# How to properly misuse Hadoop

**Marcel Huntemann**
**NERSC tutorial session**
**2/12/13**

# History

- Created by Doug Cutting (also creator of Apache Lucene).

2002

- Origin in Apache Nutch (open source web search engine).
- Realized architecture wouldn't scale to the billions of pages on the Web.

2003

- Paper about Google's distributed file system, GFS. (http://research.google.com/archive/gfs.html)

2004

- Open source implementation, Nutch Distributed File System (NDFS).
- Paper about Google's MapReduce. (http://research.google.com/archive/mapreduce.html)

# History (cont.)

2005

- All major Nutch algorithms ported to run using MapReduce and NDFS.

2006

- Independent subproject of Lucene called Hadoop.
- Doug Cutting joins Yahoo!.

2008

- Hadoop made top-level project at Apache.
- Yahoo!’s productions search index generated by a 10,000-core Hadoop cluster.
- Terabyte sort record, 209 seconds on 910-node cluster.

  [later 68 seconds by Google, then 62 seconds by Yahoo!, now at 55 seconds with GCE and MapR].

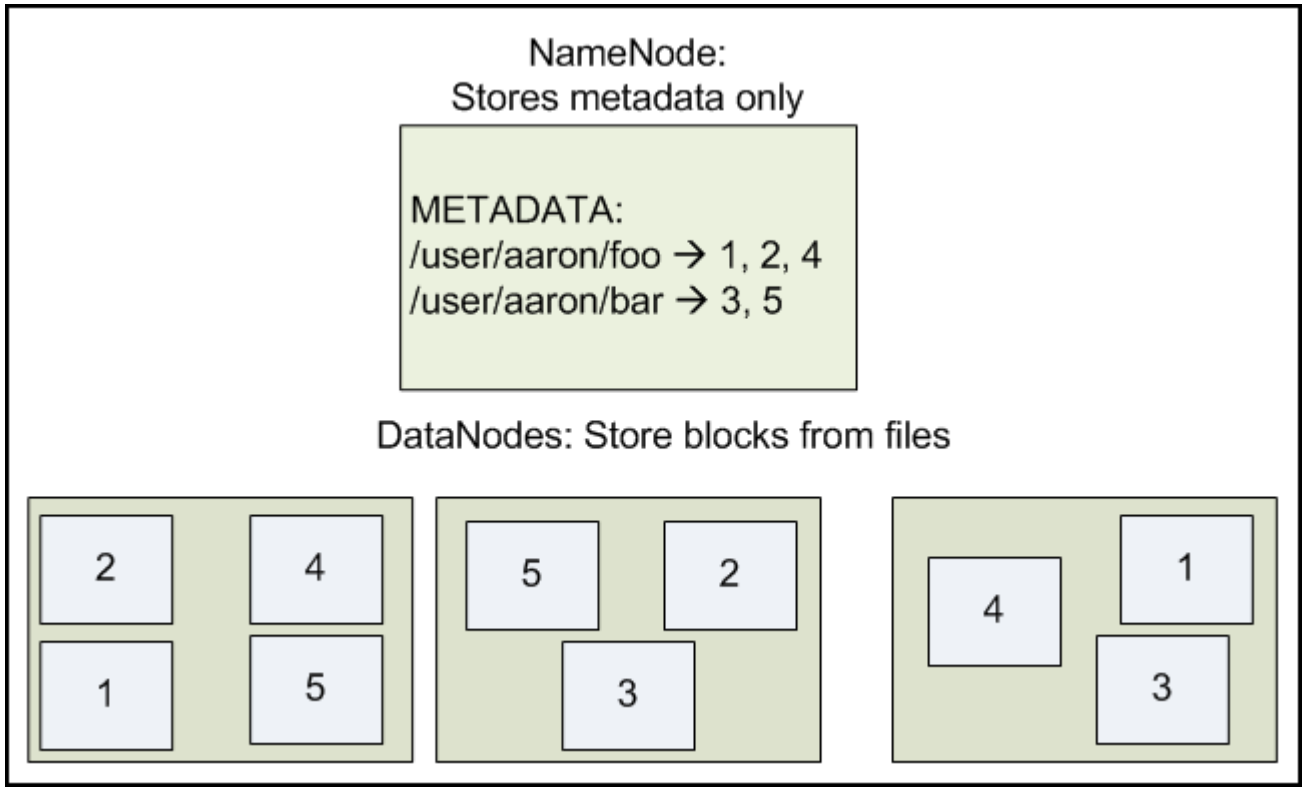
# What is hadoop?

- Open source software project.
- Framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- Designed to scale up from single servers to thousands of machines with a very high degree of fault tolerance.
  - Achieved via detection and handling of failures at the application layer.
- Consist of several modules, with the two main ones being:
  1. Hadoop Distributed File System (HDFS)
  2. MapReduce

# HDFS

- Distributed file system with a master/slave architecture.
- Single NameNode, manages the file system namespace and regulates access to files by clients.
- Multiple DataNodes (usually one per node in the cluster), manage storage attached to the nodes that they run on.
- Files are (internally) split into one or more blocks.
- Blocks are stored on a set of DataNodes.
- Built in replication.

# HDFS (cont.)




Source: http://developer.yahoo.com/hadoop/tutorial/module2.html

# MapReduce

- Programming model for processing data.
- Breaks processing into two phases:
  1. Map phase
  2. Reduce phase

  (Programmer specifies both functions.)
- Each phase has key-value pairs as input and output.

  (Types chosen by programmer.)

# MapReduce (word count example)

Pseudo code:

```
map (line_offset, line):
    for each word in line:
        emit (word, 1)


reduce (word, values):
    sum = 0
    for each value in values:
        sum = sum + value
    emit (word, sum)
```

# MapReduce (word count input)

*marcelh@palmdale:~$* ll wordCountExample/input/

total 8.0K

-rw-rw---- 1 marcelh marcelh 27 Feb  8 23:25 theTruth.txt

-rw-rw---- 1 marcelh marcelh 72 Feb  8 23:26 theTruth.txt.addendum

*marcelh@palmdale:~$* cat wordCountExample/input/theTruth.txt

**I don't like giving talks.**

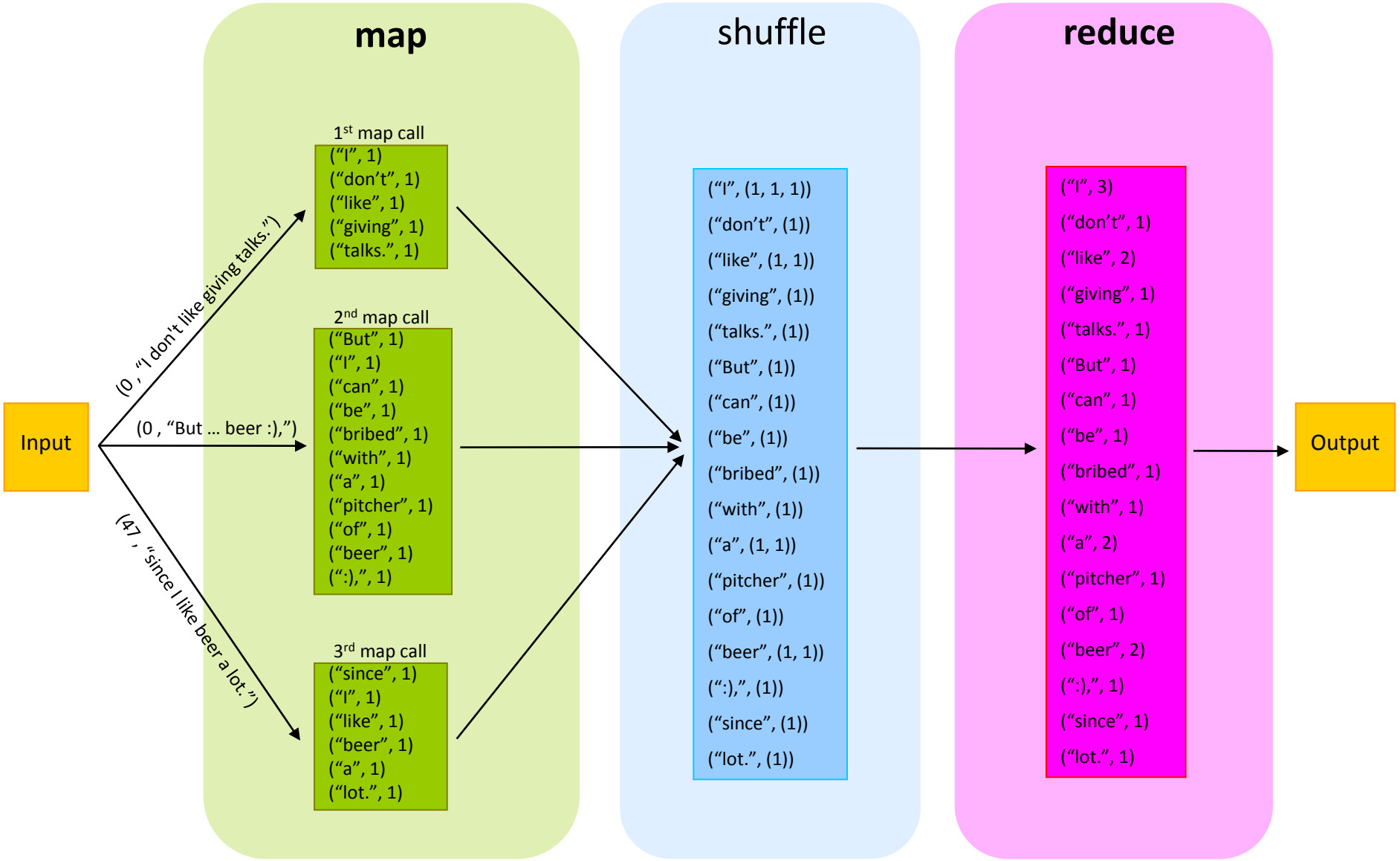
*marcelh@palmdale:~$* cat wordCountExample/input/theTruth.txt.addendum

**But I can be bribed with a pitcher of beer :),**

**since I like beer a lot.**

*marcelh@palmdale:~$*

# MapReduce (word count data flow)



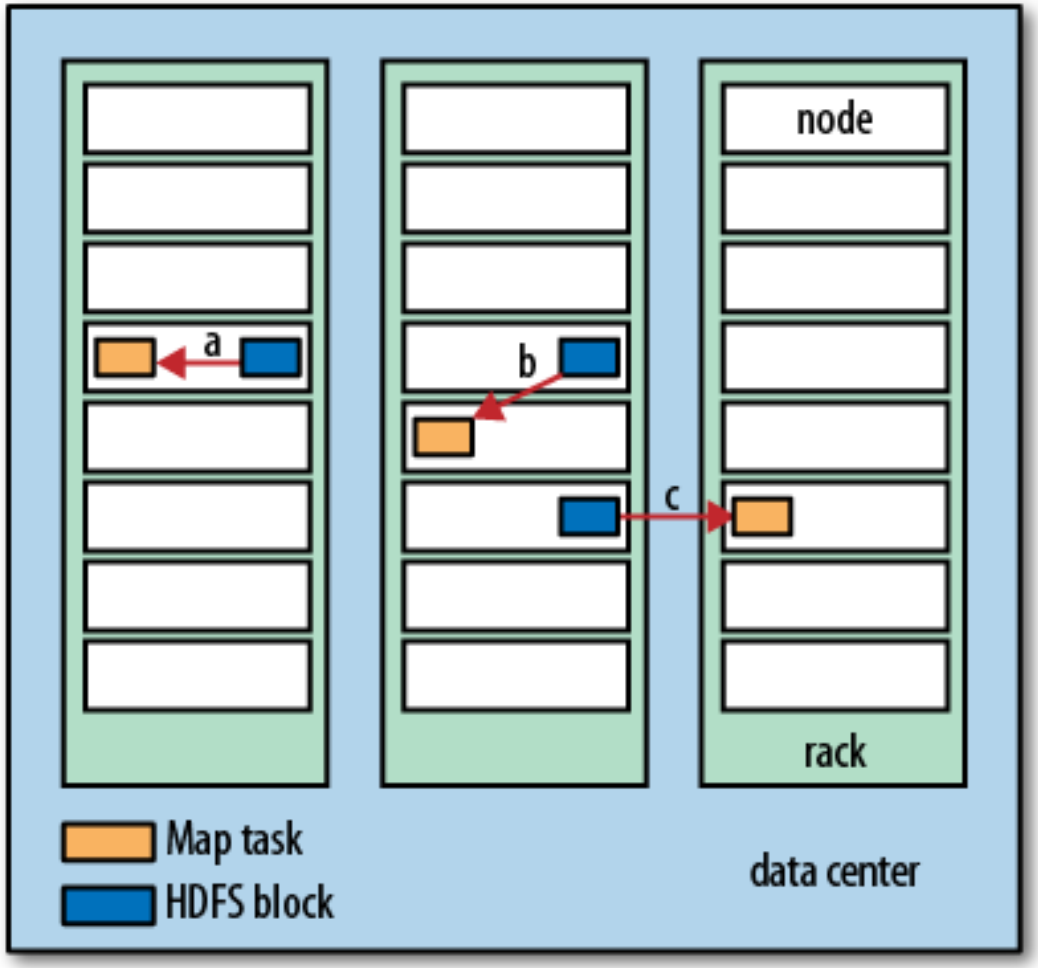
# MapReduce (word count output)

```
marcelh@palmdale:~$ hadoop jar dev/hadoop/hadoop-examples-*.jar wordcount wordCountExample/input/ wordCountExample/output/
<SNIP>
13/02/08 23:37:02 INFO mapred.JobClient:  map 100% reduce 100%
13/02/08 23:37:02 INFO mapred.JobClient: Job complete: job_local_0001
<SNIP>
marcelh@palmdale:~$ cat wordCountExample/output/part-r-00000
:), 1
But 1
I 3
a 2
be 1
beer 2
bribed 1
can 1
don't 1
giving 1
like 2
lot. 1
of 1
pitcher 1
since 1
talks. 1
with 1
marcelh@palmdale:~$
```

# All together now

- MapReduce job consists of input data, MapReduce program and configuration information.
- Job gets divided into tasks (map tasks and reduce tasks).
- Two types of nodes control job execution process:
  - Jobtracker, coordinates all jobs and keeps track of progress.
  - Tasktrackers, run tasks and send progress reports.
- Input gets divided into splits.
- One map task per split, which runs map function on each record in the split.
- Data locality optimization.

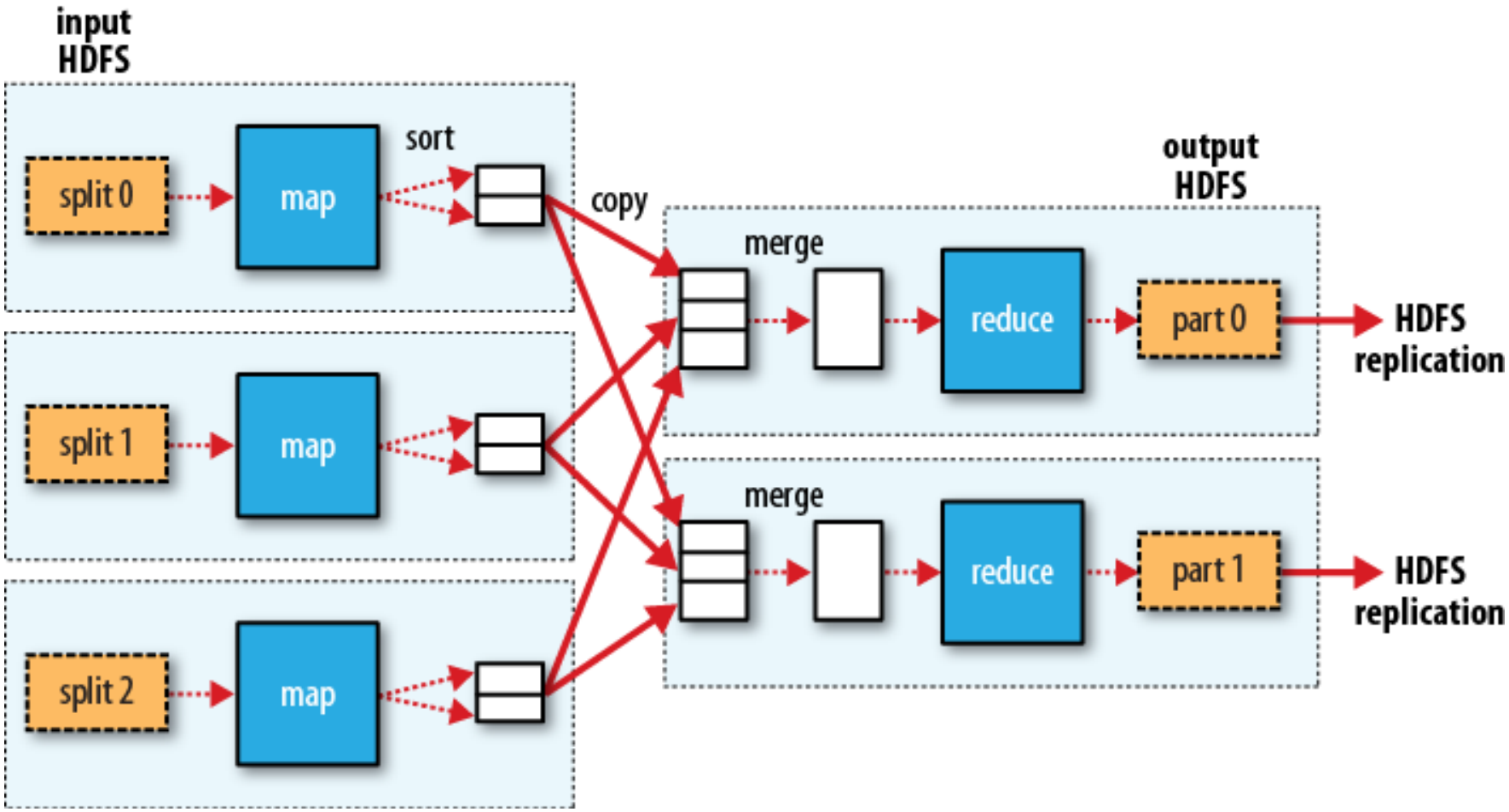
# All together now (data locality optimization)




Source: Hadoop: The Definitive Guide

Data-local (a), rack-local (b), and off-rack (c) map tasks

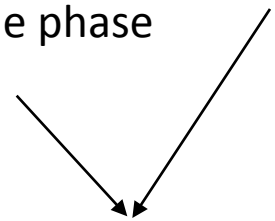
# All together now (data flow)




Source: Hadoop: The Definitive Guide

# JGI setup and experience

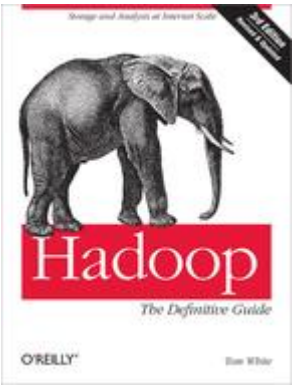
- No dedicated Hadoop cluster -> on-the-fly approach
- No HDFS -> GPFS
- Centralized Java/Hadoop installation -> local installation
- Memory abuse -> *.java.child.opts -Xmx
- Unbalanced reducer running time -> Partitioner
- Localize DBs, etc. -> Rob's localize_file
- Large data sets vs. 12 hours and number of nodes
- Large data sets vs. shuffle phase



No reducer

# Eager beaver?

http://developer.yahoo.com/hadoop/tutorial/




Tom White

Hadoop: The Definitive Guide

O'Reilly Media, Inc.

Donald Miner, Adam Shook

MapReduce Design Patterns

O'Reilly Media, Inc.

# Acknowledgements


- ✓ Kostas Mavrommatis (WIP)
- ✓ Amrita Pati
- ✓ Shane Canon
- ✓ Seung-Jin Sul
- ✓ Rob Egan
- ✓ NERSC consulting team

# Questions?