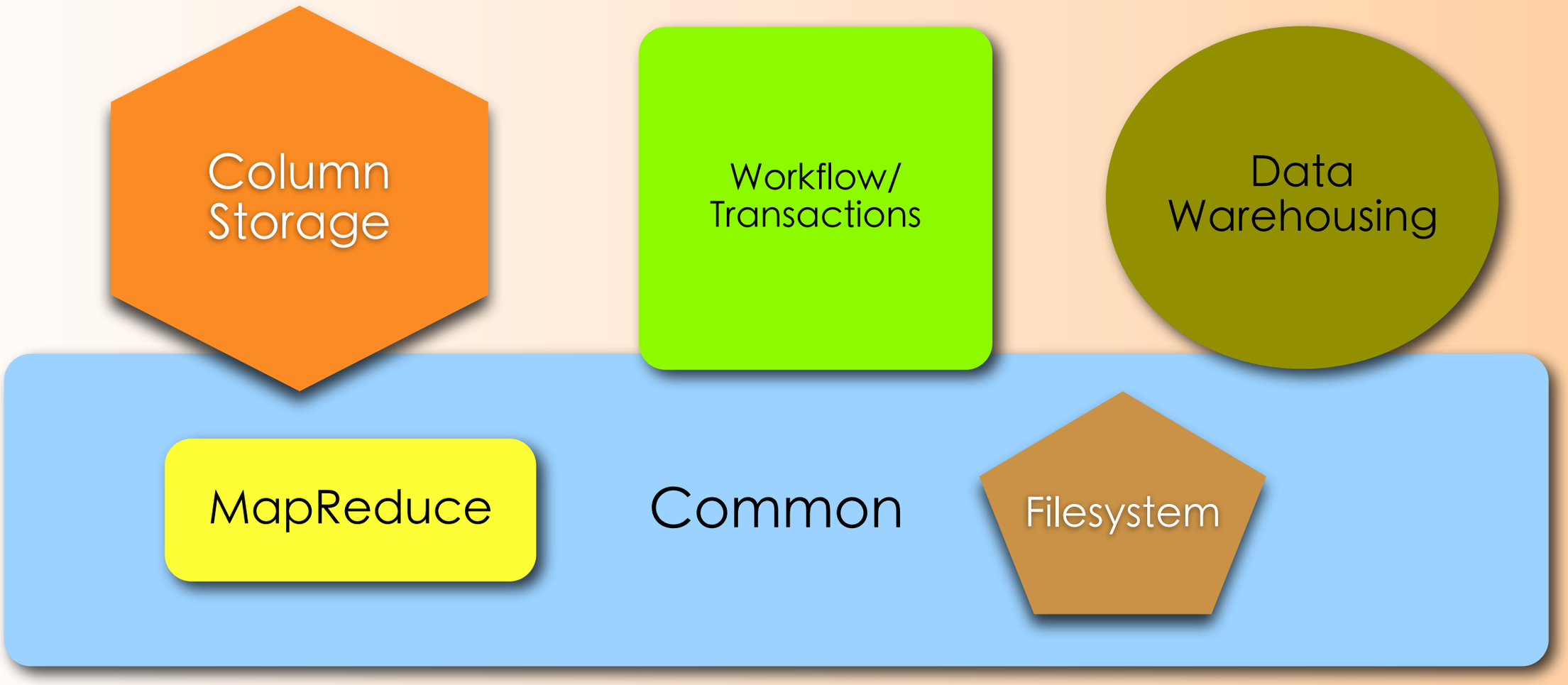
Robustness attributes
prevented from bleeding
into application code

- Data redundancy
- Node death
- Retries
- Data geography
- Parallelism
- Scalability

COMPONENTS

HADOOP COMPONENTS

HADOOP COMPONENTS



HADOOP COMPONENTS

Tool	Purpose
Common	MapReduce
HDFS	Filesystem
Pig	Analyst MapReduce language
HBase	Column-oriented data storage
Hive	SQL-like language for HBase
ZooKeeper	Workflow & distributed transactions
Chukwa	Log file processing

The Grid

DSLs

Tools

DSLs

SYNC, ASYNC

- **Pig** is **asynchronous**
- **Hive** is **asynchronous**
- **HBase** is **near realtime**
- **RDBMS** is **realtime**


PIG

PIG BASICS

- Yahoo-authored add-on
- **High-level language** for authoring data analysis programs
- Console

PIG SAMPLE

```
Person = LOAD 'people.csv' using PigStorage(',');
Names = FOREACH Person GENERATE $2 AS name;
OrderedNames = ORDER Names BY name ASC;
GroupedNames = GROUP OrderedNames BY name;
NameCount = FOREACH GroupedNames
 GENERATE group, COUNT(OrderedNames);
store NameCount into 'names.out';
```

Terminal — bash — bash — 62×20

```
[pig-example (master)⚡]>
```

Terminal — bash — bash — 62×20

```
[pig-example (master)⚡]> pig -x local get-names-from-people.pig
2010-04-20 05:58:16,851 [main] INFO  org.apache.pig.Main - Logging error messages to: /Users/mccm06/Documents/Teach/Courses/Hadoop-Intro/hadoop-examples_git/pig-example/pig_1271764696851.log
2010-04-20 05:58:52,460 [main] INFO  org.apache.pig.backend.local.executionengine.LocalPigLauncher - Successfully stored result in: "file:/Users/mccm06/Documents/Teach/Courses/Hadoop-Intro/hadoop-examples_git/pig-example/names.out"
2010-04-20 05:58:52,460 [main] INFO  org.apache.pig.backend.local.executionengine.LocalPigLauncher - Records written : 5158
2010-04-20 05:58:52,462 [main] INFO  org.apache.pig.backend.local.executionengine.LocalPigLauncher - Bytes written : 0
2010-04-20 05:58:52,462 [main] INFO  org.apache.pig.backend.local.executionengine.LocalPigLauncher - 100% complete!
2010-04-20 05:58:52,462 [main] INFO  org.apache.pig.backend.local.executionengine.LocalPigLauncher - Success!!
[pig-example (master)⚡]>
```


HIVE

HIVE BASICS

- Authored by facebook
- SQL interface to HBase
- Hive is low-level
- Hive-specific metadata
- Data warehousing

```
SELECT * FROM shakespeare
WHERE freq > 100
SORT BY freq ASC
LIMIT 10;
```

SYNC, ASYNC

- **RDBMS** SQL is **realtime**
- **Hive** is primarily **asynchronous**


HBASE

Hbase/PoweredBy

Adobe - We currently have about 30 nodes running HDFS, Hadoop and HBase in clusters ranging from 5 to 14 nodes on both production and development. We plan a deployment on an 80 nodes cluster. We are using HBase in several areas from social services to structured data and processing for internal use. We constantly write data to HBase and run mapreduce jobs to process then store it back to HBase or external systems. Our production cluster has been running since Oct 2008.

Drawn to Scale Consulting consults on HBase, Hadoop, Distributed Search, and Scalable architectures.

Filmweb is a film web portal with a large dataset of films, persons and movie-related entities. We have just started a small cluster of 3 HBase nodes to handle our web cache persistency layer. We plan to increase the cluster size, and also to start migrating some of the data from our databases which have some demanding scalability requirements.

Flurry provides mobile application analytics. We use HBase and Hadoop for all of our analytics processing, and serve all of our live requests directly out of HBase on our 16-node production cluster with billions of rows over several tables.

GumGum is an in-image ad network. We use HBase 0.20 on a 4-node Amazon EC2 Large Instance (m1.large) cluster for both real-time data and analytics. Our production cluster has been running since June 2010.

Kalooga is a discovery service for image galleries. We use Hadoop, Hbase, Chukwa and Pig on a 20-node cluster for our crawling, analysis and events processing.

Lily is an open source content repository backed by HBase and SOLR from Outerthought - scalable content applications.

Mahalo, "...the world's first human-powered search engine". All the markup that powers the wiki is stored in HBase. It's been in use for a few months now. MediaWiki - the same software that power Wikipedia - has version/revision control. Mahalo's in-house editors produce a lot of revisions per day, which was not working well in a RDBMS. An hbase-based solution for this was built and tested, and the data migrated out of MySQL and into HBase. Right now it's at something like 6 million items in HBase. The upload tool runs every hour from a shell script to back up that data, and on 6 nodes takes about 5-10 minutes to run - and does not slow down production at all.

Meetup is on a mission to help the world's people self-organize into local groups. We use Hadoop and HBase to power a site-wide, real-time activity feed system for all of our members and groups. Group activity is written directly to HBase, and indexed per member, with the member's custom feed served directly from HBase for incoming requests. We're running HBase 0.20.0 on a 11 node cluster.

Ning uses HBase to store and serve the results of processing user events and log files, which allows us to provide near-real time analytics and reporting. We use a small cluster of commodity machines with 4 cores and 16GB of RAM per machine to handle all our analytics and reporting needs.

Stumbleupon and Su.pr use HBase as a real time data storage and analytics platform. Serving directly out of HBase, various site features and statistics are kept up to date in a real time fashion. We also use HBase a map-reduce data source to overcome traditional query speed limits in MySQL.

SubRecord Project is an Open Source project that is using HBase as a repository of records (persisted map-like data) for the aspects it provides like logging, tracing or metrics. HBase and Lucene index both constitute a repo/storage for this platform.

Shopping Engine at Tokenizer is a web crawler; it uses HBase to store URLs and Outlinks (AnchorText + LinkedURL): more than a billion. It was initially designed as Nutch-Hadoop extension, then (due to very specific 'shopping' scenario) moved to SOLR + MySQL(InnoDB) (ten thousands queries per second), and now - to HBase. HBase is significantly faster due to: no need for huge transaction logs, column-oriented design exactly matches 'lazy' business logic, data compression, MapReduce support. Number of mutable 'indexes' (term from RDBMS) significantly reduced due to the fact that each 'row::column' structure is physically sorted by 'row'. MySQL InnoDB engine is best DB choice for highly-concurrent updates. However, necessity to flash a block of data to harddrive even if we changed only few bytes is obvious bottleneck. HBase greatly helps: not-so-popular in modern DBMS 'delete-insert', 'mutable primary key', and 'natural primary key' patterns become a big advantage with HBase.

Trend Micro uses HBase as a foundation for cloud scale storage for a variety of applications. We have been developing with HBase since version 0.1 and production since version 0.20.0.

Twitter runs HBase across its entire Hadoop cluster. HBase provides a distributed, read/write backup of all mysql tables in Twitter's production backend, allowing engineers to run MapReduce jobs over the data while maintaining the ability to apply periodic row updates (something that is more difficult to do with vanilla HDFS). A number of applications including people search rely on HBase internally for data generation. Additionally, the operations team uses HBase as a timeseries database for cluster-wide monitoring/performance data.

Veoh Networks uses HBase to store and process visitor(human) and entity(non-human) profiles which are used for behavioral targeting, demographic detection, and personalization services. Our site reads this data in real-time (heavily cached) and submits updates via various batch map/reduce jobs. With 25 million unique visitors a month storing this data in a traditional RDBMS is not an option. We currently have a 24 node Hadoop/HBase cluster and our profiling system is sharing this cluster with our other Hadoop data pipeline processes.

VideoSurf - "The video search engine that has taught computers to see". We're using Hbase to persist various large graphs of data and other statistics. Hbase was a real win for us because it let us store substantially larger datasets without the need for manually partitioning the data and it's column-oriented nature allowed us to create schemas that were substantially more efficient for storing and retrieving data.

Visible Technologies - We use Hadoop, HBase, Katta, and more to collect, parse, store, and search hundreds of millions of Social Media content. We get incredibly fast throughput and very low latency on commodity hardware. HBase enables our business to exist.

WorldLingo - The WorldLingo Multilingual Archive. We use HBase to store millions of documents that we scan using Map/Reduce jobs to machine translate them into all or selected target languages from our set of available machine translation languages. We currently store 12 million documents but plan to eventually reach the 450 million mark. HBase allows us to scale out as we need to grow our storage capacities. Combined with Hadoop to keep the data replicated and therefore fail-safe we have the backbone our service can rely on now and in the future. WorldLingo is using HBase since December 2007 and is along with a few others one of the longest running HBase installation. Currently we are running the latest HBase 0.20 and serving directly from it: MultilingualArchive.

Stumbleupon and Su.pr use HBase as a real time data storage and analytics platform. Serving directly out of HBase, various site features and statistics are kept up to date in a real time fashion. We also use HBase a map-reduce data source to overcome traditional query speed limits in MySQL.

SubRecord Project is an Open Source project that is using HBase as a repository of records (persisted map-like data) for the aspects it provides like logging, tracing or metrics. HBase and Lucene index both constitute a repo/storage for this platform.

Shopping Engine at Tokenizer is a web crawler; it uses HBase to store URLs and Outlinks (AnchorText + LinkedURL): more than a billion. It was initially designed as Nutch-Hadoop extension, then (due to very specific 'shopping' scenario) moved to SOLR + MySQL(InnoDB) (ten thousands queries per second), and now - to HBase. HBase is significantly faster due to: no need for huge transaction logs, column-oriented design exactly matches 'lazy' business logic, data compression, MapReduce support. Number of mutable 'indexes' (term from RDBMS) significantly reduced due to the fact that each 'row::column' structure is physically sorted by 'row'. MySQL InnoDB engine is best DB choice for highly-concurrent updates. However, necessity to flash a block of data to harddrive even if we changed only few bytes is obvious bottleneck. HBase greatly helps: not-so-popular in modern DBMS 'delete-insert', 'mutable primary key', and 'natural primary key' patterns become a big advantage with HBase.

Trend Micro uses HBase as a foundation for cloud scale storage for a variety of applications. We have been developing with HBase since version 0.1 and production since version 0.20.0.

Twitter runs HBase across its entire Hadoop cluster. HBase provides a distributed, read/write backup of all mysql tables in Twitter's production backend, allowing engineers to run MapReduce jobs over the data while maintaining the ability to apply periodic row updates (something that is more difficult to do with vanilla HDFS). A number of applications including people search rely on HBase internally for data generation. Additionally, the operations team uses HBase as a timeseries database for cluster-wide monitoring/performance data.

Veoh Networks uses HBase to store and process visitor(human) and entity(non-human) profiles which are used for behavioral targeting, demographic detection, and personalization services. Our site reads this data in real-time (heavily cached) and submits updates via various batch map/reduce jobs. With 25 million unique visitors a month storing this data in a traditional RDBMS is not an option. We currently have a 24 node Hadoop/HBase cluster and our profiling system is sharing this cluster with our other Hadoop data pipeline processes.

VideoSurf - "The video search engine that has taught computers to see". We're using Hbase to persist various large graphs of data and other statistics. Hbase was a real win for us because it let us store substantially larger datasets without the need for manually partitioning the data and it's column-oriented nature allowed us to create schemas that were substantially more efficient for storing and retrieving data.

Visible Technologies - We use Hadoop, HBase, Katta, and more to collect, parse, store, and search hundreds of millions of Social Media content. We get incredibly fast throughput and very low latency on commodity hardware. HBase enables our business to exist.

WorldLingo - The WorldLingo Multilingual Archive. We use HBase to store millions of documents that we scan using Map/Reduce jobs to machine translate them into all or selected target languages from our set of available machine translation languages. We currently store 12 million documents but plan to eventually reach the 450 million mark. HBase allows us to scale out as we need to grow our storage capacities. Combined with Hadoop to keep the data replicated and therefore fail-safe we have the backbone our service can rely on now and in the future. WorldLingo is using HBase since December 2007 and is along with a few others one of the longest running HBase installation. Currently we are running the latest HBase 0.20 and serving directly from it: MultilingualArchive.

HBASE BASICS

- Map-oriented storage
- Key value pairs
- Column families
- Stores to HDFS
- Fast
- Usable for synchronous responses

```
hbase>
help

create 'mylittletable', 'mylittlecolumnfamily'
describe 'mylittletable'
```

```
put 'mylittletable', 'r2', 'mylittlecolumnfamily', 'x'

get 'mylittletable', 'r2'
scan 'mylittletable'
```

SQOOP

Overview

Downloads

- Cloudera's Hadoop Distribution
- Get Virtual Machine
- CDH for Amazon
- Cloudera Desktop
- Sqoop

Learn Hadoop

Get Support

Blog

Sqoop

SQL-to-Hadoop database import tool

Sqoop is a tool designed to import data from relational databases into Hadoop. Sqoop uses JDBC to connect to a database. It examines each table's schema and automatically generates the necessary classes to import data into the Hadoop Distributed File System (HDFS). Sqoop then creates and launches a MapReduce job to read tables from the database via DBInputFormat, the JDBC-based InputFormat. Tables are read into a set of files in HDFS. Sqoop supports both SequenceFile and text-based target and includes performance enhancements for loading data from MySQL.

Getting Sqoop

Sqoop is an open-source program contributed to the Apache Hadoop project. The most recent release of Cloudera's Distribution for Hadoop contains Sqoop.

If you're already using our RPM packages, running the command `yum update hadoop`

Instructions

The instructions below describe how to get started using Sqoop to import your data into Hadoop:

- Example Usage
- Connecting to a Database Server
- Listing Available Tables

SQOOP

A cloudera utility

- Imports from RDBMS
- Outputs plaintext, SequenceFile, or Hive

```
sqoop --connect jdbc:mysql://database.example.com/
```

The Grid DSLs Tools

Tools

MONITORING

WEB STATUS PANELS

NameNode

http://localhost:50070/

JobTracker

http://localhost:50030/

Quick Links

localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Fri Apr 30 09:49:45 EDT 2010
Version: 0.20.1, r810220
Compiled: Tue Sep 1 20:55:56 UTC 2009 by oom
Identifier: 201004300949

Cluster Summary (Heap Size is 80.56 MB/994.81 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	3	1	2	2	4.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

EXTENDED FAMILY MEMBERS

Cascading

ANOTHER DSL

- Java-based
- Different vocabulary
- Abstraction from MapReduce
- Uses Hadoop
- Cascalog for Clojure

Cascading

Search

Search

Site

Home
About
Features
News
Powered By
Contact

Recent News

September 22, 2010
Memcached, Membase, and ElasticSearch Integration
September 16, 2010
O'Reilly Strata Conference - Last Call
August 26, 2010
Bixo Hackathon
August 25, 2010
O'Reilly Strata Conference
August 2, 2010
Cascading 1.1.2

Recent Articles

BigDataCamp 2010

Documentation

Diving In

Welcome

Cascading is a Query API and Query Planner used for defining and executing complex, scale-free, and fault tolerant da processing workflows on a Hadoop cluster.

Cascading is a thin Java library that sits on top of Hadoop's MapReduce layer. It is not a new text based query syntax (like Pig) or another complex system that must be installed on a cluster and maintained (like Hive). Though Cascading both complimentary to and is a valid alternative to either application.

Cascading is simply a query processing API that lets the developer quickly assemble complex distributed processes without having to "think" in MapReduce. And to efficiently schedule them based on their dependencies. Obviously sim data processing applications are supported as well, as complex applications tend to start simple.

Cascading is Open Source and dual licensed under the GPL and OEM/Commercial Licenses. OEM/Commercial Licenses and Developer Support can be obtained through Concurrent, Inc.

Cascading has a strong community of users and contributors, see our Cascading modules page for related projects a extensions.

Cascading, extensions, and related libraries are also hosted in the Conjars maven repository maintained by Concurren Inc. The repository is open to the public.

Read more about Cascadings features or thumb through the Cascading User Guide.

Recent Events

Memcached, Membase, and ElasticSearch Integration

September 22, 2010 12:40 PM | Permalink

We have added a link to the Cascading.Memecached project on GitHub to the Modules and Extensions page.

This sub-project provides Memcached API integration allowing Cascading Flows to push data into various memcached API compliant applications like ElasticSearch and Membase.

« New Cascalog features: outer joins, combiners, sorting, and more | Main | Fun with equality in Clojure »

Introducing Cascalog: a Clojure-based query language for Hadoop

WEDNESDAY, APRIL 14, 2010 CASCADING, CASCALOG, CLOJURE, HADOOP

I'm very excited to be releasing Cascalog as open-source today. Cascalog is a Clojure-based query language for Hadoop inspired by Datalog.

Highlights

- **Simple** - Functions, filters, and aggregators all use the same syntax. Joins are implicit and natural.
- **Expressive** - Logical composition is very powerful, and you can run arbitrary Clojure code in your query with little effort.
- **Interactive** - Run queries from the Clojure REPL.
- **Scalable** - Cascalog queries run as a series of MapReduce jobs.
- **Query anything** - Query HDFS data, database data, and/or local data by making use of Cascading's "Tap" abstraction
- **Careful handling of null values** - Null values can make life difficult. Cascalog has a feature called "non-nullable variables" that makes dealing with nulls

Follow Nathan on

Twitter
GitHub
LinkedIn
Blog RSS

Search

Featured Posts

You should blog even if you have no readers

Karmasphere

HADOOP STUDIO

- Free and commercial offerings
- Workflow designer
- IDE based testing

JavaApplication1 – NetBeans IDE 6.8

<default conf...

Search (⌘+I)

Projects | Files | Services

- JavaApplication1
  - Source Packages
    - <default package>
      - NewKarmasphereHadoopJob.jav
    - javaapplication1
      - Main.java
  - Test Packages
  - Libraries
  - Test Libraries

Navigator

Members View

- NewKarmasphereHadoopJob :: JobSubmiss
  - NewKarmasphereHadoopJob()
  - initJobSubmission(JobSubmission submi
  - main(String[] args)

NewKarmasphereHadoopJob.java

Source | Workflow | Force Refresh | Properties | Deploy

HadoopBootstrap (Object)

Input Resources

Input File: null Browse...

Integration Data

Advanced Configuration

Key	Value	Type
karmasphere.inte...	true	Boolean
karmasphere.inte...	true	Boolean
mapred.reduce.t...	1	Integer

HadoopInput (TextInputFormat)

Basic Information

Class name: org.apache.hadoop.mapred.TextI

Input type: Path

Output type: HadoopInputRecord

Integration Data

#	Key	Valu

Tasks

Description	File	Location
TODO code application logic here	Main.java	...Main.java:

TODO: 1 in all opened projects

Cloudera Desktop

Developer Center

Overview

Downloads

- Cloudera's Hadoop Distribution
- Get Virtual Machine
- CDH for Amazon
- Cloudera Desktop
- Sqoop

Learn Hadoop

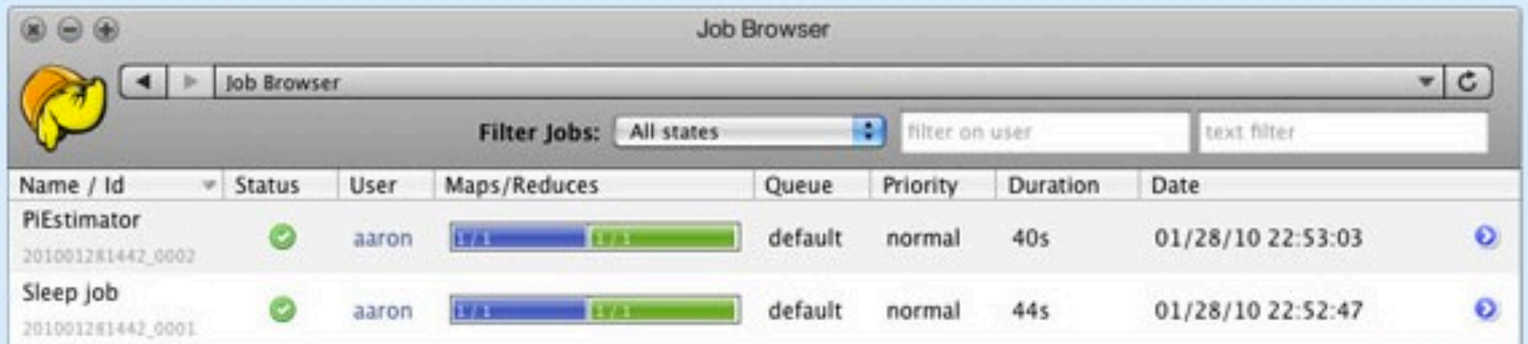
Get Support

Blog

Cloudera Desktop

A unified user interface for users and operators of Hadoop clusters.

Cloudera Desktop is a graphical user interface for the many tools required to operate and develop for Hadoop. These tools are collected into a desktop environment and delivered as a Web application that simplifies cluster administration and job development. With Cloudera Desktop, the world-class performance and scalability of Apache Hadoop is now accessible to anyone in your organization.



GRID ON THE CLOUD


amazon
web services™

ELASTIC MAP REDUCE

- Cluster MapReduce by the hour
- Easiest grid to set up
- Storage on S3

EMR PRICING

US – N. Virginia | US – N. California | EU – Ireland

Standard On-Demand Instances	Amazon EC2 Price per hour (On-Demand Instances)	Amazon Elastic MapReduce Price per hour
Small (Default)	$0.085 per hour	$0.015 per hour
Large	$0.34 per hour	$0.06 per hour
Extra Large	$0.68 per hour	$0.12 per hour
High Memory On-Demand Instances	**Amazon EC2 Price per hour (On-Demand Instances)**	**Amazon Elastic MapReduce Price per hour**
m2.2xlarge	$1.20 per hour	$0.21 per hour
m2.4xlarge	$2.40 per hour	$0.42 per hour
High CPU On-Demand Instances	**Amazon EC2 Price per hour (On-Demand Instances)**	**Amazon Elastic MapReduce Price per hour**
Medium	$0.17 per hour	$0.03 per hour
Extra Large	$0.68 per hour	$0.12 per hour

amazon web services™

SUMMARY

SAPIR-WHORF HYPOTHESIS... REMIXED

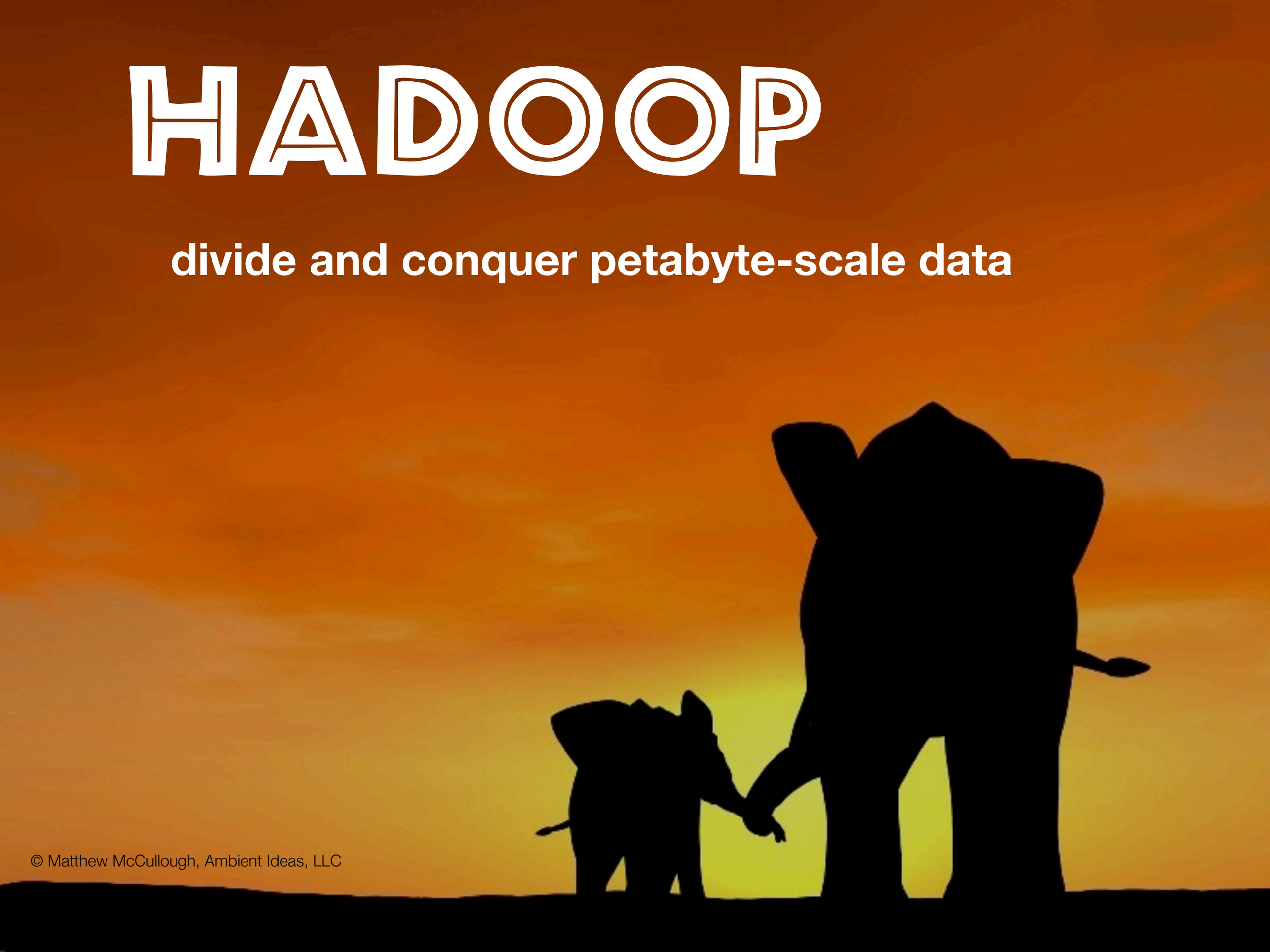
SAPIR-WHORF HYPOTHESIS... REMIXED

The **ability** to store and process **massive data** influences what you decide to store.

In the next several years,
the **scale of data problems**
we are aiming to solve
will **grow** near-**exponentially**.

HADOOP

divide and conquer petabyte-scale data



METADATA

Matthew McCullough
Ambient Ideas, LLC
matthewm@ambientideas.com
http://ambientideas.com/blog
@matthewmccull

Code

http://github.com/matthewmccullough/hadoop-intro

REFERENCES

REFERENCES

- http://labs.google.com/papers/mapreduce.html
- http://labs.google.com/papers/gfs.html
- http://labs.google.com/papers/bigtable.html
- http://wiki.apache.org/hadoop/Hbase/PoweredBy
- http://mahout.apache.org/
- http://mahout.apache.org/taste.html
- http://www.cascading.org/

REFERENCES

- http://hadoop.apache.org
- http://delicious.com/matthew.mccullough/hadoop
- http://code.google.com/edu/parallel/
- http://www.infoworld.com/d/open-source/whats-new-york-times-doing-hadoop-392?r=788
- http://atbrox.com/2009/10/01/mapreduce-and-hadoop-academic-papers/
- http://www.cloudera.com/

IMAGE CREDITS

- http://www.fontspace.com/david-rakowski/tribeca
- http://www.cern.ch/
- http://www.robinmajumdar.com/2006/08/05/google-dalles-data-centre-has-serious-cooling-needs/
- http://www.greenm3.com/2009/10/googles-secret-to-efficient-data-center-design-ability-to-predict-performance.html
- http://upload.wikimedia.org/wikipedia/commons/f/fc/CERN_LHC_Tunnel1.jpg
- http://www.flickr.com/photos/mandj98/3804322095/
- http://www.flickr.com/photos/8583446@N05/3304141843/
- http://www.flickr.com/photos/joits/219824254/
- http://www.flickr.com/photos/streetfly_jz/2312194534/
- http://www.flickr.com/photos/sybrenstuvel/2811467787/
- http://www.flickr.com/photos/lacklusters/2080288154/
- http://www.flickr.com/photos/sybrenstuvel/2811467787/
- http://www.flickr.com/photos/robryb/14826417/sizes/l/
- http://www.flickr.com/photos/mckaysavage/1037160492/sizes/l/
- http://www.flickr.com/photos/robryb/14826486/sizes/l/
- All others, iStockPhoto.com