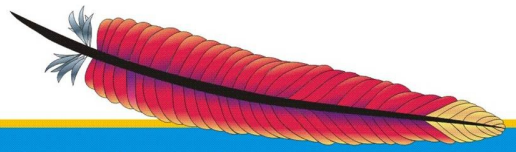
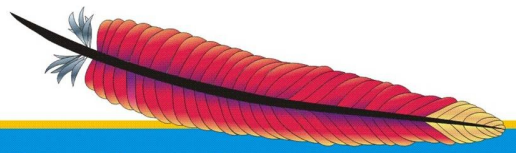
A Tour of Apache Hadoop

Tom White
Lexeme Ltd
www.lexemetech.com
tomwhite@apache.org



Itinerary

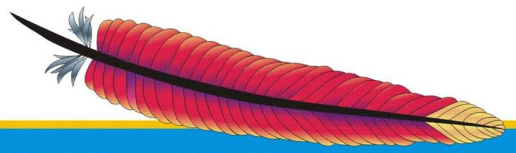
- What is Hadoop?
- Components
  - Distributed File System
  - MapReduce
  - HBase
- Related Projects



What is Hadoop?

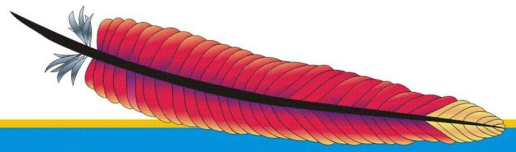
The Problem

- Existing tools are struggling to process today's large datasets
- How long to grep 1TB of log files?
- Why is this a problem for me?



How Does Hadoop Help?

- Hadoop provides a framework for storing and processing petabytes of data.
- Storage: HDFS, HBase
- Processing: MapReduce

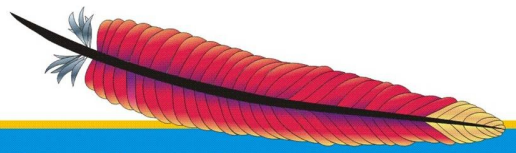


A Brief History of Hadoop

- Feb 2003 – First MapReduce library written at Google
- Oct 2003 – Google File System paper published
- Dec 2004 – Google MapReduce paper published
- Jul 2005 – Doug Cutting reports that Nutch now uses new MapReduce implementation
- Nov 2006 – Google Bigtable paper published
- Feb 2006 – Hadoop code moves out of Nutch into new Lucene sub-project
- Feb 2007 – First HBase code drop from Mike Cafarella
- Apr 2007 – Yahoo! running Hadoop on 1000-node cluster
- Jan 2008 – Hadoop made an Apache Top Level Project

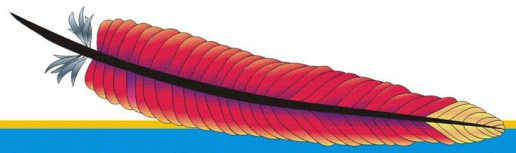
Hadoop Organization

- Apache Top Level Project
- Two sub-projects
  - Core (15 committers)
    - HDFS
    - MapReduce
  - HBase (3 committers)

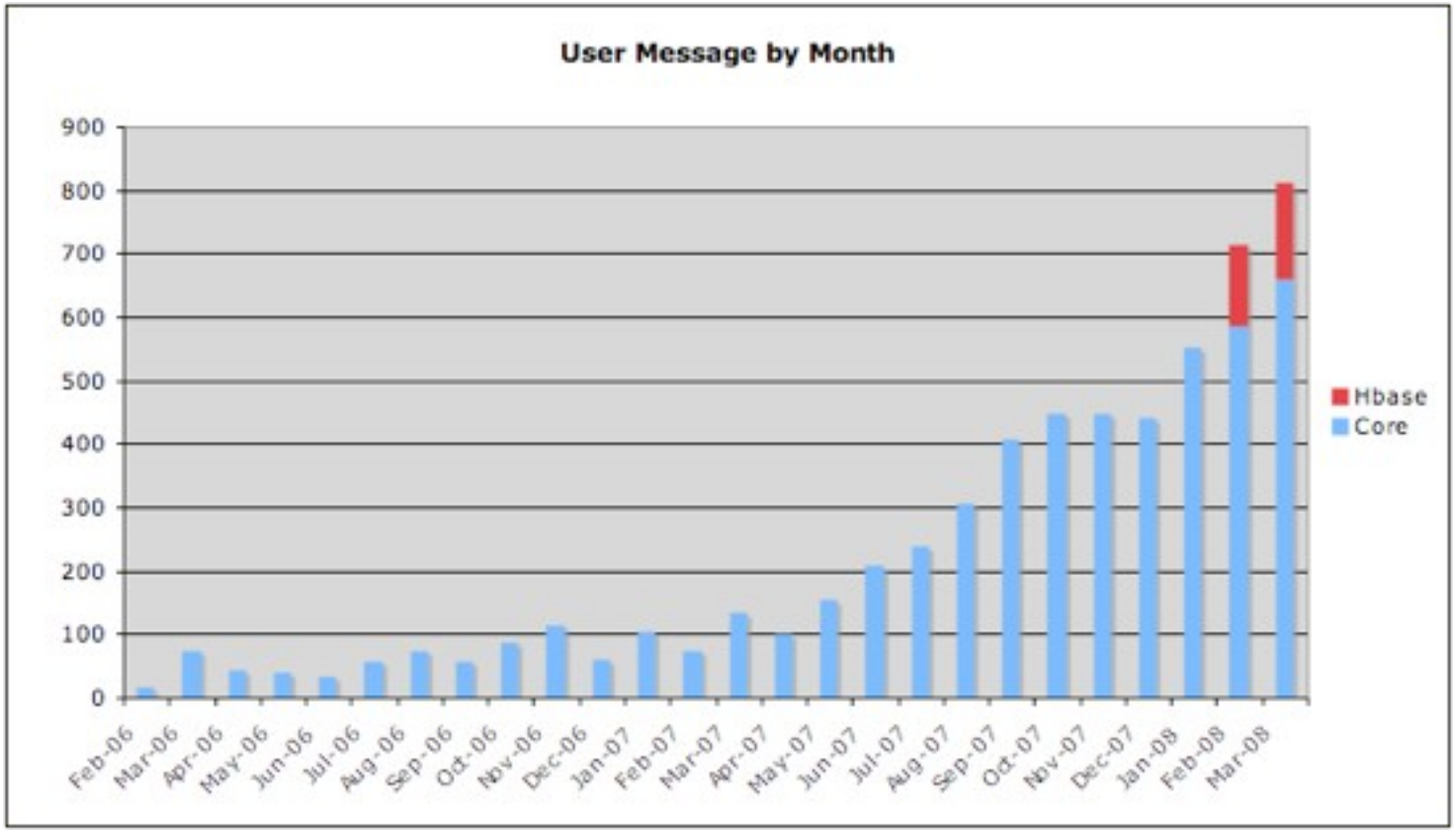


Hadoop Users

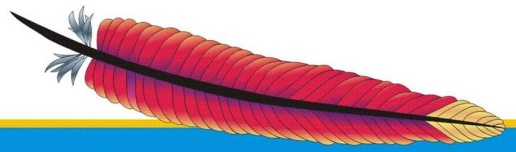
- Clusters from 1 to 2000 nodes
  - A9, Facebook, Joost, Last.fm, Yahoo! and many more
- Broad academic interest
  - IBM/Google cloud computing initiative
  - CMU/Yahoo! supercomputing cluster
- Hadoop Summit hosted by Yahoo! last month attracted over 300 attendees



Hadoop Growth

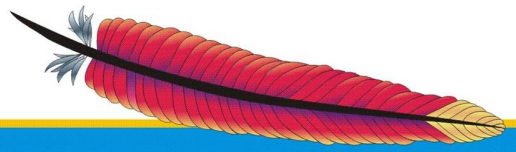


Hadoop Distributed File System



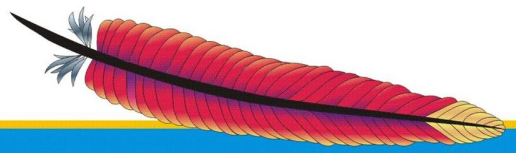
Hadoop Distributed File System - Goals

- Store large data sets
- Cope with hardware failure
- Emphasise streaming data access
- Non-goal: POSIX compliance

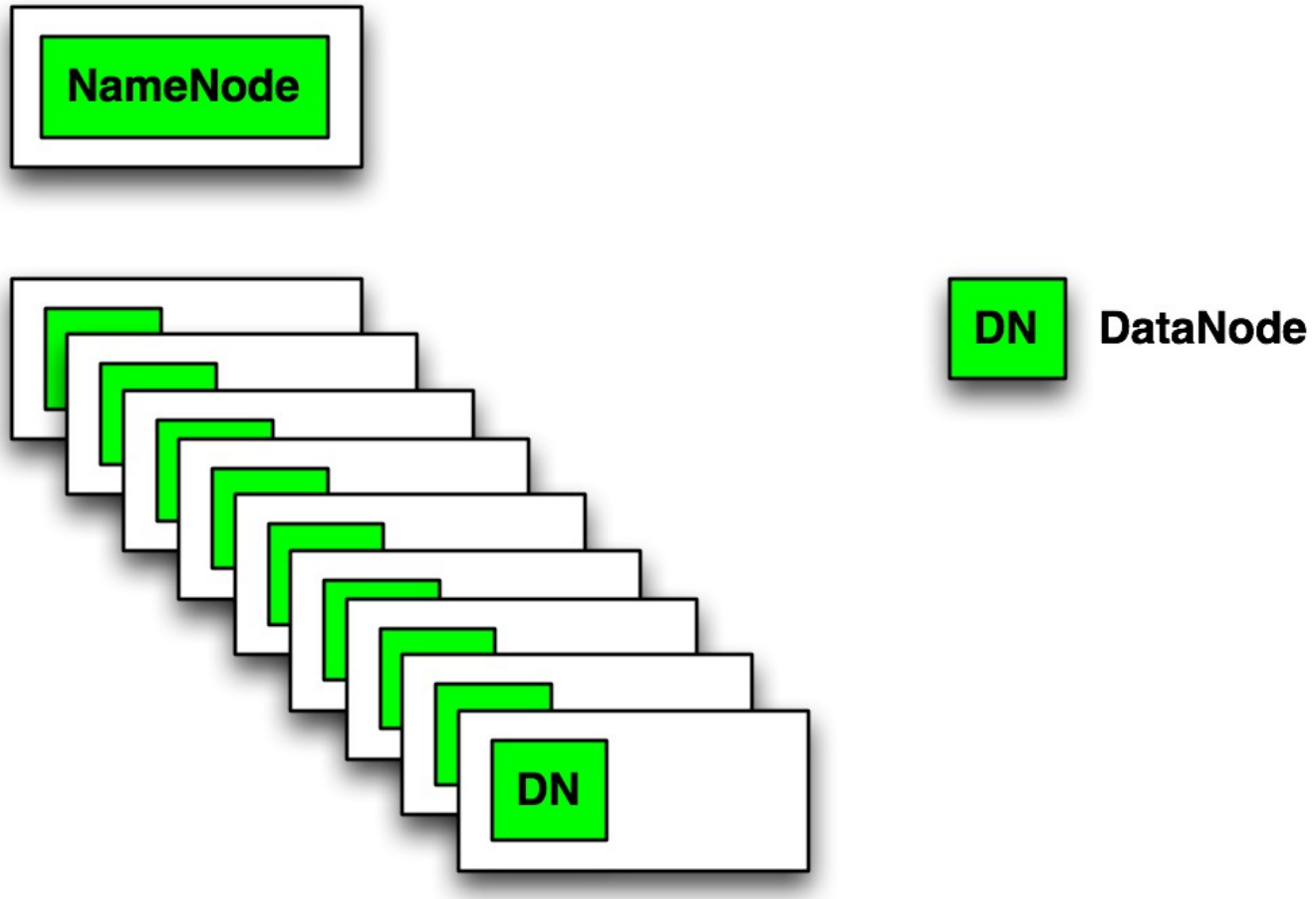


HDFS - Design

- Files are broken into blocks of 64 MB
- Datanodes handle block storage
- Single Namenode manages metadata and block placement
- Permissions

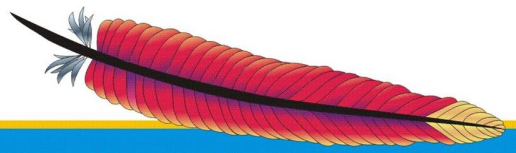


HDFS - Topology



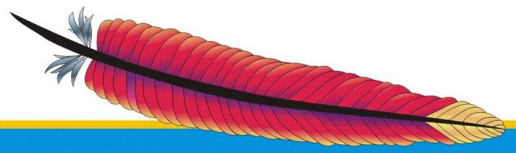
HDFS - Replication

- Default is 3x replication
- Block placement algorithm is rack-aware
- Dynamic control of replication factor
- Balancer application to re-balance cluster in background



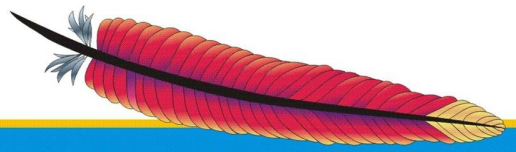
HDFS – Java API

```
URI uri = new URI("hdfs://namenode/");
FileSystem fs = FileSystem.get(uri, new Configuration());
Path file = new Path("answer");

DataOutputStream out = fs.create(file);
out.writeInt(42);
out.close();

DataInputStream in = fs.open(file);
System.out.println(in.readInt());
in.close();

fs.delete(file);
```



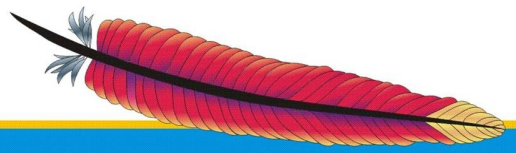
HDFS – Shell API

```
bin/hadoop fs -put food food
bin/hadoop fs -setrep 10 food
bin/hadoop fs -ls
bin/hadoop fs -cat food
bin/hadoop fs -rm food
bin/hadoop fs -lsr
bin/hadoop fs -cat .Trash/Current/food
```



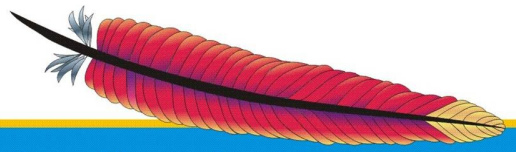
HDFS – Other Interfaces

- C - libhdfs
- HTTP FileSystem
- Web interface
- Eclipse plugin



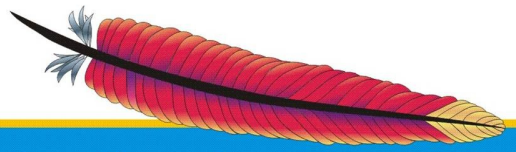
HDFS – Future

- File Appends (HADOOP-1700)
- WebDAV
- FUSE integration
- Performance



Hadoop File Systems

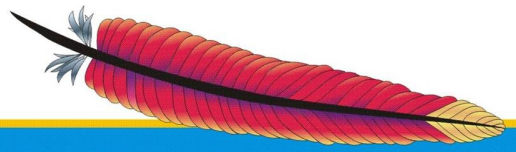
- FileSystem is a Java Interface
- Local disk (file://)
- In memory (ramfs://)
- Kosmos File System (kfs://)
- Amazon S3 (s3://)



MapReduce

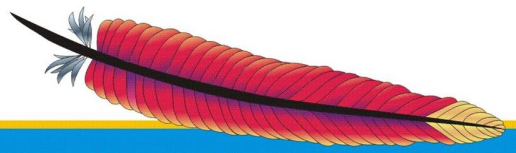
Hadoop MapReduce - Goals

- Process large data sets
- Cope with hardware failure
- High throughput
- Non-goal: low latency

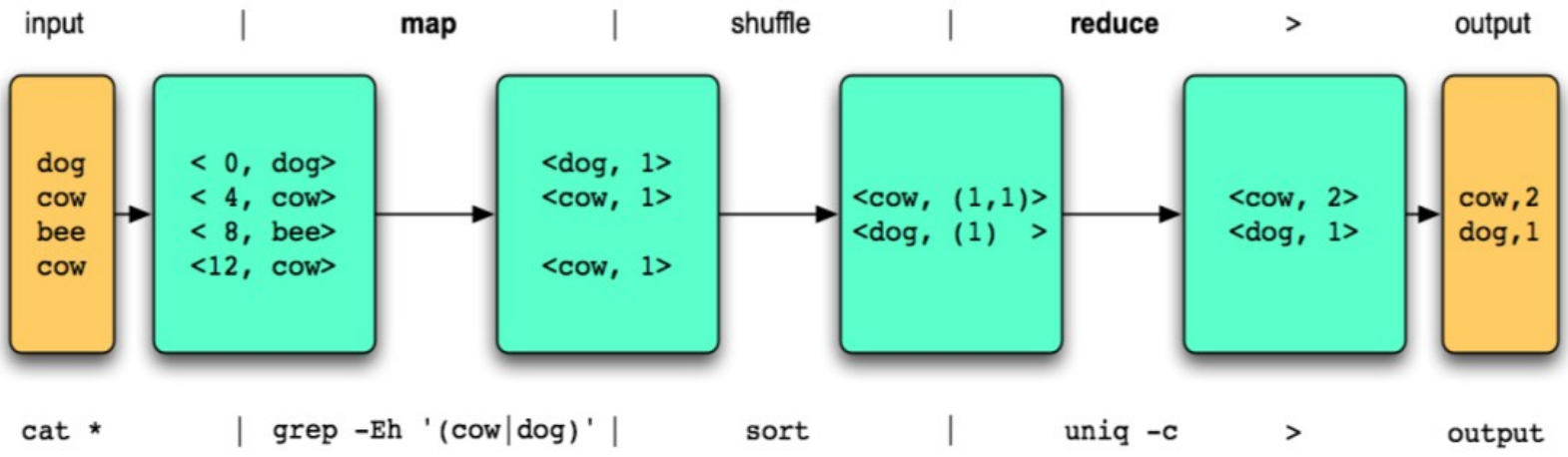


MapReduce - Example

- Simple programming model: key-value pairs
- General form:
  - Map: `(K1, V1) → list(K2, V2)`
  - Reduce: `(K2, list(V2)) → list(K3, V3)`
- grep
  - Map: `(offset, line) → [(match, 1)]`
  - Reduce: `(match, [1, 1, ...]) → [(match, n)]`



MapReduce – Logical Flow

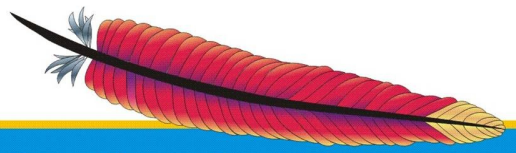


MapReduce – Map Class

```
private Pattern pattern;
public void configure(JobConf conf) {
  pattern = Pattern.compile(conf.get("regex"));
}

public void map(LongWritable key, Text val,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  if (pattern.matcher(val.toString()).matches()) {
    output.collect(val, new IntWritable(1));
  }
}
```

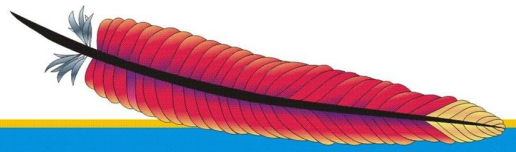
MapReduce – Reduce Class

```
public void reduce(Text key, Iterator<IntWritable> vals,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  int sum = 0;
  while (vals.hasNext()) {
    sum += vals.next().get();
  }
  output.collect(key, new IntWritable(sum));
}
```

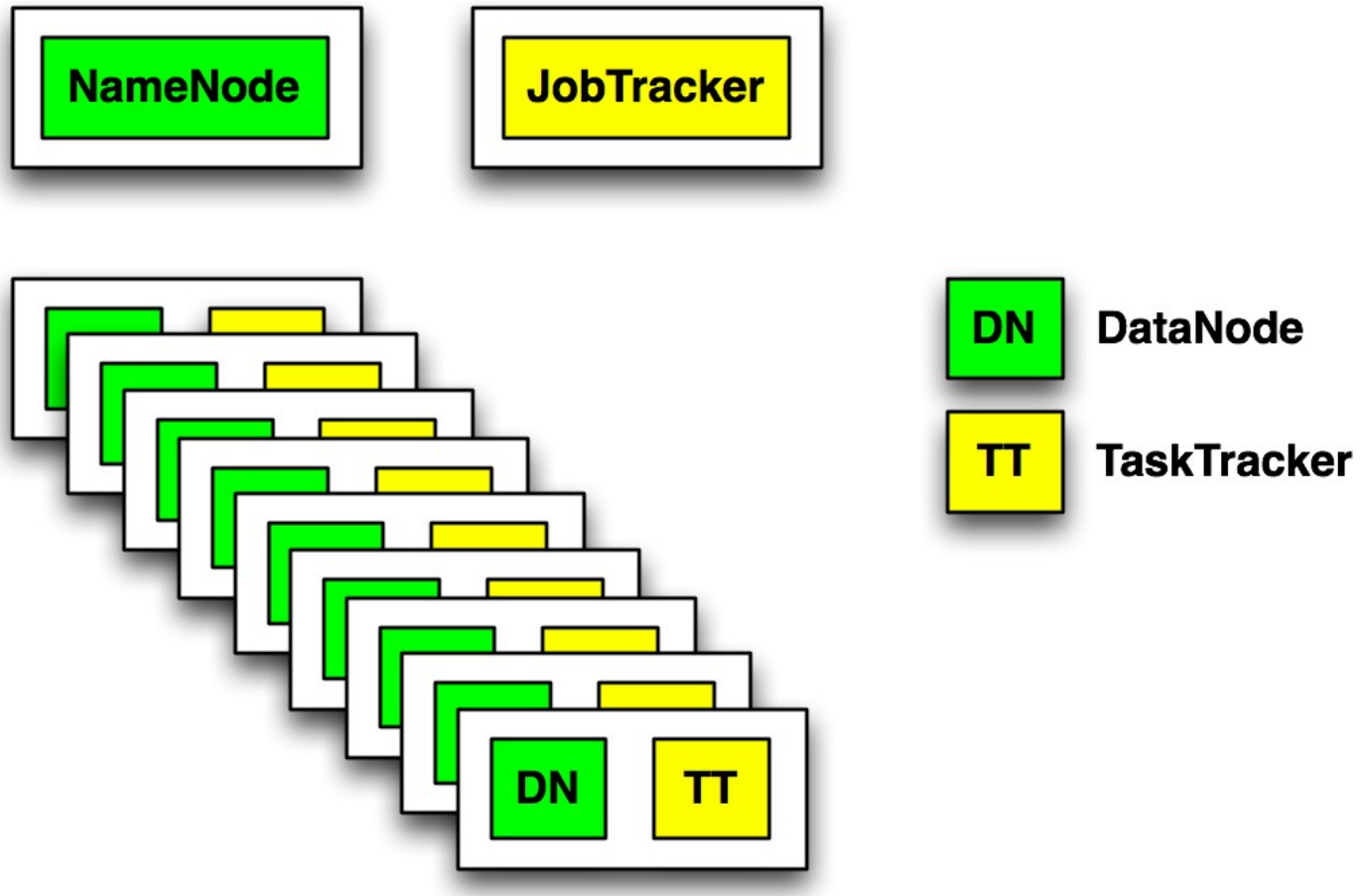


MapReduce – Job Code

```
JobConf conf = new JobConf(GrepDemo.class);
conf.setMapperClass(Map.class);
conf.set("regex", "(cow|dog)");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setReducerClass(Reduce.class);
conf.setInputPath(new Path("mr/input/grep"));
conf.setOutputPath(new Path("mr/output/grep"));
JobClient.runJob(conf);
```

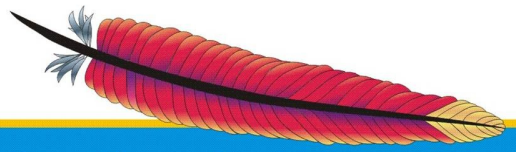


MapReduce - Topology



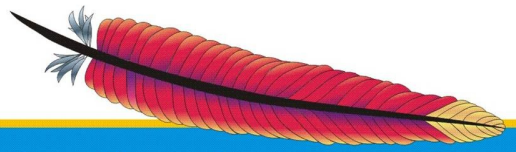
MapReduce - Locality

- “Moving computation is cheaper than moving data”
- Map tasks are scheduled on the same node that the input data resides on.

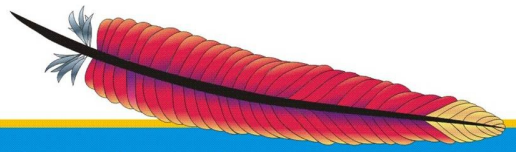


MapReduce - APIs

- Java
- Streaming
  - via stdin/stdout
- Pipes C++
  - via sockets

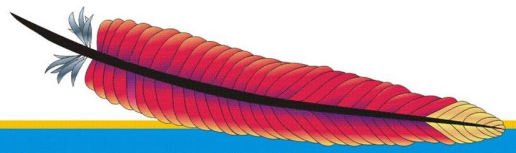


HBase



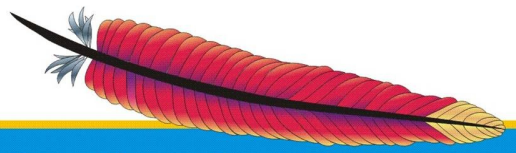
HBase - What

- Modelled on Google's Bigtable
- Row/column store
- Billions of rows x millions of columns
- Column-oriented – nulls are free
- Untyped – stores byte[]



HBase – Data Model

Row	Timestamp	Column family animal:		Column family repairs:
		animal:type	animal:size	repairs:cost
enclosure1	t2	zebra		1000 EUR
	t1	lion	big	
enclosure2	...	...	...	...



HBase – Data Storage

Column family animal:

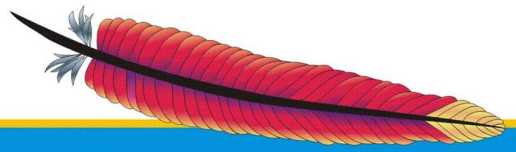
(enclosure1, t2, animal:type)	zebra
(enclosure1, t1, animal:size)	big
(enclosure1, t1, animal:type)	lion

Column family repairs:

(enclosure1, t1, repairs:cost)	1000 EUR

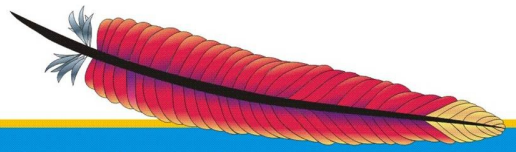
HBase – Code

```
HTable table = ...
Text row = new Text("enclosure1");
Text col1 = new Text("animal:type");
Text col2 = new Text("animal:size");
BatchUpdate update = new BatchUpdate(row);
update.put(col1, "lion".getBytes("UTF-8"));
update.put(col2, "big".getBytes("UTF-8"));
table.commit(update);

update = new BatchUpdate(row);
update.put(col1, "zebra".getBytes("UTF-8"));
table.commit(update);
```



HBase - Querying

- Retrieve a cell
- Retrieve a row
- Scan through range of rows



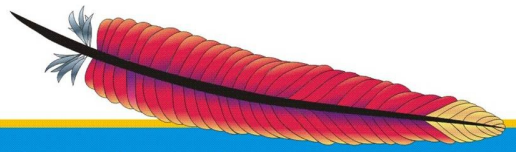
Related Projects

Related Projects

- Pig - incubator.apache.org/pig/
  - a high-level imperative language for analyzing large datasets

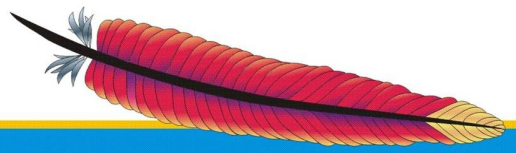
```
animals = LOAD 'mr/input/grep/animals';
f = FILTER animals BY $0 matches '(cow|dog)';
STORE f INTO 'output';
```

- Mahout – lucene.apache.org/mahout/
  - scalable machine learning libraries
- ZooKeeper – sf.net/projects/zookeeper
  - a reliable coordination system



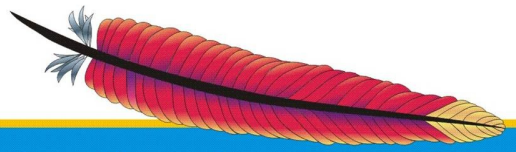
Related Projects (contd.)

- Thrift - developers.facebook.com/thrift/
  - "An inter-language RPC and serialization framework."
- Jaql - www.jaql.org
  - A parallel query language for JSON.
- Cascading - www.cascading.org
  - An API for building dataset processing flows.



Hadoop Status

- Hadoop Core 0.17.0 (April 2008)
  - Rack awareness for MapReduce
  - Pluggable Serializers
  - Deprecated code removed
- HBase 0.1.0 (March 2008)
  - Hadoop 0.16 version with bug fixes
- HBase 0.2.0 (April 2008?)
  - New HTable API
  - Scalability and robustness



Questions?

- http://hadoop.apache.org/
- tomwhite@apache.org