
Sperasoft
Your game dev partner
Big
Data
Systems
hadoop

Use Cases

- Before 2004 “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 etc.”
- Nutch search system at 2004 was effectively limited to 100M web pages

Hadoop History

- 2002: Doug Cutting started Nutch: crawler & search system
- 2003: GoogleFS paper
- 2004: Start of NDFS project (Nutch Distributed FS)
- 2004: Google MapReduce paper
- 2005: MapReduce implementation in Nutch
- 2006: HDFS and MapReduce to Hadoop subproject
- 2008: Yahoo! Production search index by a 10000-core Hadoop cluster
- 2008: Hadoop – top-level Apache project

Hadoop Objectives

- **Need to process Multi Petabyte Datasets**
- **Need to provide framework for reliable application execution**
- **Need to encapsulate nodes failures from application developer.**
  - Failure is expected, rather than exceptional.
  - The number of nodes in a cluster is not constant.
- **Need common infrastructure**
  - Efficient, reliable, Open Source Apache License

Hadoop

- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce
- Hadoop Common

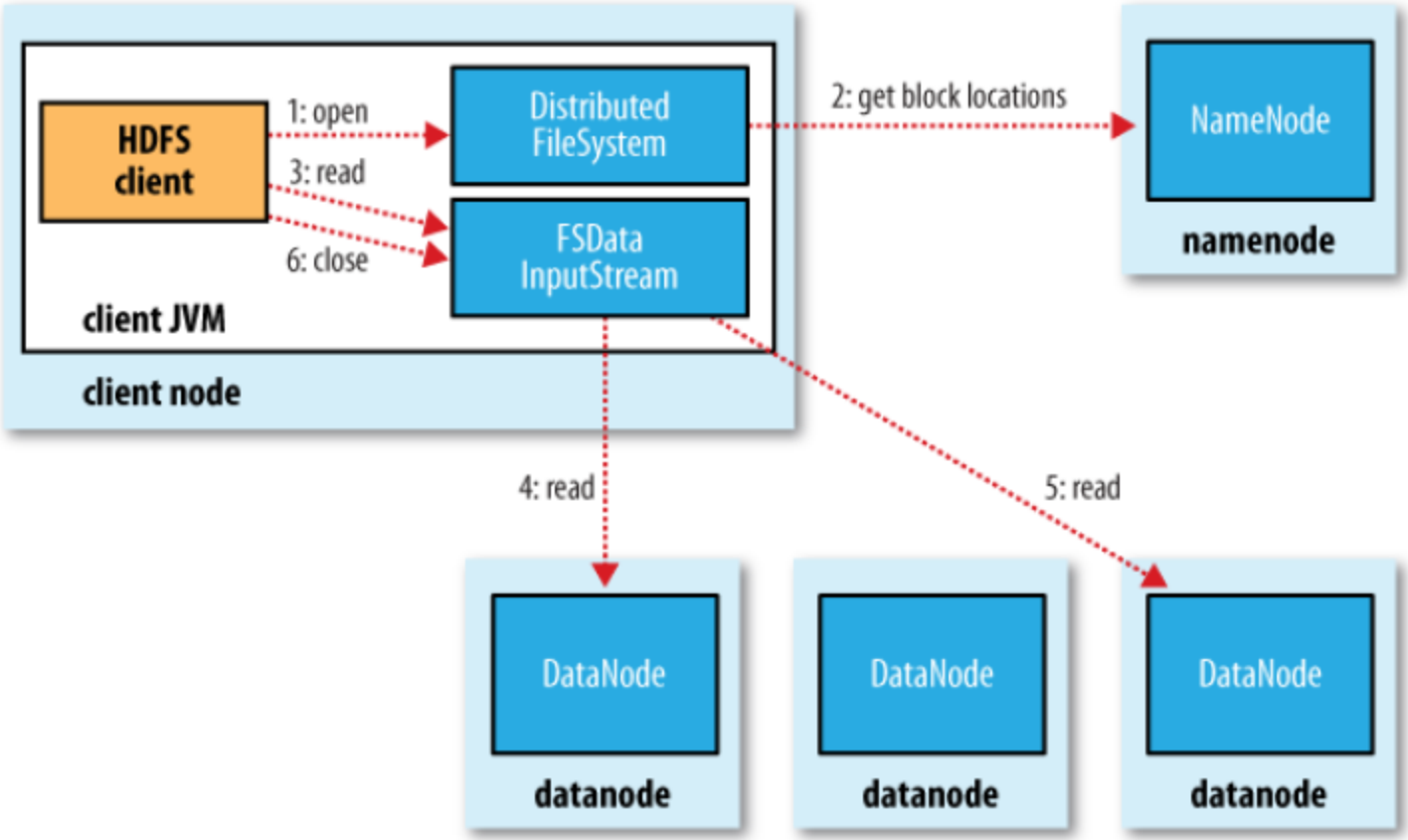
Goals of GFS/HDFS

- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 PB
- **Assumes Commodity Hardware**
  - Files are replicated to handle hardware failure
  - Detect failures and recovers from them
- **Optimized for Batch Processing**
  - Data locations exposed so that computations can move to where data resides
  - Provides very high aggregate bandwidth

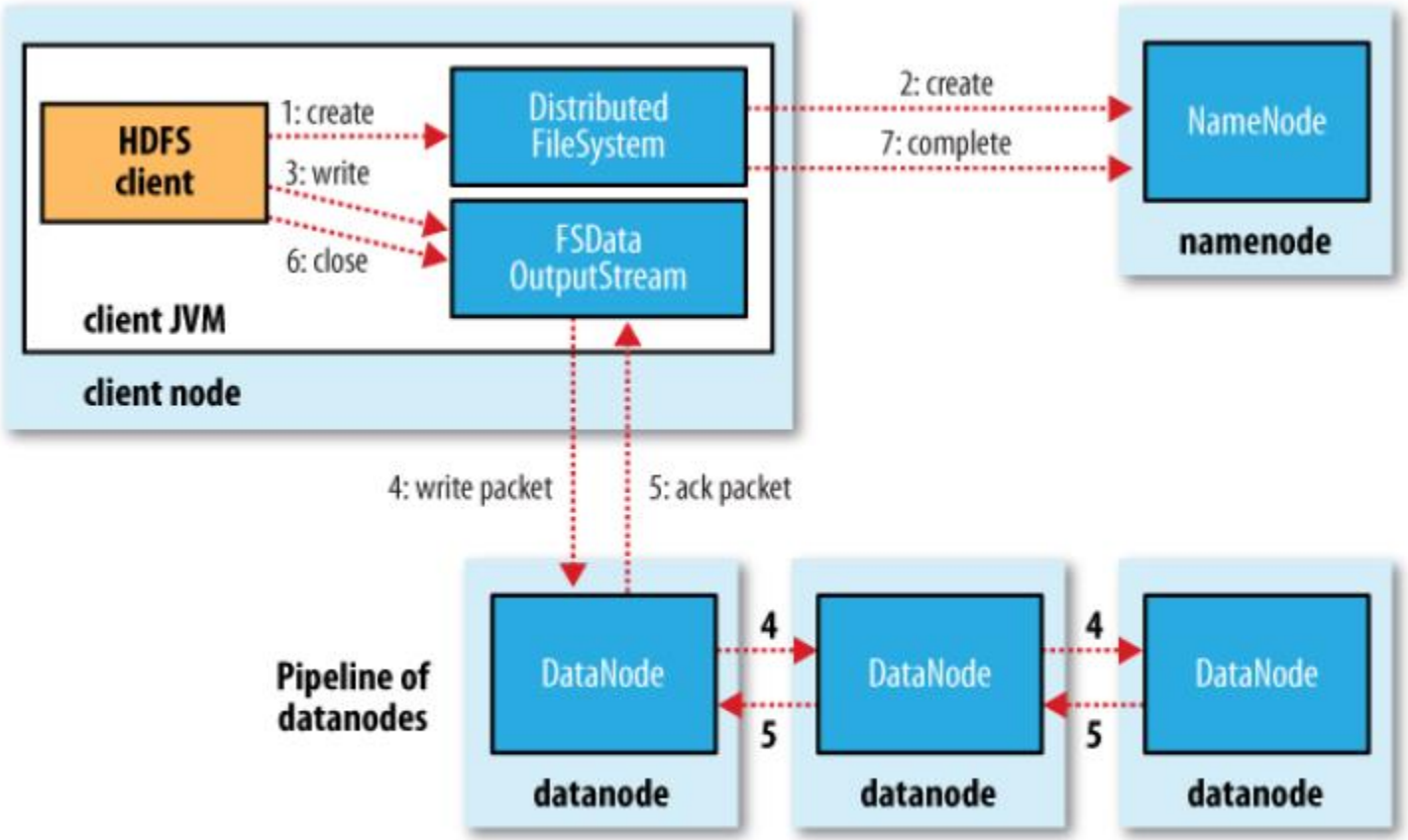
HFDS Details

- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- **Files are broken up into blocks**
  - Typically 128 MB block size
  - Each block replicated on multiple DataNodes
- **Intelligent Client**
  - Client can find location of blocks
  - Client accesses data directly from DataNode

Client reading data from HDFS



Client writing data to HDFS



Compression

Compression format	Tool	Algorithm	Filename extension	Multiple files	Splittable
DEFLATE[a]	N/A	DEFLATE	*.deflate*	No	No
gzip	*gzip*	DEFLATE	*.gz*	No	No
bzip2	*bzip2*	bzip2	*.bz2*	No	Yes
LZO	*lzop*	LZO	*.lzo*	No	No

HDFS User Interface

- **Java API**
- **Command Line**
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
  - hadoop dfsadmin –report
  - hadoop dfsadmin -decommission datanodename
- **Web Interface**
  - http://host:port/dfshealth.jsp

HDFS Web UI

Sperasoft
Your game dev partner

NameNode 'domU-12-31-38-00-68-47.compute-1.internal:9000'

Started: Fri Mar 20 00:31:09 UTC 2009
Version: 0.18, r
Compiled: Thu Mar 12 18:17:37 UTC 2009 by root
Upgrades: There are no upgrades in progress.

Browse the filesystem

Cluster Summary

8 files and directories, 6 blocks = 14 total. Heap Size is 4.94 MB / 992.31 MB (0%)

Capacity	:	587.08 GB
DFS Remaining	:	545.39 GB
DFS Used	:	96 KB
DFS Used%	:	0 %
Live Nodes	:	4
Dead Nodes	:	0

Live Datanodes : 4

Node	Last Contact	Admin State	Size (GB)	Used (%)	Used (%)	Remaining (GB)	Blocks
domU-12-31-38-00-51-B3	2	In Service	146.77	0		136.35	4
domU-12-31-38-00-52-03	2	In Service	146.77	0		136.35	6
domU-12-31-38-00-79-31	0	In Service	146.77	0		136.35	4
domU-12-31-38-00-79-58	2	In Service	146.77	0		136.35	6

Dead Datanodes : 0

Hadoop MapReduce

- **The Map-Reduce programming model**
  - Framework for distributed processing of large data sets
  - Pluggable user code runs in generic framework
- **Common design pattern in data processing**

  cat * | grep | sort | uniq -c | cat > file

  input | **map** | shuffle | **reduce** | output
- **Natural for:**
  - Log processing
  - Web search indexing
  - Ad-hoc queries

Lifecycle of a MapReduce Job

```
File  Edit  Options  Buffers  Tools  Java  Help

public class WordCount {

  public static class Map extends MapReduceBase implements
                Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
                    output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
  }}}

  public static class Reduce extends MapReduceBase implements
                Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
                       IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) { sum += values.next().get(); }
      output.collect(key, new IntWritable(sum));
  }}

  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }}

--:---  mapreduce.java   All L9     (Java/l Abbrev)----------------------------------------
Wrote /home/shivnath/Desktop/mapreduce.java
```

Map function

Reduce function

Run this program as a MapReduce job

MapReduce in Hadoop (1)

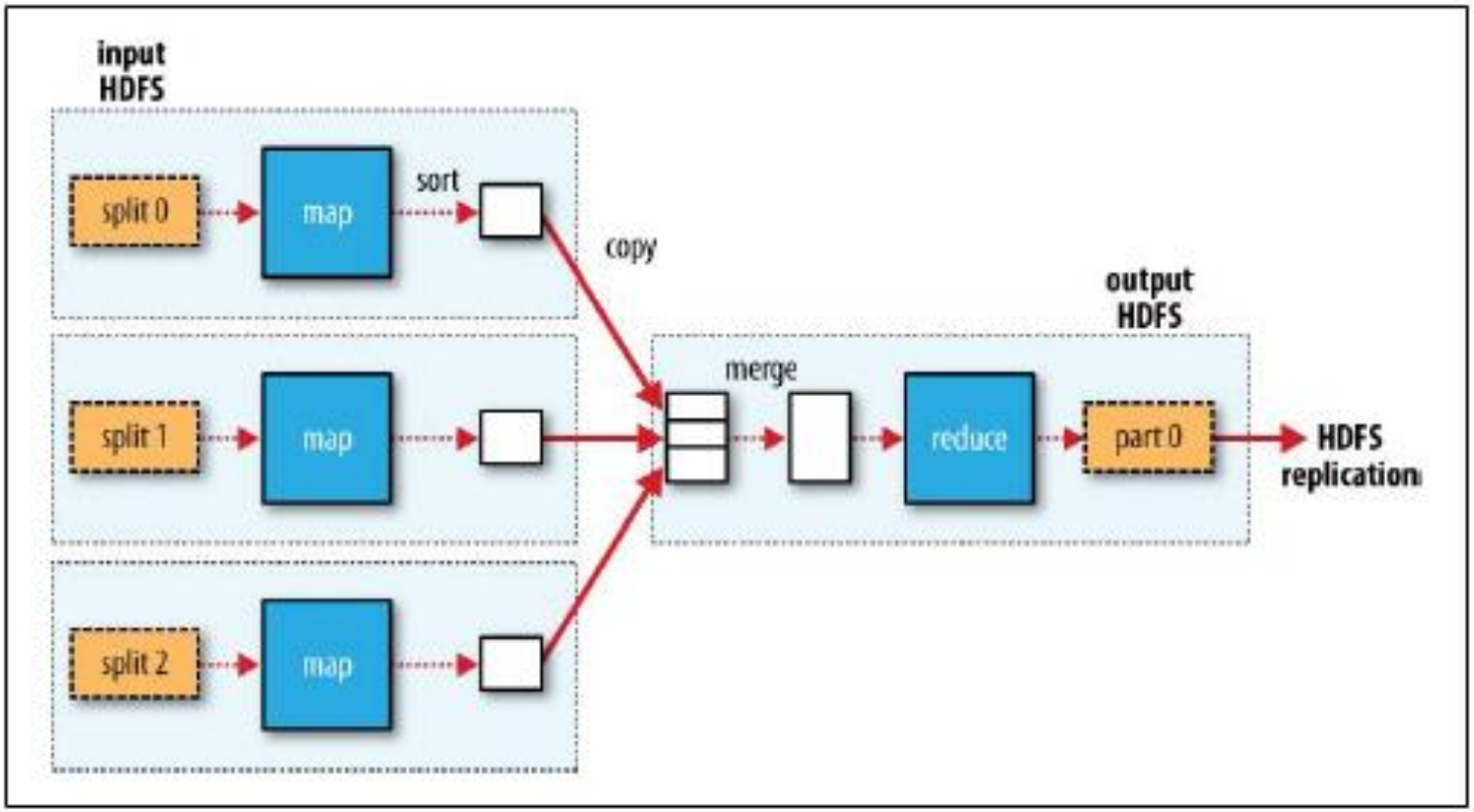


Figure 2-2. MapReduce data flow with a single reduce task

MapReduce in Hadoop (2)

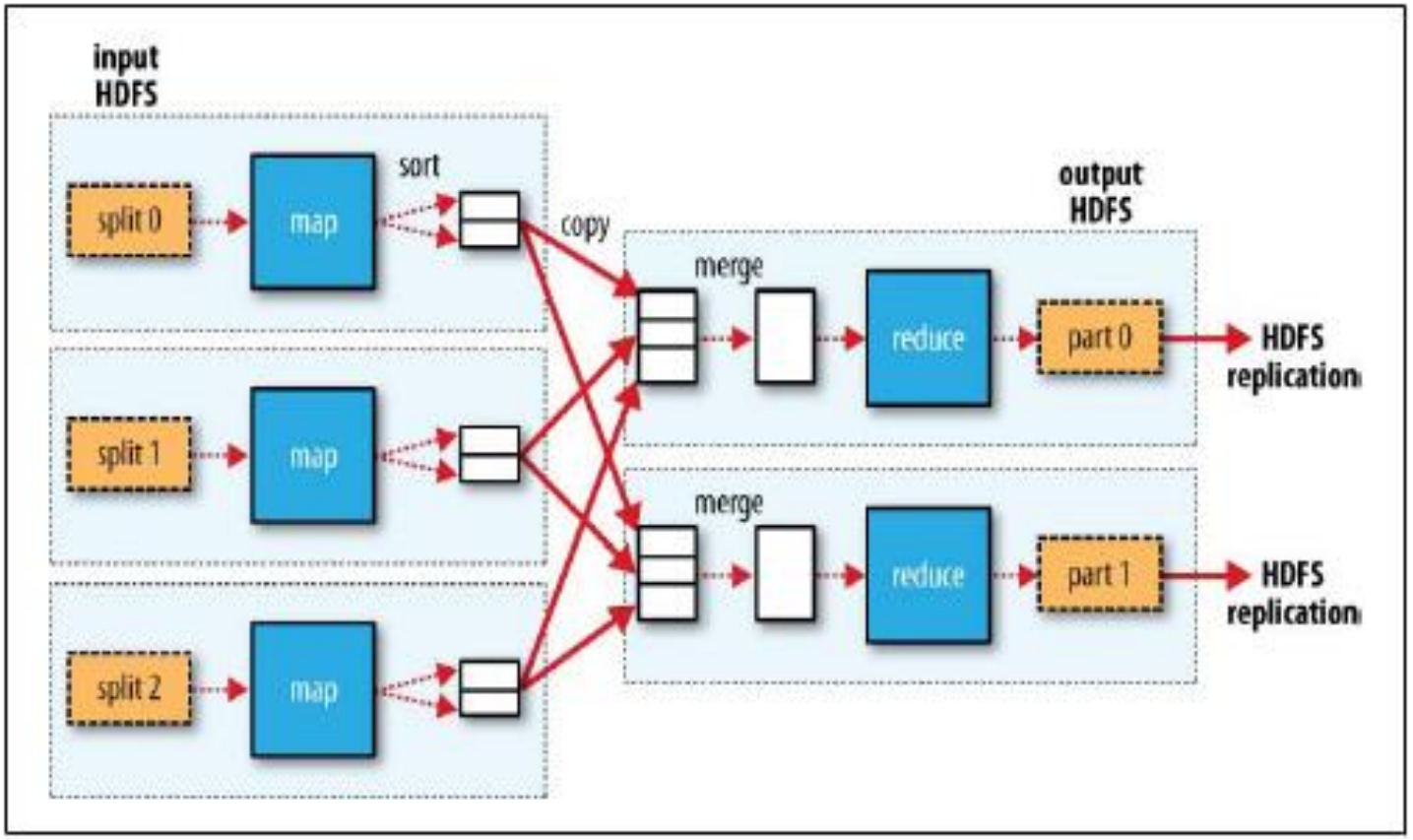


Figure 2-3. MapReduce data flow with multiple reduce tasks

MapReduce in Hadoop (3)

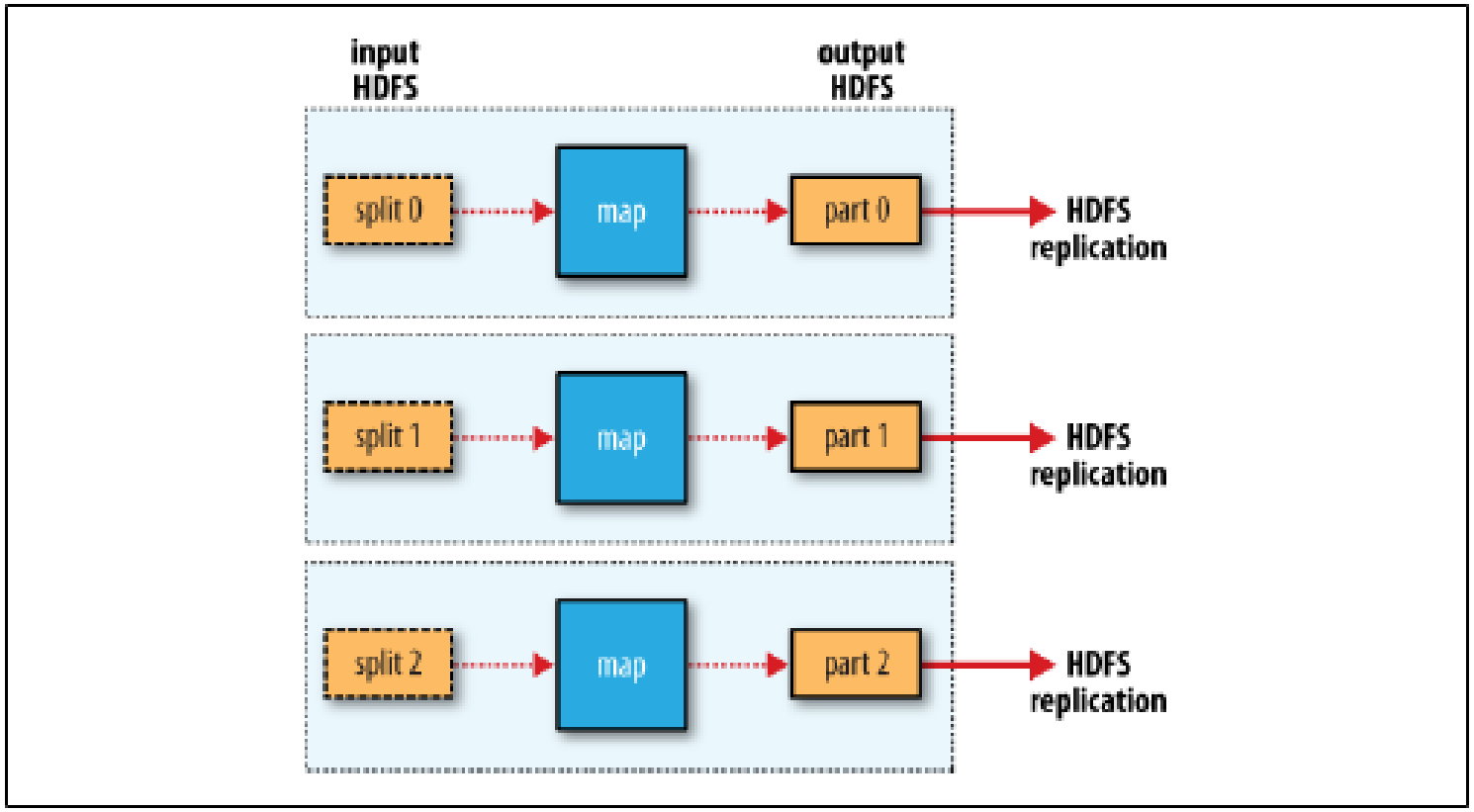
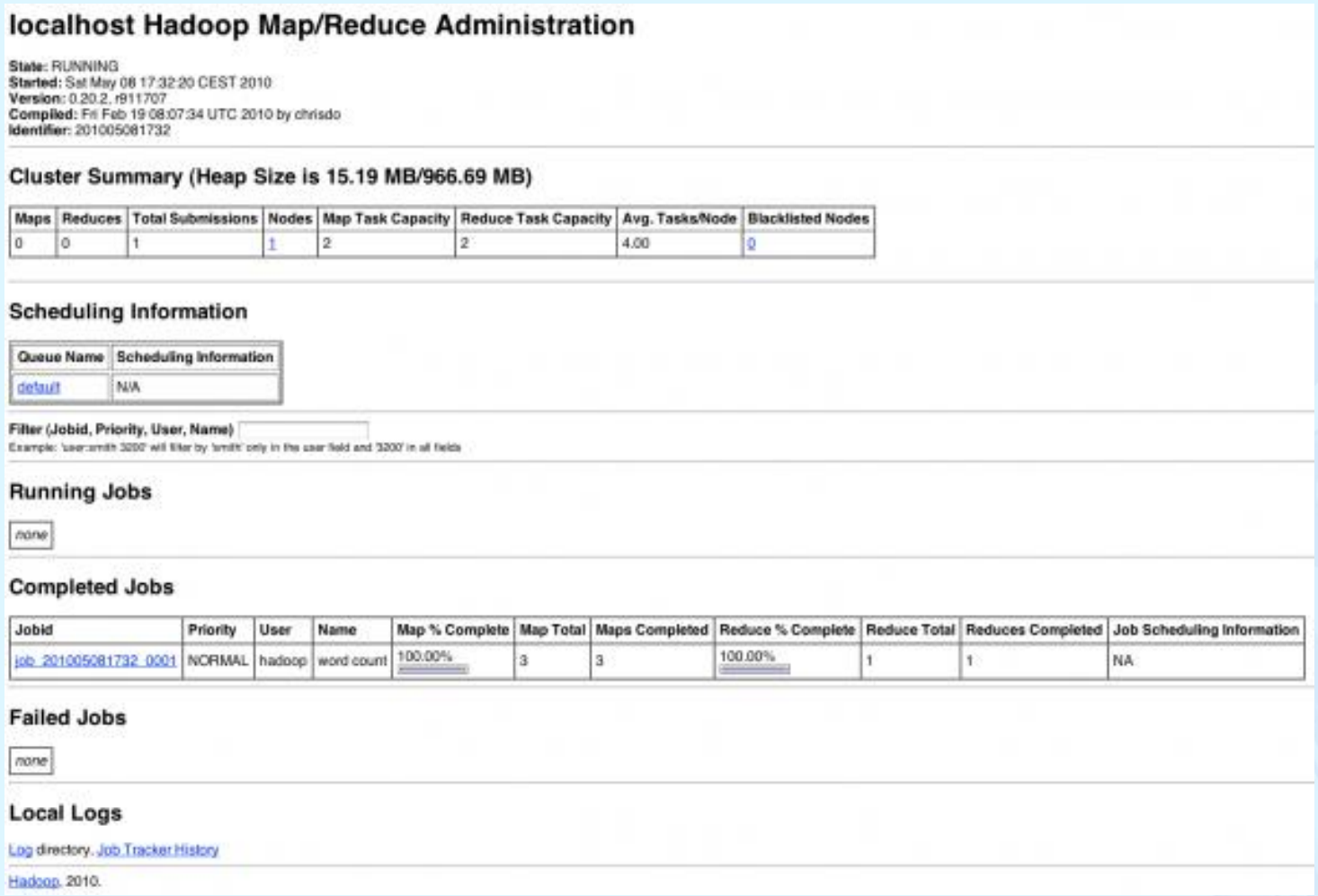


Figure 2-4. MapReduce data flow with no reduce tasks

Hadoop WebUI



localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Sat May 08 17:32:20 CEST 2010
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Identifier: 201005081732

Cluster Summary (Heap Size is 15.19 MB/966.69 MB)

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

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201005081732_0001	NORMAL	hadoop	word count	100.00%	3	3	100.00%	1	1	NA

Failed Jobs

none

Local Logs

Log directory, Job Tracker History

Hadoop, 2010.

Hadoop WebUI

Hadoop job_200709211549_0003 on localhost

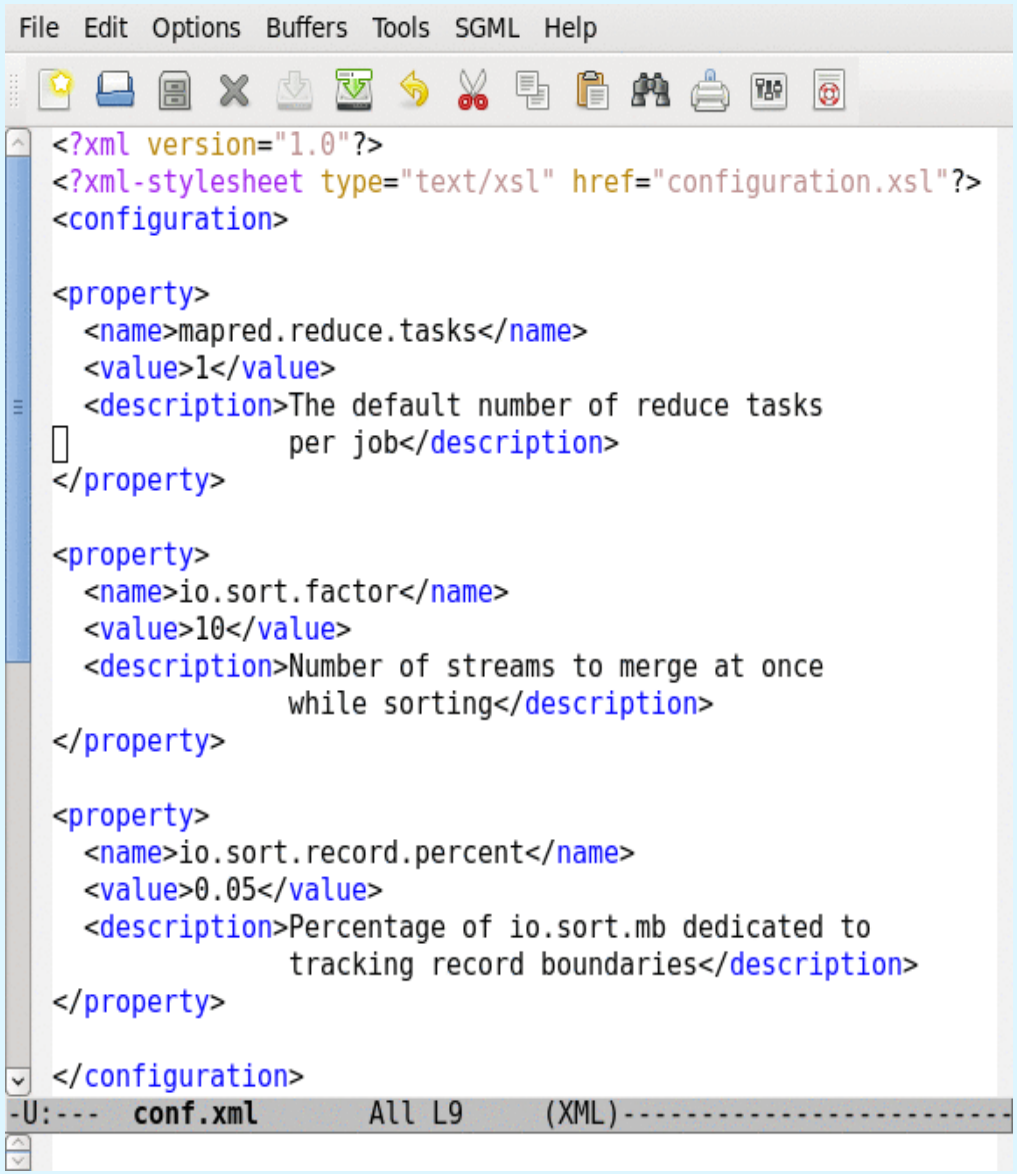
User: hadoop
Job Name: streamjob34453.jar
Job File: /usr/local/hadoop-datastore/hadoop-hadoop/mapred/system/job_200709211549_0003/job.xml
Status: Succeeded
Started at : Fri Sep 21 16:07:10 CEST 2007
Finished at: Fri Sep 21 16:07:26 CEST 2007
Finished in: 16sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	3	0	0	3	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched map tasks	0	0	3
	Launched reduce tasks	0	0	1
	Data-local map tasks	0	0	3
Map-Reduce Framework	Map input records	77,637	0	77,637
	Map output records	103,909	0	103,909
	Map input bytes	3,659,910	0	3,659,910
	Map output bytes	1,083,767	0	1,083,767
	Reduce input groups	0	85,095	85,095
	Reduce input records	0	103,909	103,909
	Reduce output records	0	85,095	85,095

Change priority from NORMAL to: VERY_HIGH HIGH LOW VERY_LOW

Hadoop Configuration



- 190+ parameters in Hadoop
- Set manually or defaults are used

Summary

Pro:

- Cheap components
- Replication
- Fault tolerance
- Parallel processing
- Free license
- Linear scalability
- Amazon support

Con:

- No realtime
- Difficult to add MR tasks
- File edit is not supported
- High support cost

Examples

- Distributed Grep
- Count of URL Access Frequency
- Reverse Web-Link Graph
- Inverted Index

Hadoop

- Streaming
- Hive
- Pig
- HBase

Hadoop Streaming (1)

API to MapReduce that uses Unix standard streams as the interface between Hadoop and your program

MAP: map.rb

```
#!/usr/bin/env ruby
STDIN.each_line do |line|
  val = line
  year, temp, q = val[15,4], val[87,5], val[92,1]
  puts "#{year}\t#{temp}" if (temp != "+9999" && q =~ /[01459]/)
end
```

LOCAL EXECUTION

```
% cat input/ncdc/sample.txt | map.rb
1950    +0000
1950    +0022
1950    -0011
1949    +0111
1949    +0078
```

Hadoop Streaming (2)

REDUCE: reduce.rb

```
#!/usr/bin/env ruby
last_key, max_val = nil, 0
STDIN.each_line do |line|
  key, val = line.split("\t")
  if last_key && last_key != key
    puts "#{last_key}\t#{max_val}"
    last_key, max_val = key, val.to_i
  else
    last_key, max_val = key, [max_val, val.to_i].max
  end
end
puts "#{last_key}\t#{max_val}" if last_key
```

LOCAL EXECUTION

```
% cat input/ncdc/sample.txt | map.rb | sort | reduce.rb
1949    111
1950    22
```

Hadoop Streaming (3)

Sperasoft
Your game dev partner

HADOOP EXECUTION

```
% hadoop jar \
   $HADOOP_INSTALL/contrib/streaming/hadoop-*-streaming.jar \
   -input input/ncdc/sample.txt \
   -output output \
   -mapper map.rb \
   -reducer reduce.rb
```

Hive: overview

- Intuitive
  - Make the unstructured data looks like tables regardless how it really lay out
  - SQL based query can be directly against these tables
  - Generate specify execution plan for this query
- What's Hive
  - A data warehousing system to store structured data on Hadoop file system
  - Provide an easy query these data by execution Hadoop MapReduce plans

Hive: architecture

Map Reduce

HDFS

Web UI + Hive CLI + JDBC/ODBC

Browse, Query, DDL

User-defined Map-reduce Scripts

MetaStore

Thrift API

Hive QL

Parser

Planner

Execution

Optimizer

UDF/UDAF

substr
sum
average

SerDe

CSV
Thrift
Regex

FileFormats

TextFile
SequenceFile
RCFile

Hive: shell

```
hive> SHOW TABLES;
hive> CREATE TABLE shakespeare (freq
   INT, word STRING)  ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE;
hive> DESCRIBE shakespeare;
```

loading data…

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

```
-- max_temp.pig: Finds the maximum temperature by year

records = LOAD 'input/ncdc/micro-tab/sample.txt'
          AS (year:chararray, temperature:int, quality:int);

filtered_records = FILTER records
          BY temperature != 9999
          AND (quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9);

grouped_records = GROUP filtered_records BY year;

max_temp = FOREACH grouped_records
          GENERATE group, MAX(filtered_records.temperature);

DUMP max_temp;
```

RDBMS scaling story (1)

Initial public launch

Move from local workstation to shared, remote hosted MySQL instance with a well-defined schema.

Service becomes more popular; too many reads hitting the database

Add memcached to cache common queries. Reads are now no longer strictly ACID; cached data must expire.

Service continues to grow in popularity; too many writes hitting the database

Scale MySQL vertically by buying a beefed up server with 16 cores, 128 GB of RAM,

and banks of 15 k RPM hard drives. Costly.

RDBMS scaling story (1)

New features increases query complexity; now we have too many joins

Denormalize your data to reduce joins.

Rising popularity swamps the server; things are too slow

Stop doing any server-side computations.

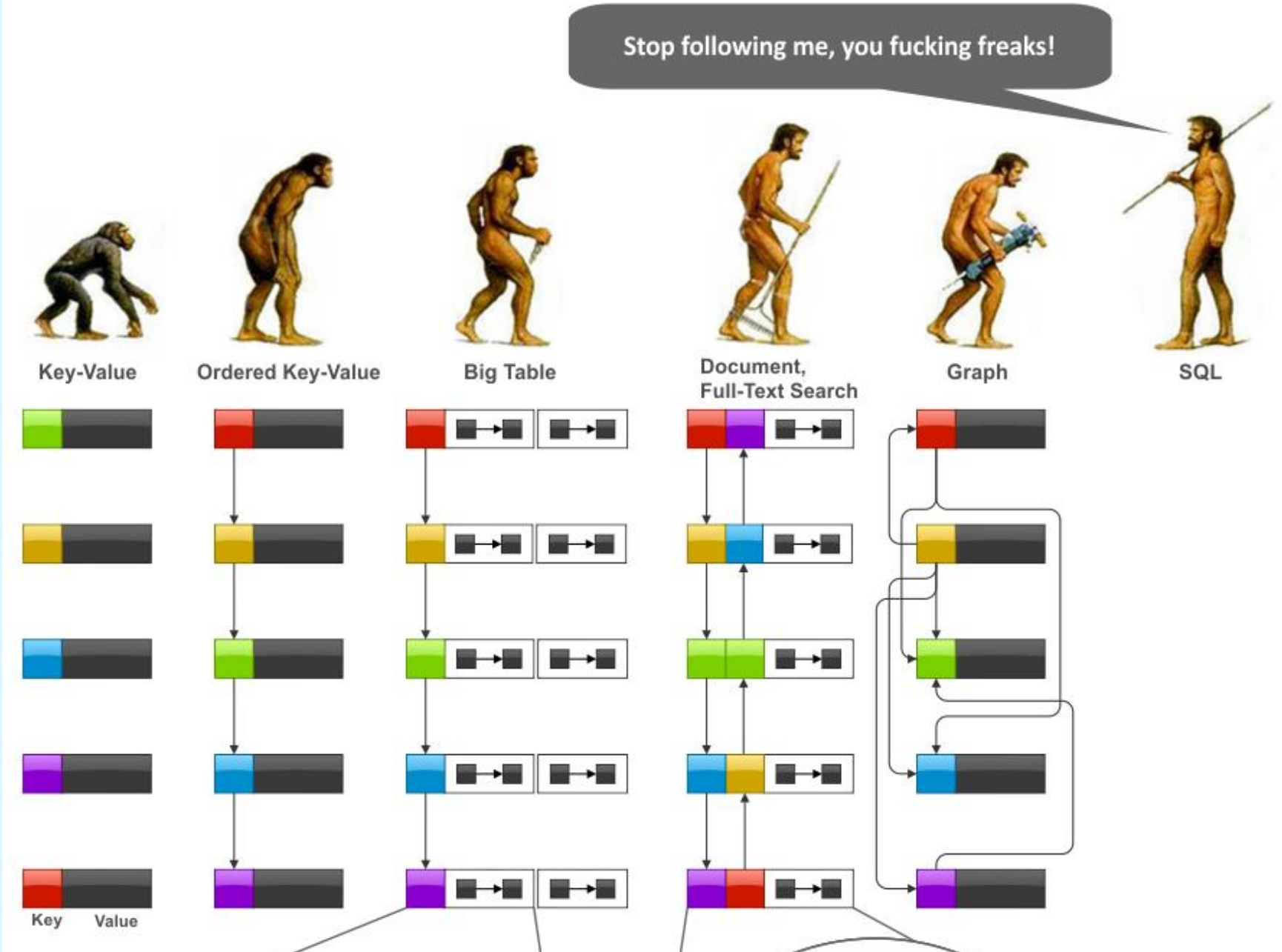
Some queries are still too slow

Periodically prematerialize the most complex queries, try to stop joining in most cases.

Reads are OK, but writes are getting slower and slower

Drop secondary indexes and triggers (no indexes?).

NoSQL


Stop following me, you fucking freaks!
Key-Value
Ordered Key-Value
Big Table
Document,
Full-Text Search
Graph
SQL
Key
Value

Hbase: differences from RDBMS

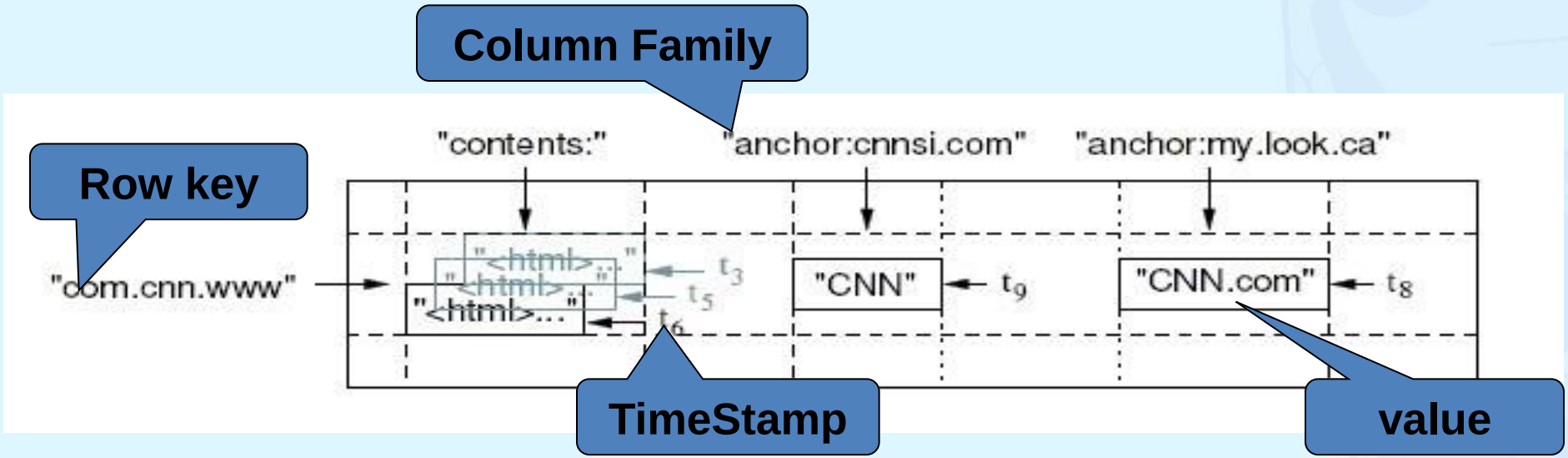
- Tables have one primary index, the *row key*
- No join operators
- Data is unstructured and untyped
- No accessed or manipulated via SQL
  - Programmatic access via Java, REST, or Thrift APIs
- There are three types of lookups:
  - Fast lookup using row key and optional timestamp
  - Full table scan
  - Range scan from region start to end

Hbase: benefits over RDBMS

- Automatic partitioning
- Scale linearly and automatically with new nodes
- Commodity hardware
- Fault tolerance: Apache Zookeeper
- Batch processing: Apache Hadoop

Hbase: data model

- Tables are sorted by Row
- Table schema only define it's *column families* .
  - Each family consists of any number of columns
  - Each column consists of any number of versions
  - Columns only exist when inserted, NULLs are free.
  - Columns within a family are sorted and stored together
- Everything except table names are byte[]
- (Row, Family: Column, Timestamp) → Value



Hbase: members

- *Master*
  - Responsible for monitoring region servers
  - Load balancing for regions
  - Redirect client to correct region servers
- *regionserver* slaves
  - Serving requests (Write/Read/Scan) of Client
  - Send HeartBeat to Master

Hbase: shell

```
$ hbase shell
> create 'test', 'data'
0 row(s) in 4.3066 seconds
> list
test
1 row(s) in 0.1485 seconds
> put 'test', 'row1', 'data:1', 'value1'
0 row(s) in 0.0454 seconds
> put 'test', 'row2', 'data:2', 'value2'
0 row(s) in 0.0035 seconds
> scan 'test'
ROW COLUMN+CELL
row1 column=data:1, timestamp=1240148026198, value=value1
row2 column=data:2, timestamp=1240148040035, value=value2
2 row(s) in 0.0825 seconds
```

Hbase: Web UI

Master: localhost:60000

Local logs, Thread Dump, Log Level

Master Attributes

Attribute Name	Value	Description
HBase Version	0.91.0-SNAPSHOT, r1127782	HBase version and svn revision
HBase Compiled	Thu May 26 10:28:47 CEST 2011, larsgeorge	When HBase version was compiled and by whom
Hadoop Version	0.20-append-r1057313, r1057313	Hadoop version and svn revision
Hadoop Compiled	Wed Feb 9 22:25:52 PST 2011, Stack	When Hadoop version was compiled and by whom
HBase Root Directory	hdfs://localhost:8020/hbase	Location of HBase home directory
HBase Cluster ID	698e057d-78ac-4d01-8db9-3cec937bc619	Unique identifier generated for each HBase cluster
Load average	4	Average number of regions per regionserver. Naive computation.
Zookeeper Quorum	localhost:2181	Addresses of all registered ZK servers. For more, see zk dump.

Currently running tasks

No tasks currently running on this node.

Catalog Tables

Table	Description
-ROOT-	The -ROOT- table holds references to all .META. regions.
.META.	The .META. table holds references to all User Table regions

User Tables

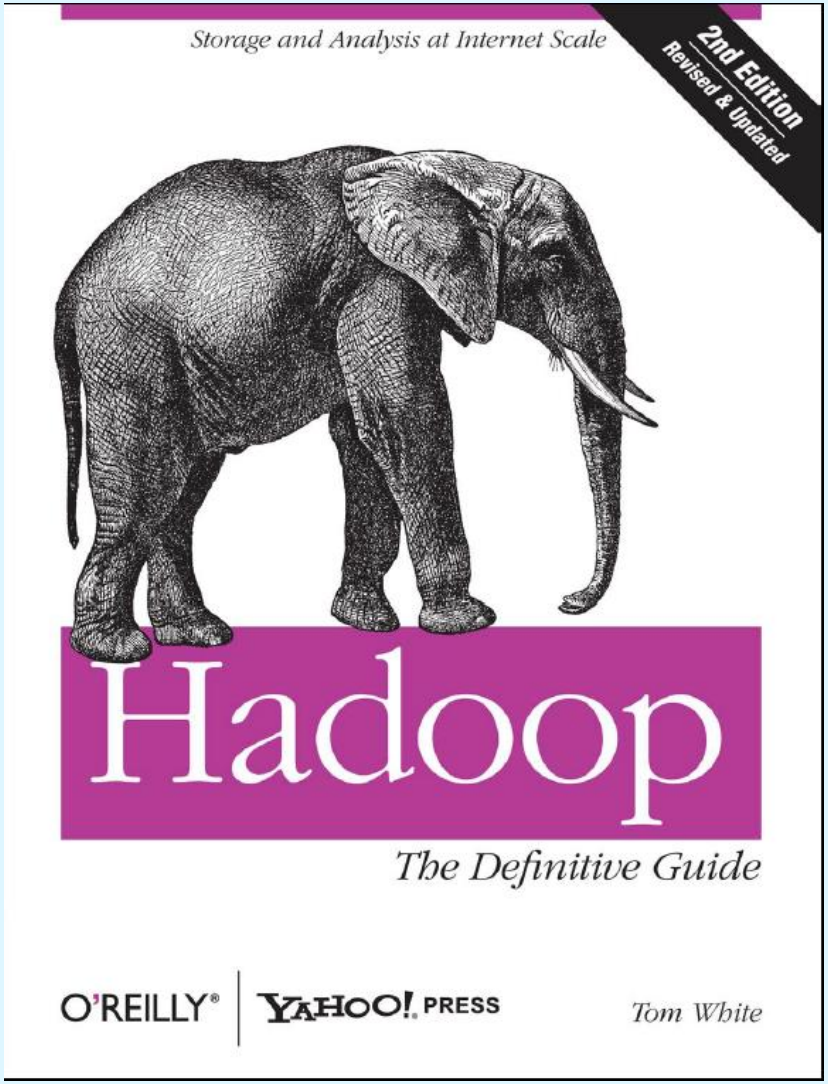
3 table(s) in set.

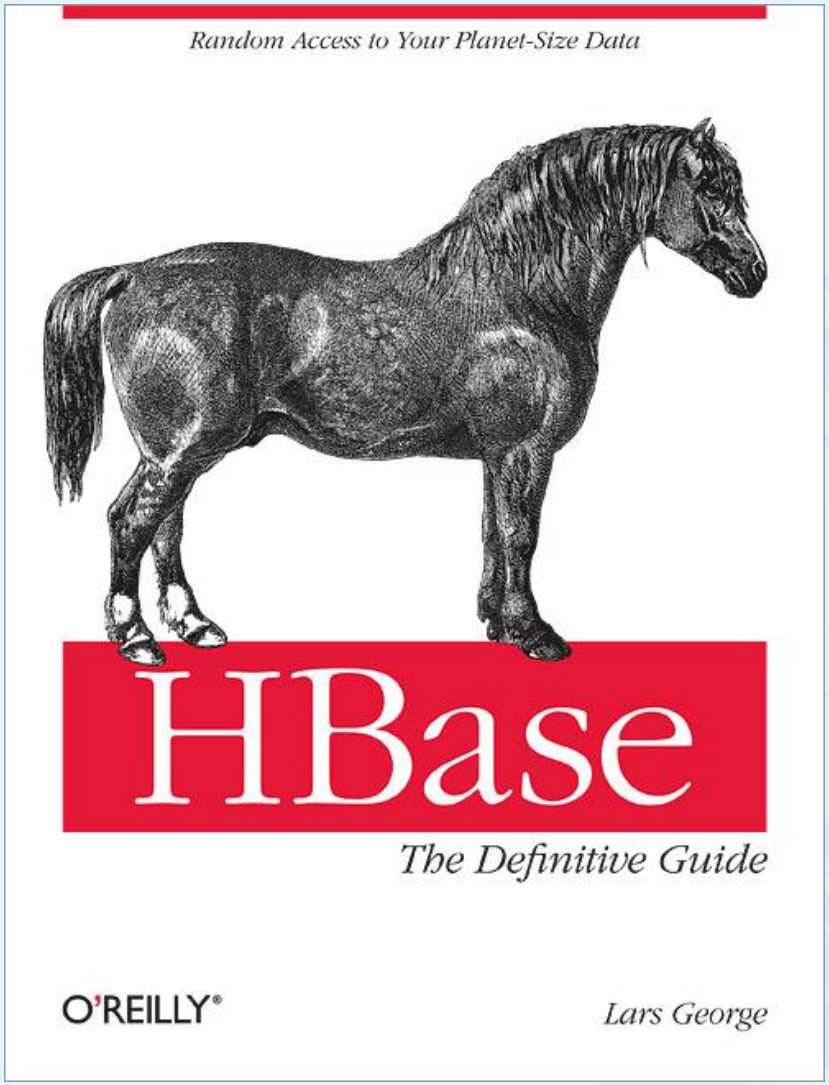
Table	Description
testtable	{NAME => 'testtable', FAMILIES => [{NAME => 'colfam1', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '3', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}
user	{NAME => 'user', DEFERRED_LOG_FLUSH => 'false', READONLY => 'false', MEMSTORE_FLUSHSIZE => '67108864', MAX_FILESIZE => '268435456', FAMILIES => [{NAME => 'data', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '3', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}
usertable	{NAME => 'usertable', FAMILIES => [{NAME => 'family', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', TTL => '2147483647', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}

Who uses Hadoop?

- Amazon
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

Books


Storage and Analysis at Internet Scale
2nd Edition
Revised & Updated
Hadoop
The Definitive Guide
O'REILLY
YAHOO! PRESS
Tom White


Random Access to Your Planet-Size Data
HBase
The Definitive Guide
O'REILLY
Lars George