



Hadoop

OSDC.TW 2008 (Taipei, Apr 12, 2008)

Vivek Ratan

Architect, Grid Computing Group

Yahoo! Bangalore (India)

Purpose of the talk

- **Target audience:**
 - Users of Hadoop
 - Developers who want to contribute to Hadoop
- **Describe Hadoop**
 - Hadoop Distributed File System
 - Map-Reduce
- **Motivation for the Open source community**

Hadoop

Hadoop

- **Framework for running applications on large clusters built of commodity hardware**
 - from one machine to thousands
- **Lets one easily write and run applications that process vast amounts of data (petabytes).**
 - User submits Map-Reduce job to Hadoop
 - Hadoop breaks jobs into many tasks, executes them in parallel
 - Hadoop deals with fault tolerance, scale, etc.

Hadoop

- **Framework for running applications on large clusters built of commodity hardware**
 - from one machine to thousands
- **Lets one easily write and run applications that process vast amounts of data (petabytes).**
 - User submits Map-Reduce job to Hadoop
 - Hadoop breaks jobs into many tasks, executes them in parallel
 - Hadoop deals with fault tolerance, scale, etc.
- **Open Source**
 - Top level Apache project

Hadoop is good for ...

Hadoop is good for ...

- **Batch data processing, not real-time / user facing**
 - Log Processing
 - Document Analysis and Indexing
 - Web Graphs and Crawling
- **Highly parallel, data intensive, distributed applications**
 - Bandwidth to data is a constraint
 - Number of CPUs is a constraint

Hadoop is ...

Hadoop is ...

- **Scalable: store and process petabytes, scale by adding HW**

Hadoop is ...

- **Scalable: store and process petabytes, scale by adding HW**
 - horizontal scaling

Hadoop is ...

- **Scalable: store and process petabytes, scale by adding HW**
 - horizontal scaling
- **Reliable: data is replicated, failed tasks are rerun**

Hadoop is ...

- **Scalable: store and process petabytes, scale by adding HW**
 - horizontal scaling
- **Reliable: data is replicated, failed tasks are rerun**
- **Used by many Enterprises**
 - Yahoo is the biggest contributor and user
 - Amazon, Facebook, IBM, lots of smaller companies
 - Many Applications run on 20-100 nodes, some run on thousands (<http://wiki.apache.org/hadoop/PoweredBy>)

Hadoop is ...

- **Scalable: store and process petabytes, scale by adding HW**
 - horizontal scaling
- **Reliable: data is replicated, failed tasks are rerun**
- **Used by many Enterprises**
 - Yahoo is the biggest contributor and user
 - Amazon, Facebook, IBM, lots of smaller companies
 - Many Applications run on 20-100 nodes, some run on thousands (<http://wiki.apache.org/hadoop/PoweredBy>)
- **Written in Java**
 - But users can write code in other languages



Summary, so far

- **Hadoop: framework for running apps on many machines.**
 - Can scale to petabytes of data, thousands of machines
- **Good for data processing, highly parallel apps**
- **Scales very well, reliable**
- **Used by many companies**

Hadoop components

Hadoop components

- **Hadoop Distributed File System (HDFS)**
 - scalable, fault-tolerant file system
- **Distributed Processing Framework**
 - Using Map-Reduce metaphor
- **Other systems built around Hadoop:**
 - PIG: high-level language for data analysis
 - HBase: Storage for semi-structured data



HDFS Goals

HDFS Goals

- **Highly fault-tolerant**
 - runs on commodity HW, which can fail frequently
- **High throughput of data access**
 - Streaming access to data

HDFS Goals

- **Highly fault-tolerant**
 - runs on commodity HW, which can fail frequently
- **High throughput of data access**
 - Streaming access to data
- **Large files**
 - Typical file is gigabytes to terabytes in size
 - Support for tens of millions of files
- **Simple coherency**
 - Write-once-read-many access model

HDFS Architecture

HDFS Architecture

- **Master-slave architecture**
 - Single master: *namenode*
 - Many slaves: *datanodes*
- **Files are broken in to large blocks.**
 - Typically 128 MB
 - Replicated to several *datanodes*, for reliability
- **Client talks to both namenode and datanodes**

HDFS architecture

HDFS architecture

Namenode (the master)

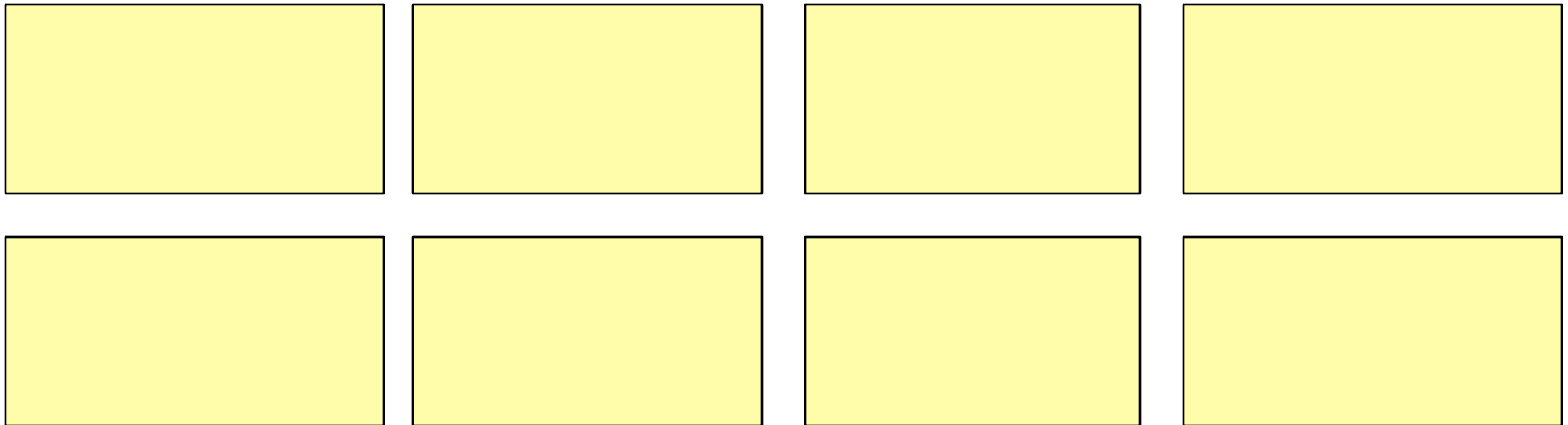
name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

Datanodes (the slaves)

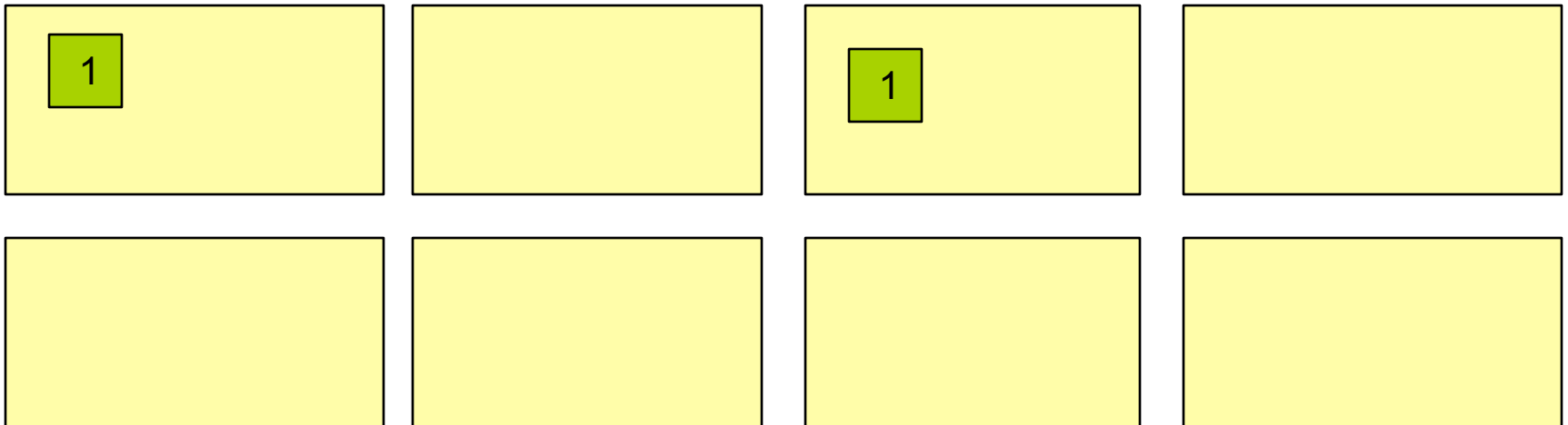


HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

Datanodes (the slaves)

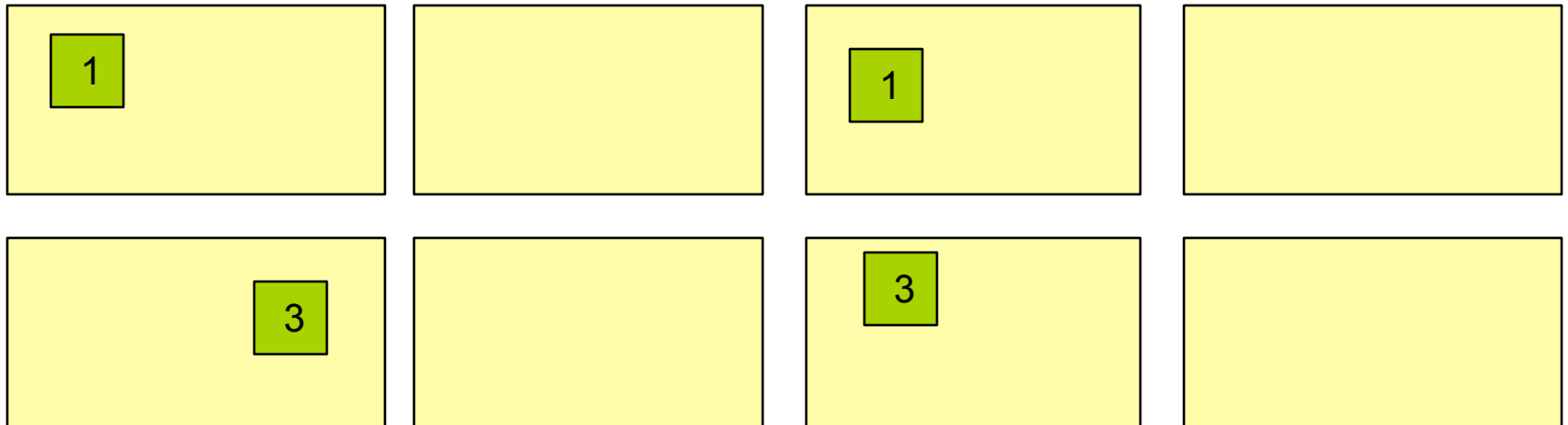


HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

Datanodes (the slaves)

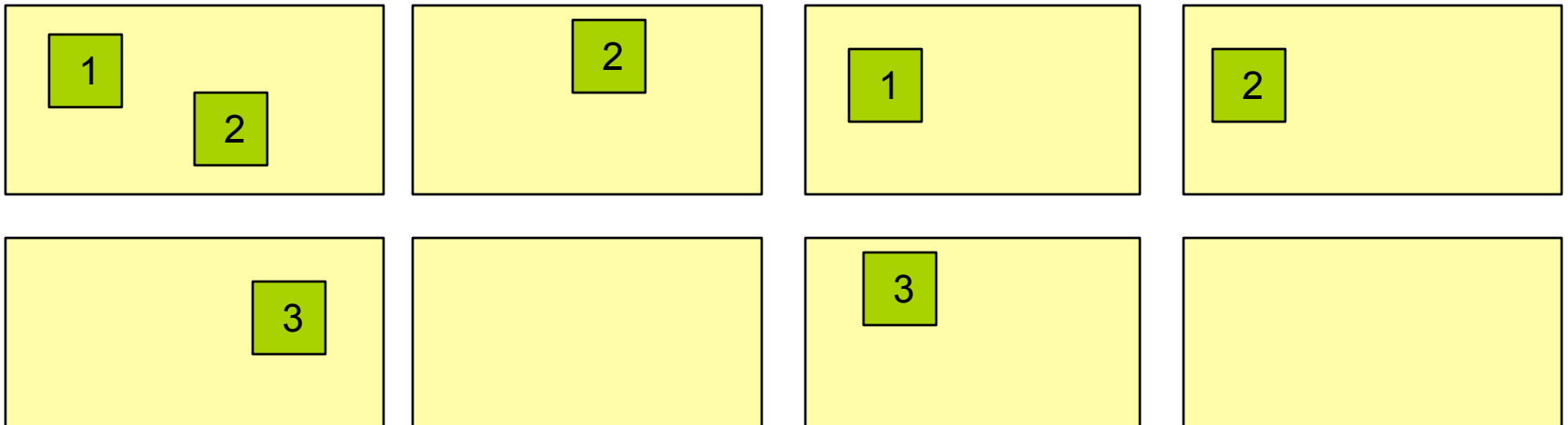


HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

Datanodes (the slaves)

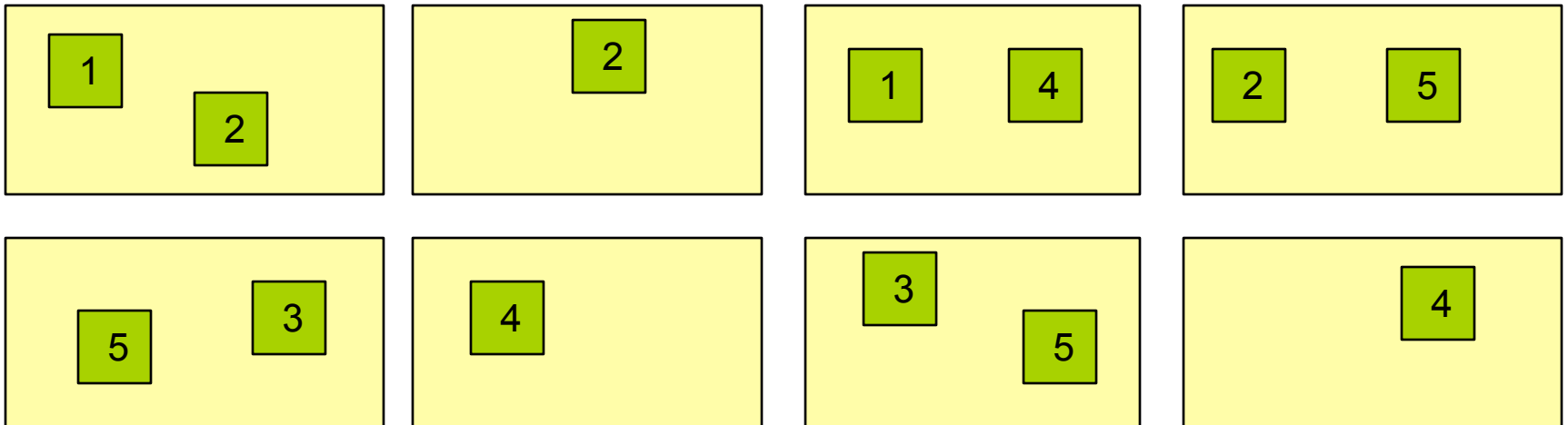


HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.zip, copies:3, blocks:{2,4,5}

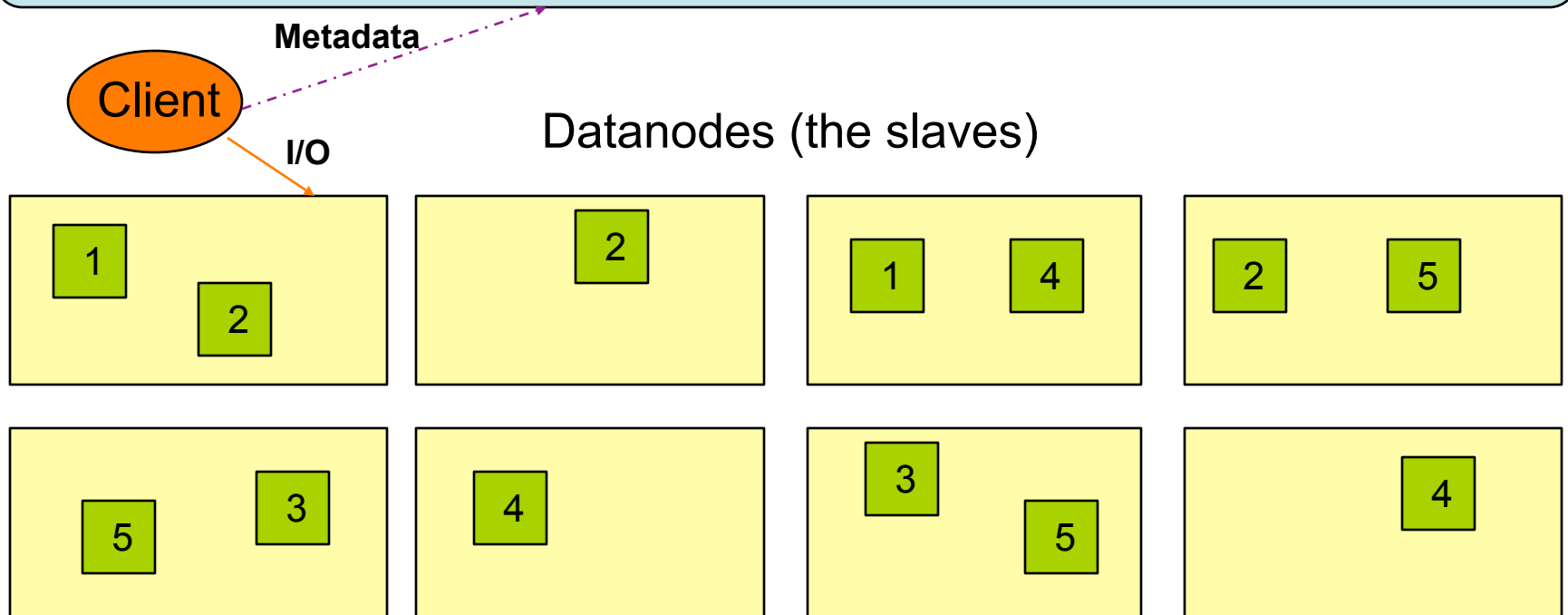
Datanodes (the slaves)



HDFS architecture

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.zip, copies:3, blocks:{2,4,5}



Summary, so far

- **HDFS: Hadoop Distributed File System**
- **Built to handle large files.**
- **Fault tolerant (replication)**
- **Master-slave architecture**

Distributed processing using Map-Reduce

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$
 - Reduce: $(K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)$

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$
 - Reduce: $(K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)$
- **User writes Map and Reduce functions. Hadoop takes care of the rest:**
 - Partitions job into lots of *tasks*

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$
 - Reduce: $(K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)$
- **User writes Map and Reduce functions. Hadoop takes care of the rest:**
 - Partitions job into lots of *tasks*
 - Schedules tasks on nodes close to data

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$
 - Reduce: $(K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)$
- **User writes Map and Reduce functions. Hadoop takes care of the rest:**
 - Partitions job into lots of *tasks*
 - Schedules tasks on nodes close to data
 - Monitors tasks

Distributed processing using Map-Reduce

- **Map-Reduce is the programming model used in Hadoop**
- **General form:**
 - Map: $(K1, V1) \rightarrow \text{list}(K2, V2)$
 - Reduce: $(K2, \text{list}(V2)) \rightarrow \text{list}(K3, V3)$
- **User writes Map and Reduce functions. Hadoop takes care of the rest:**
 - Partitions job into lots of *tasks*
 - Schedules tasks on nodes close to data
 - Monitors tasks
 - Kills and restarts if they fail/hang/disappear

Map-Reduce example: Logical flow

Map-Reduce example: Logical flow

- **Ex: grep for # of occurrences of 'cow' and 'dog'**

Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'

input

```
dog  
cow  
bee  
cow
```

Map-Reduce example: Logical flow

- **Ex: grep for # of occurrences of 'cow' and 'dog'**

input

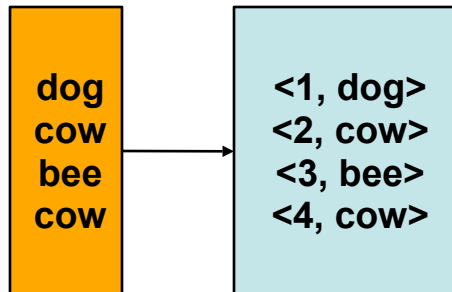
```
dog
cow
bee
cow
```

cat *

Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'

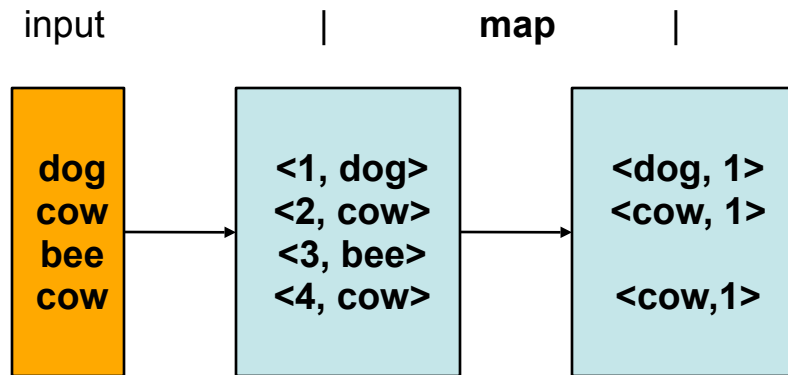
input



cat *

Map-Reduce example: Logical flow

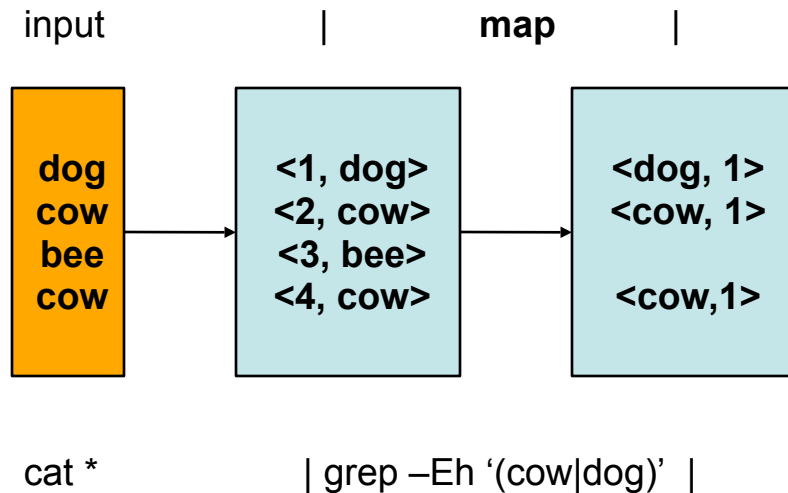
- Ex: grep for # of occurrences of 'cow' and 'dog'



cat *

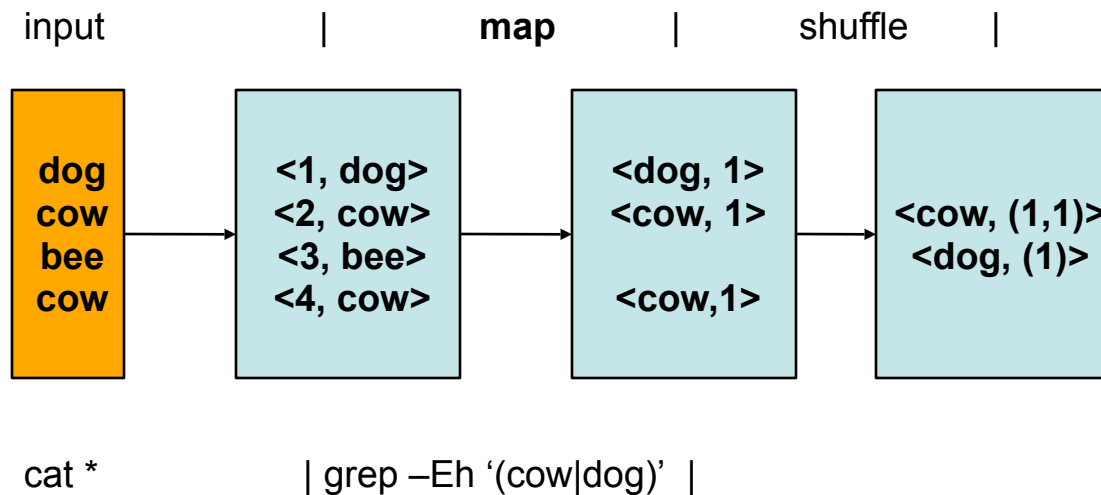
Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'



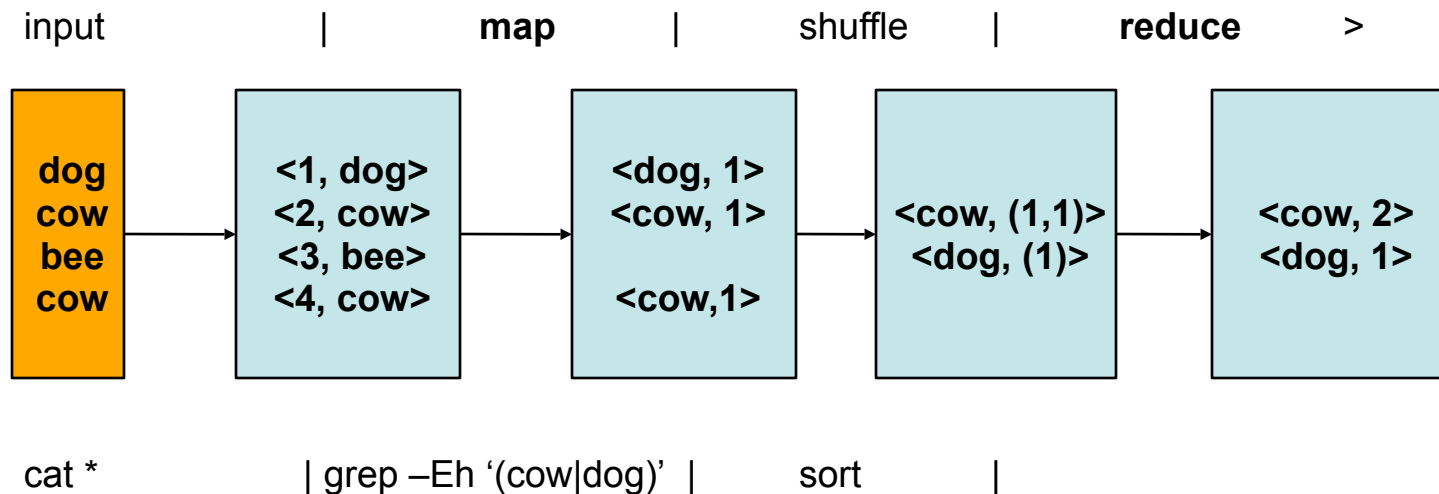
Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'



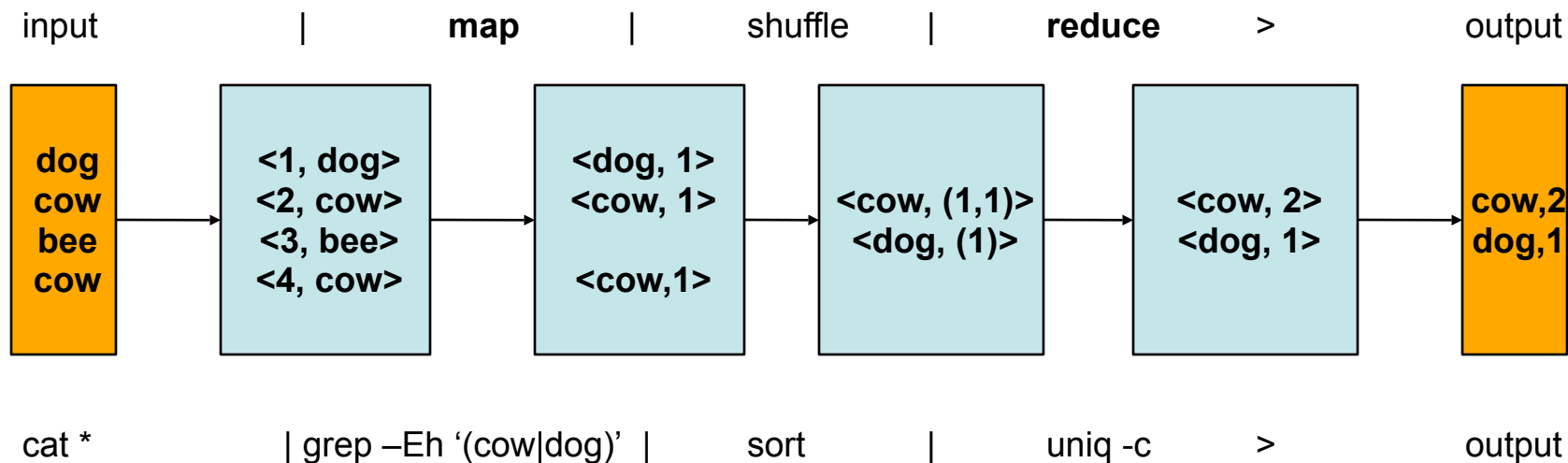
Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'



Map-Reduce example: Logical flow

- Ex: grep for # of occurrences of 'cow' and 'dog'



Physical flow

Physical flow

input
HDFS

split 0

split 1

split 2

split 3

split 4

Physical flow

input
HDFS

split 0

split 1

split 2

split 3

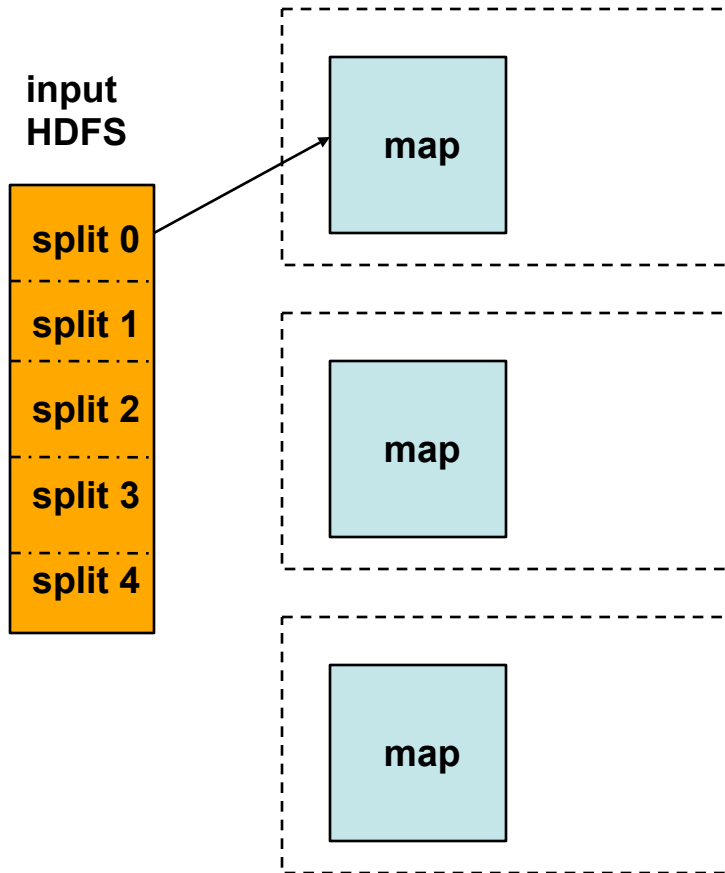
split 4

map

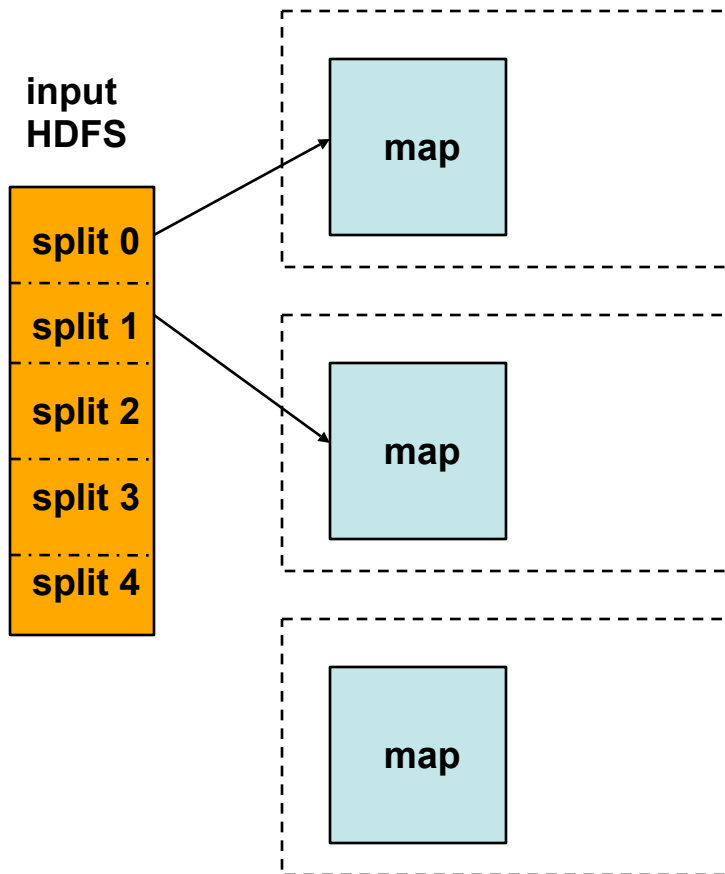
map

map

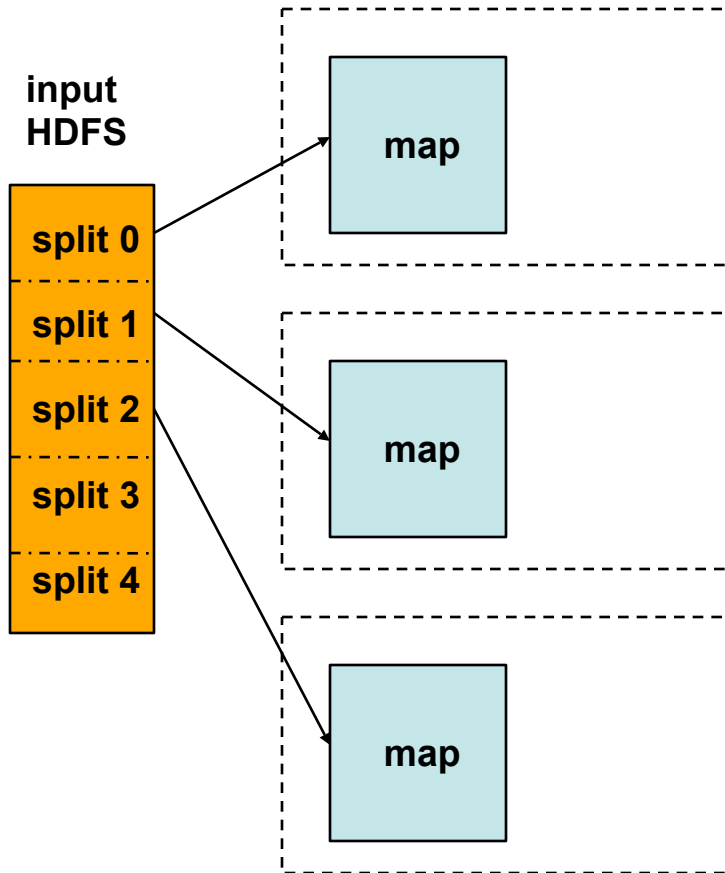
Physical flow



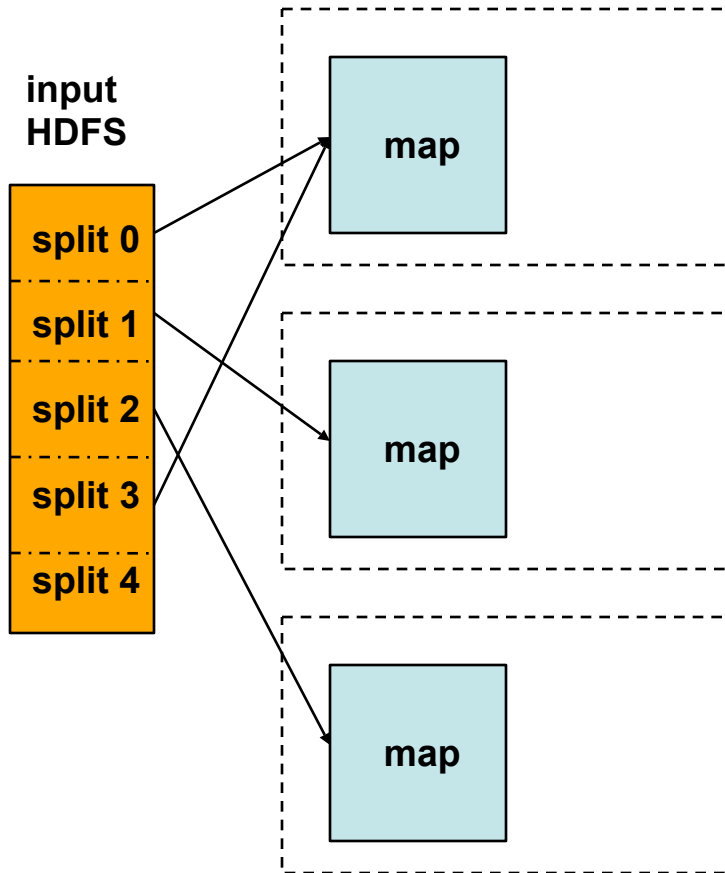
Physical flow



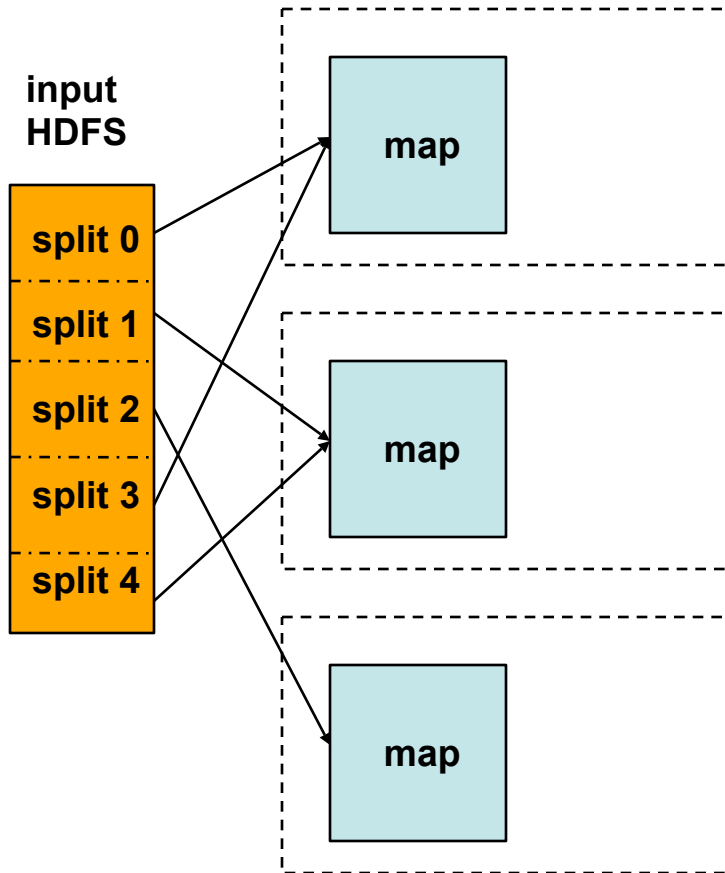
Physical flow



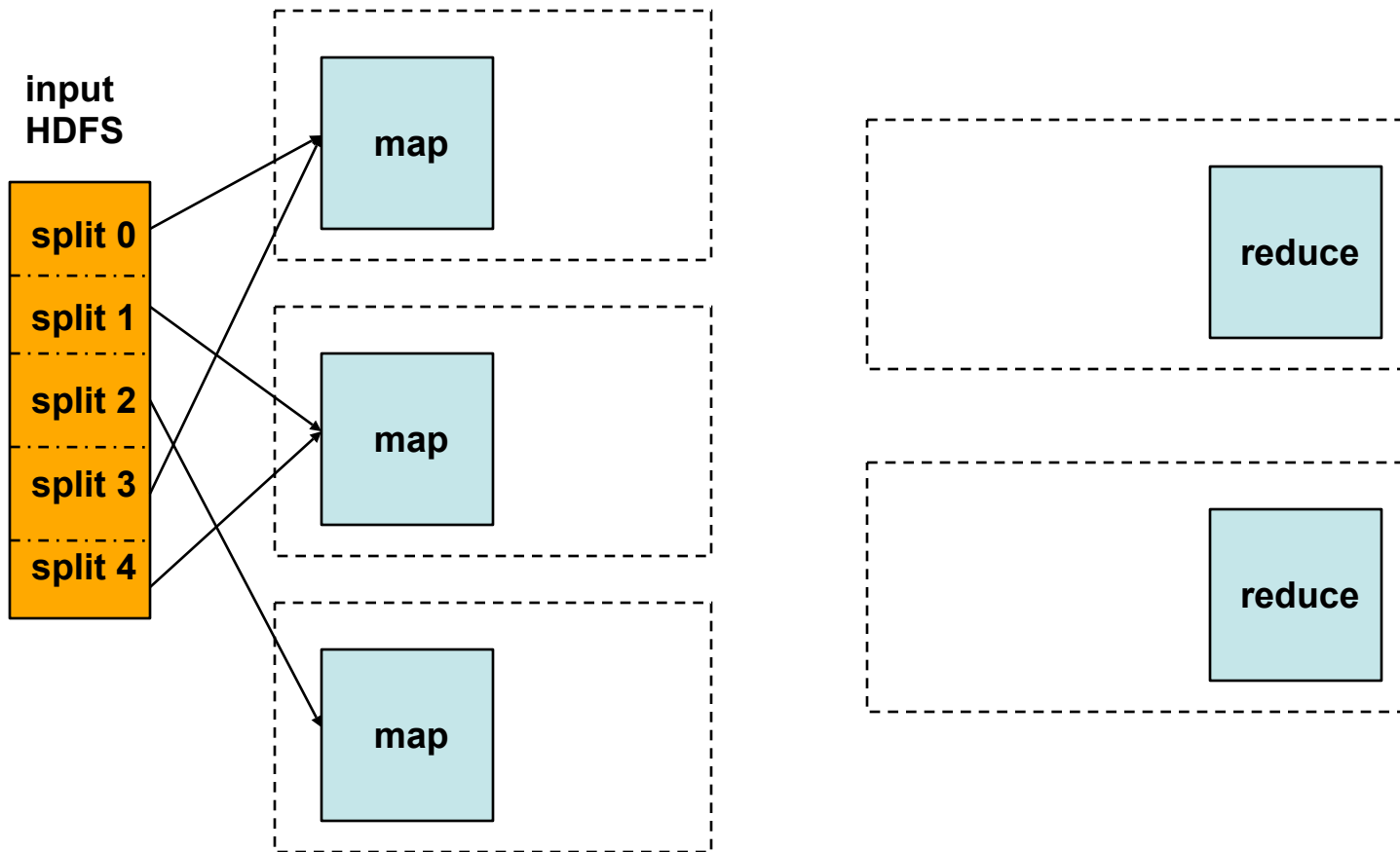
Physical flow



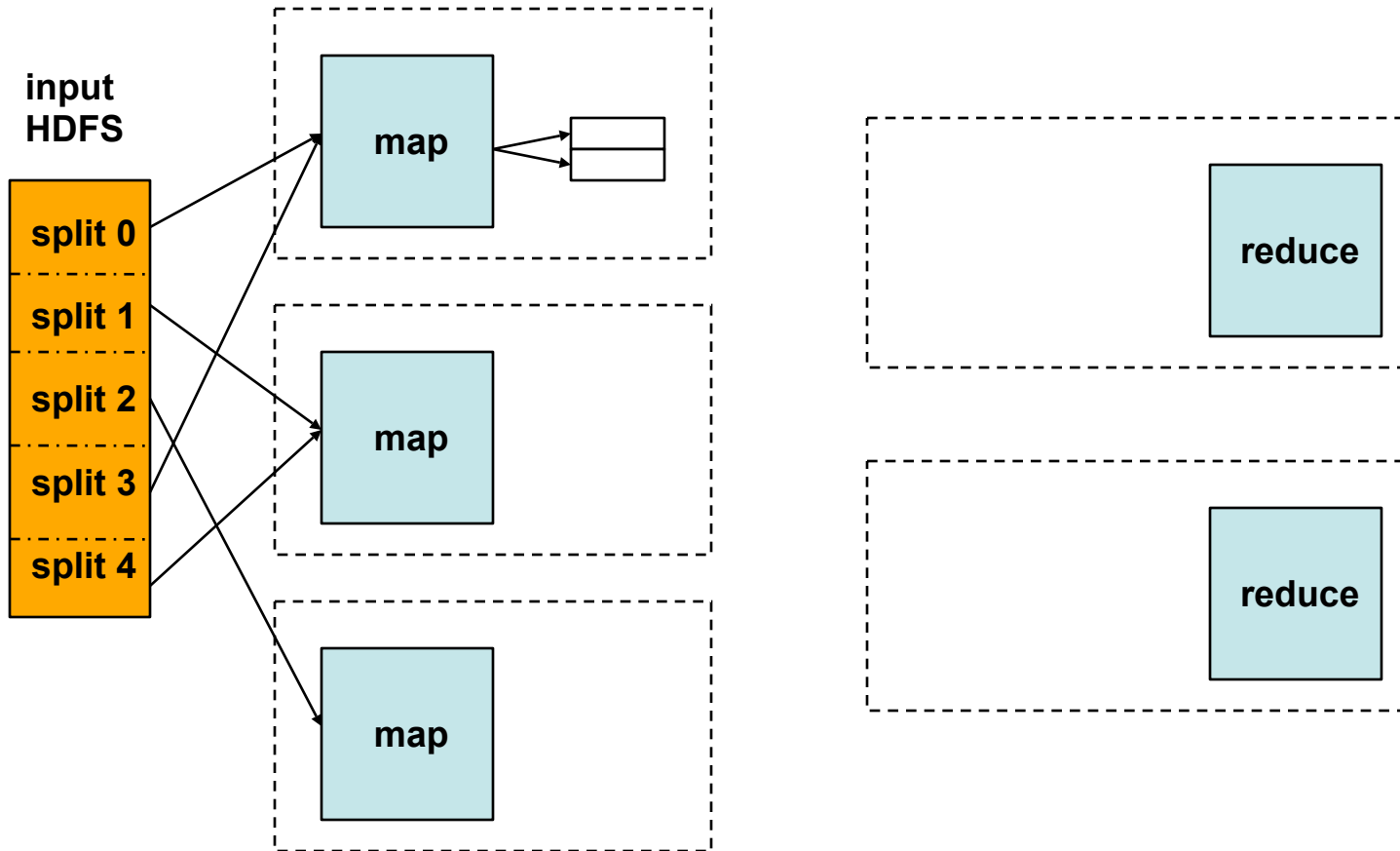
Physical flow



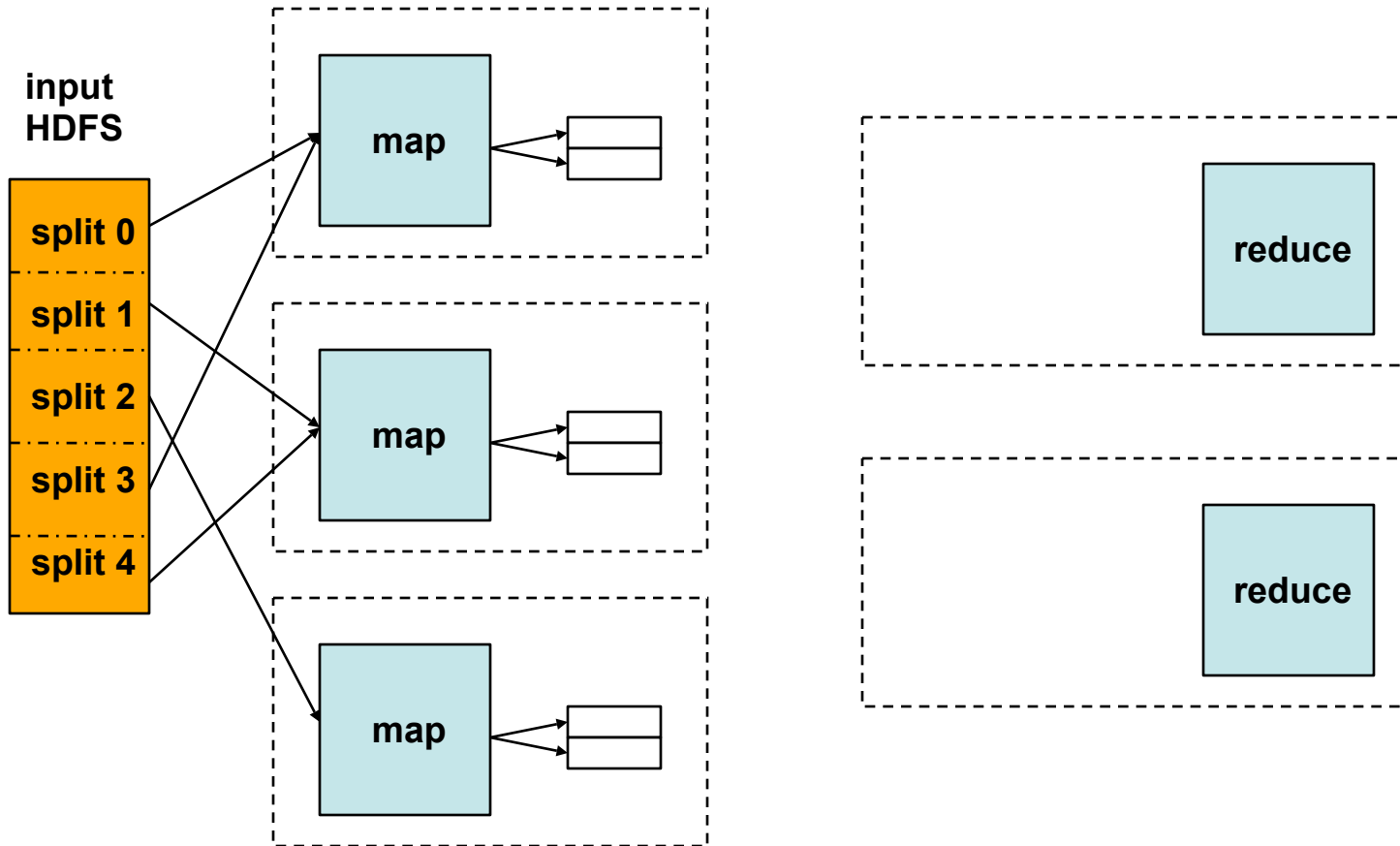
Physical flow



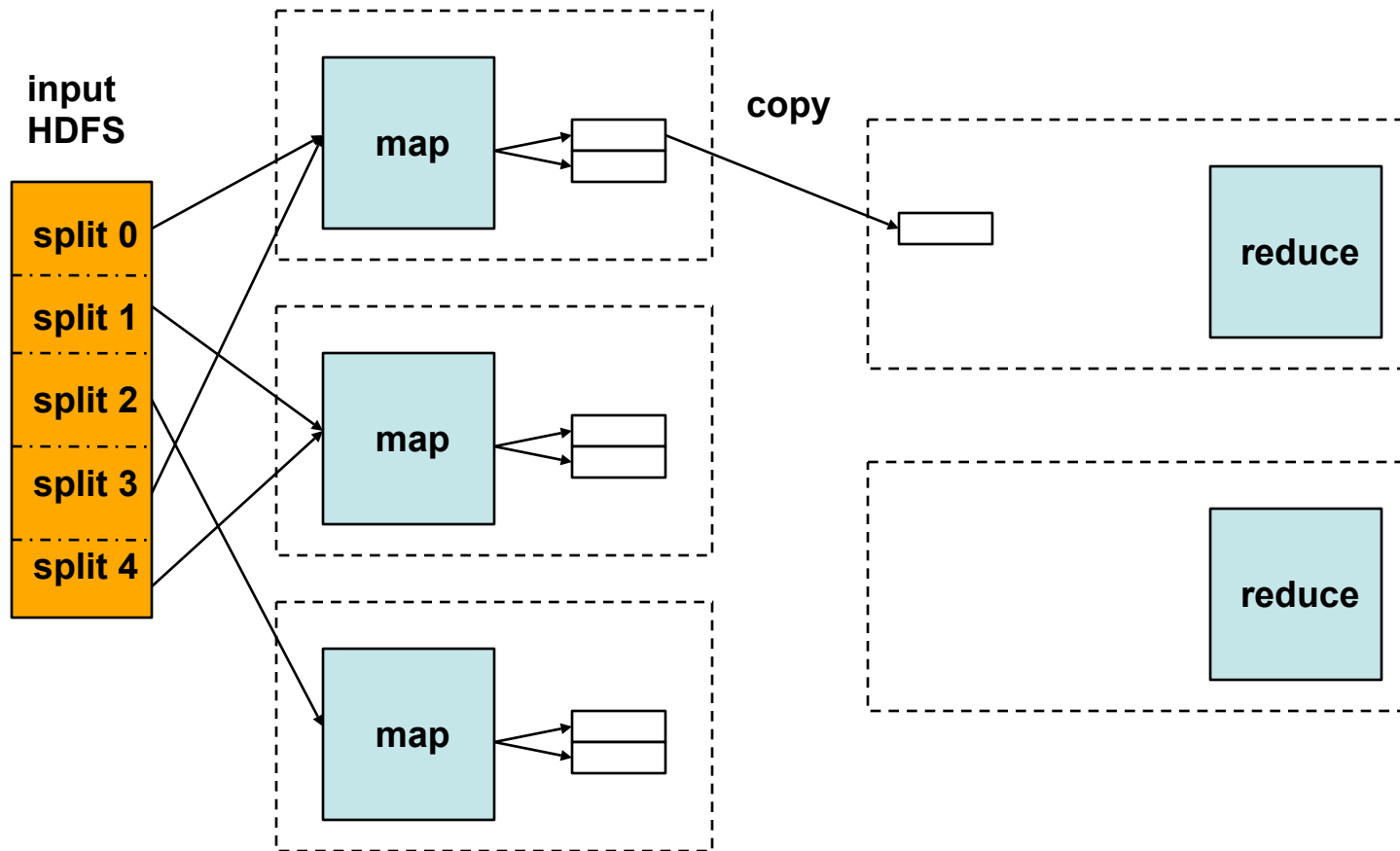
Physical flow



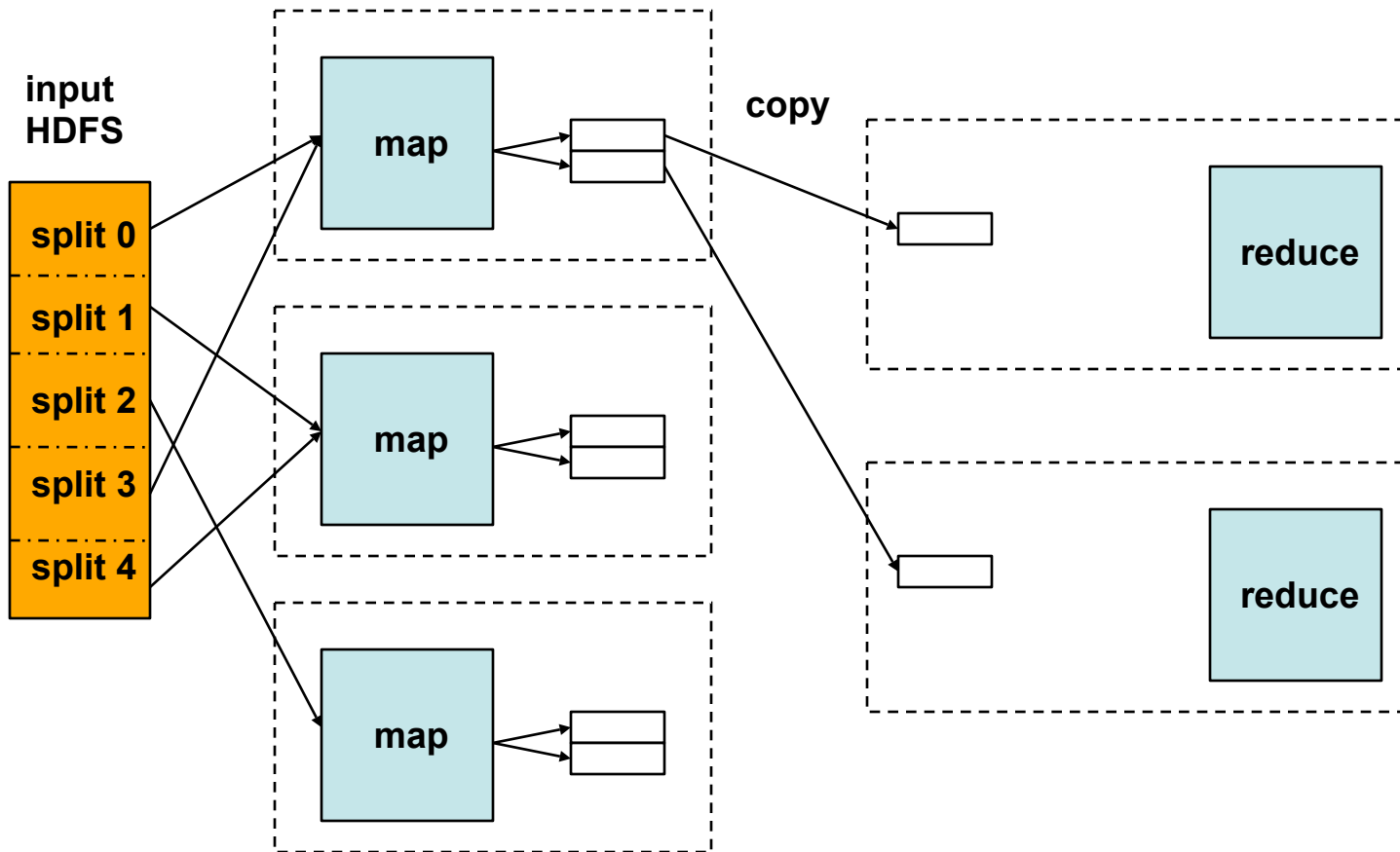
Physical flow



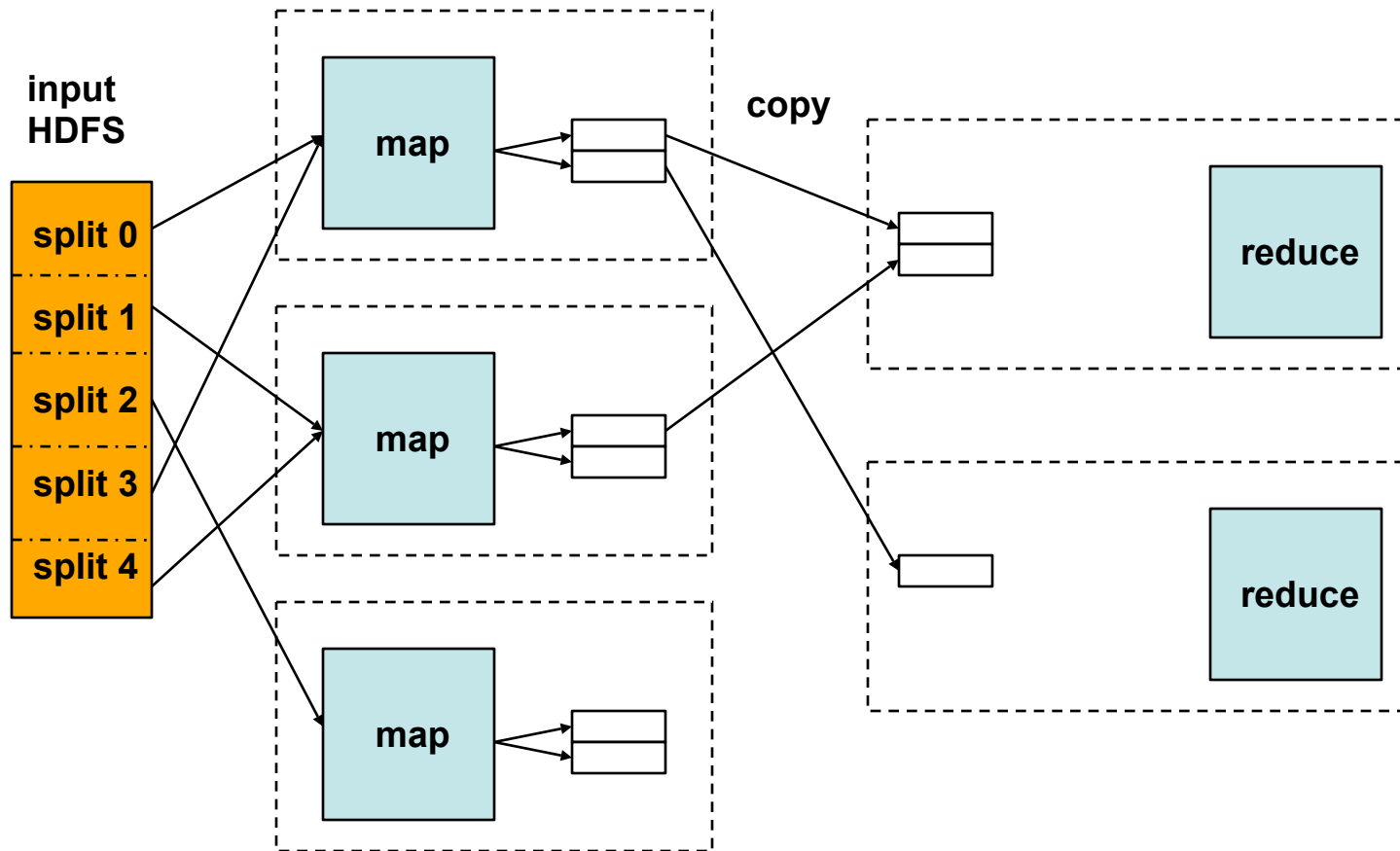
Physical flow



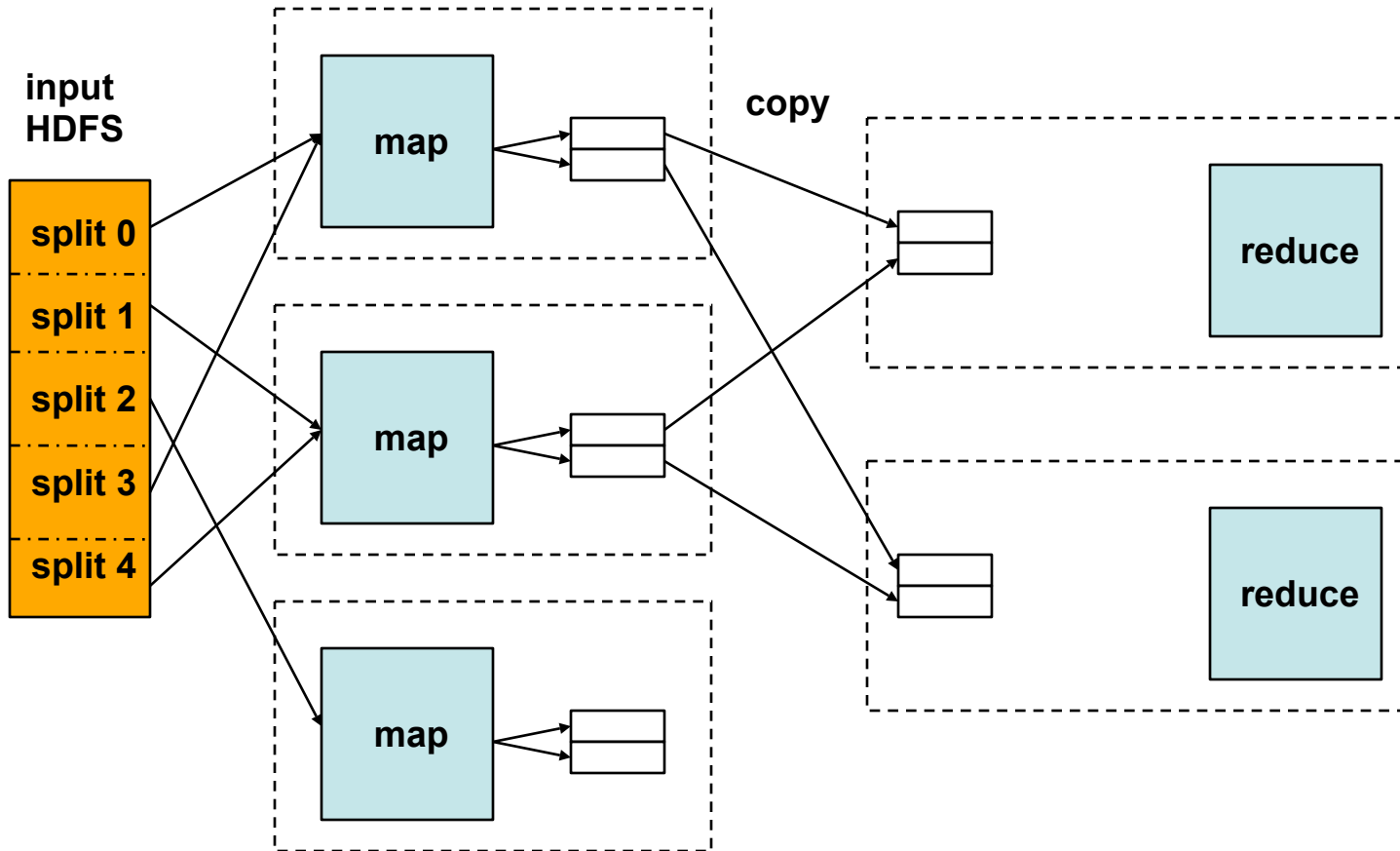
Physical flow



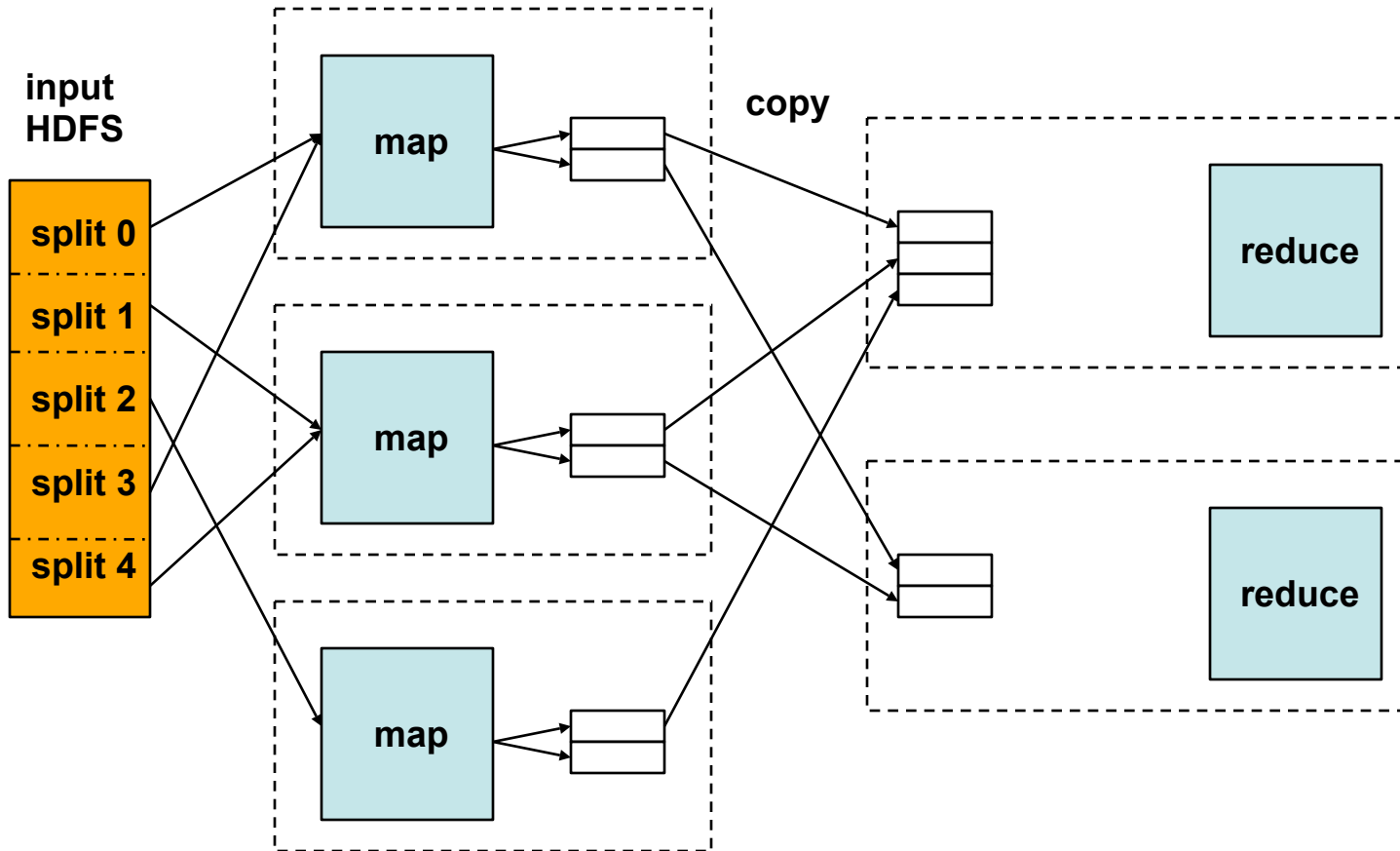
Physical flow



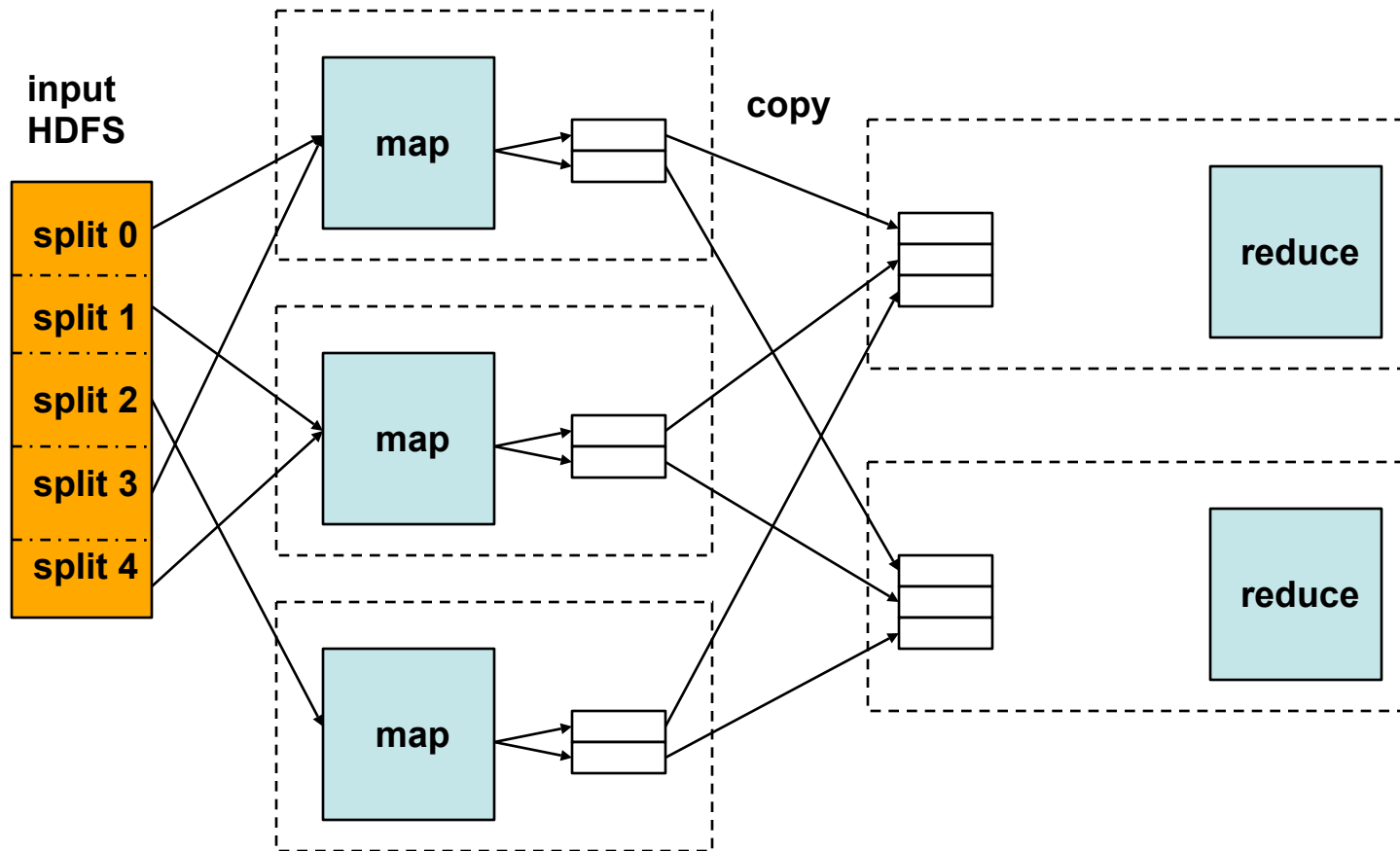
Physical flow



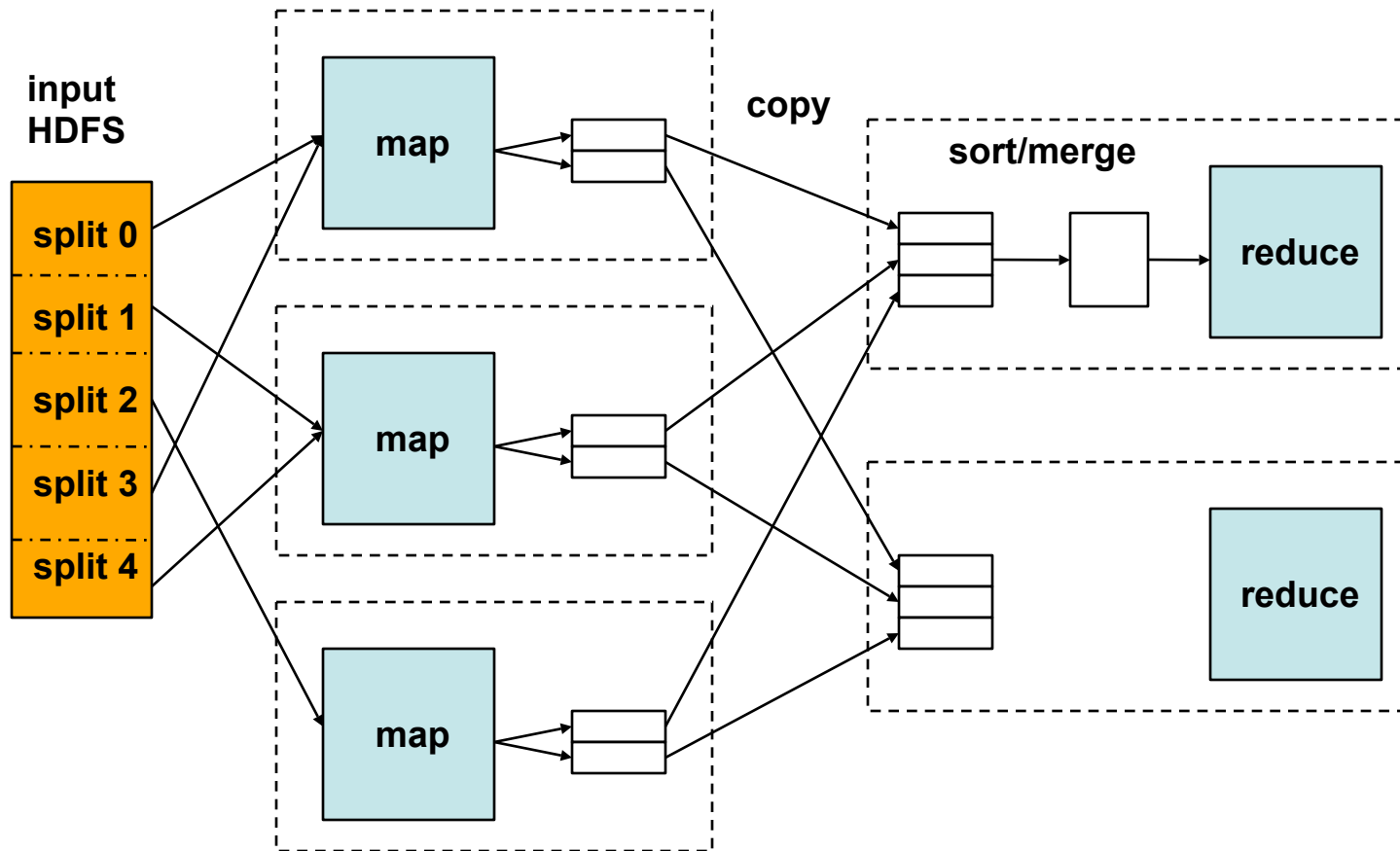
Physical flow



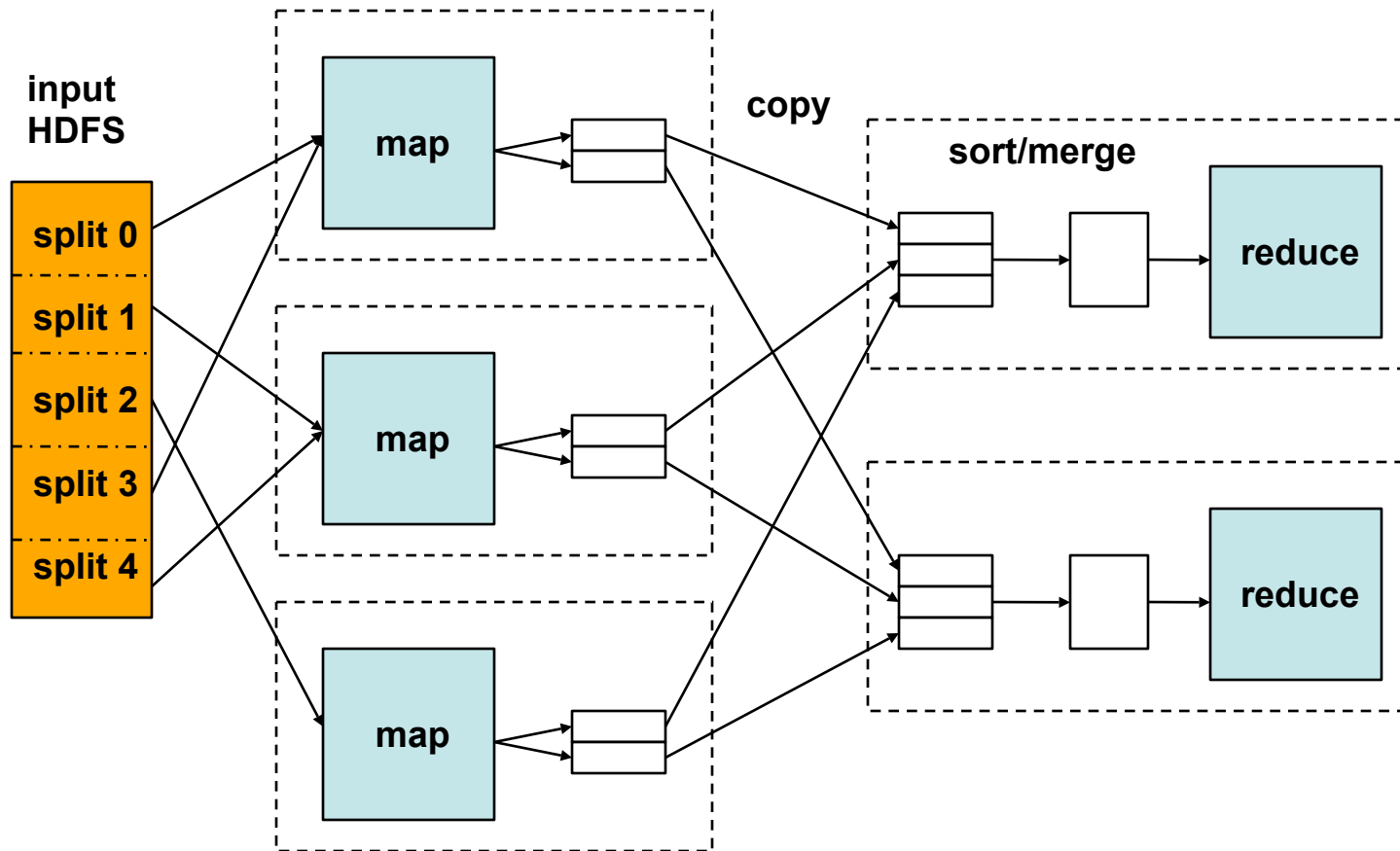
Physical flow



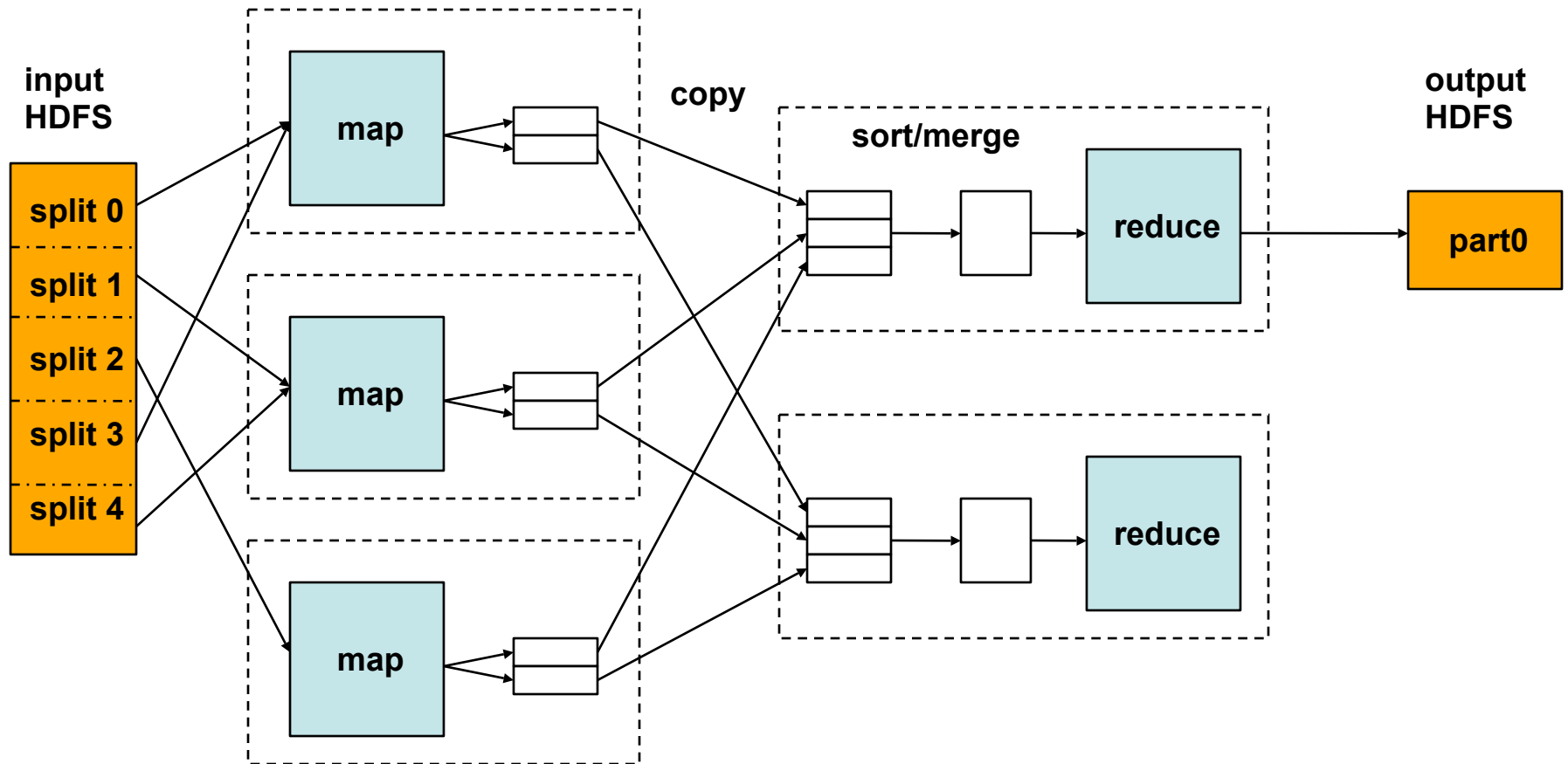
Physical flow



Physical flow



Physical flow



Why Map-Reduce?

- **May seem like an arbitrary model...**
- **generalizes common patterns**
 - for very large data collections
- **natural for:**
 - statistics, e.g., counting
 - offline databases (index creation, update)
 - extraction, refinement, exploration, etc.
- **permits optimizations**
 - scalable, distributed, data-locality, etc.

Map-Reduce architecture

Map-Reduce architecture

- **Master-Slave architecture**

Map-Reduce architecture

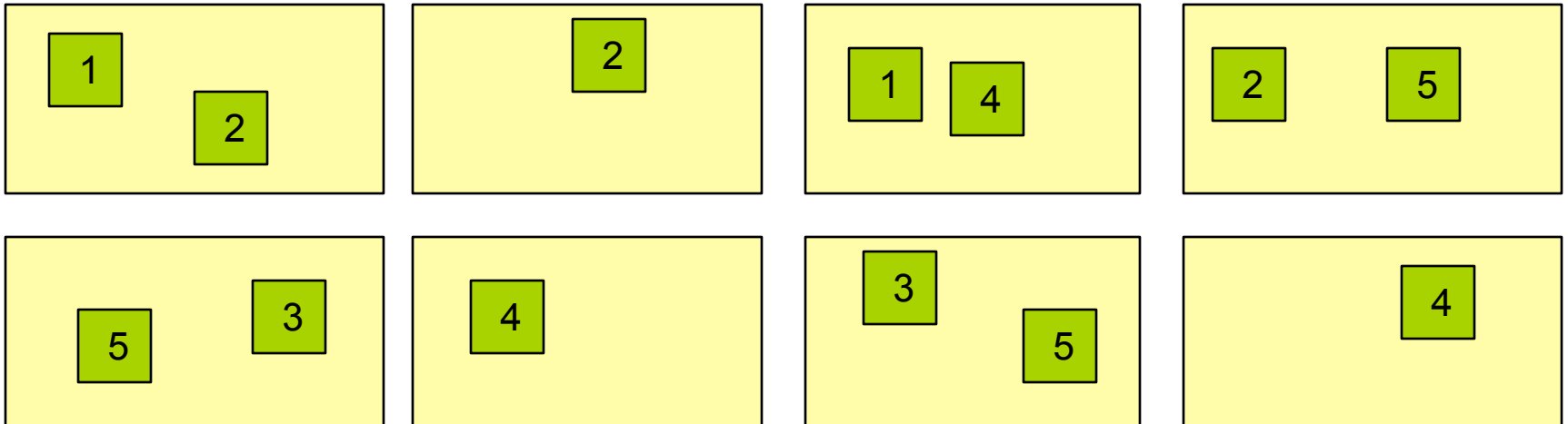
- **Master-Slave architecture**
- **Master: JobTracker**
 - Accepts MR jobs submitted by users
 - Assigns Map and Reduce tasks to TaskTrackers (slaves)
 - Monitors task and TaskTracker status, re-executes tasks upon failure
- **Slaves: TaskTrackers**
 - Run Map and Reduce tasks upon instruction from the Jobtracker
 - Manage storage and transmission of intermediate output

Map-Reduce + HDFS

Map-Reduce + HDFS

Namenode

file1 (1,3)
file2 (2,4,5)



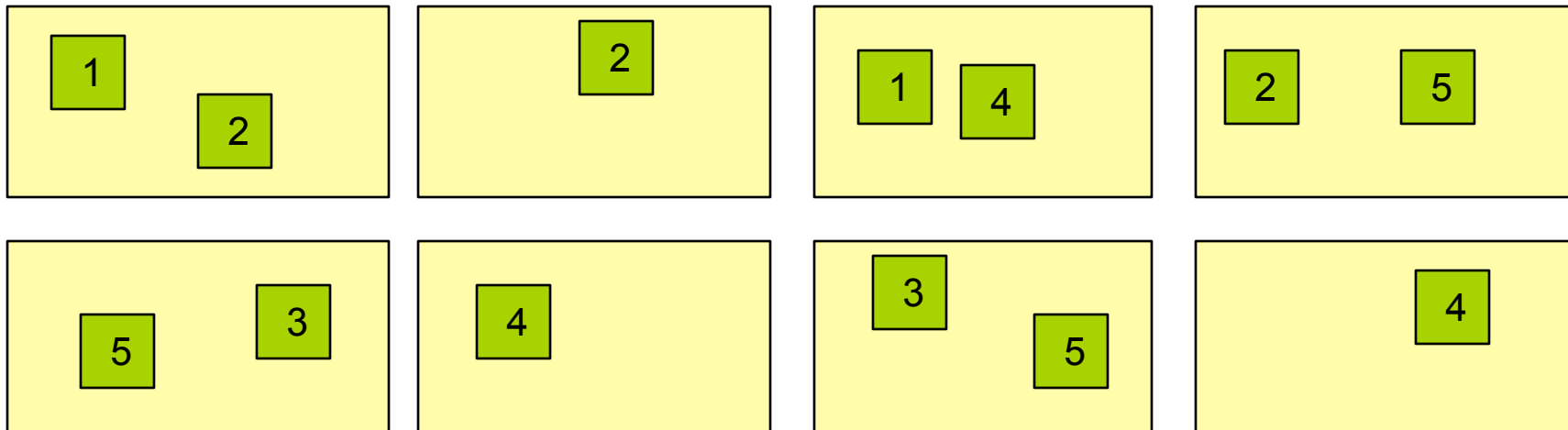
Map-Reduce + HDFS

Namenode

file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks



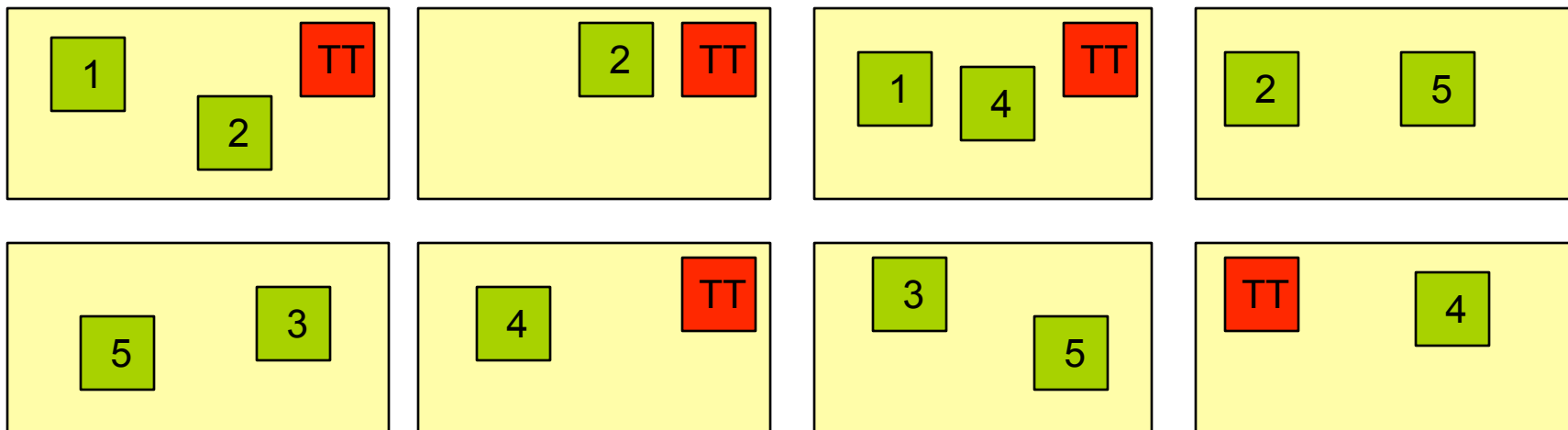
Map-Reduce + HDFS

Namenode

file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks



Map-Reduce + HDFS

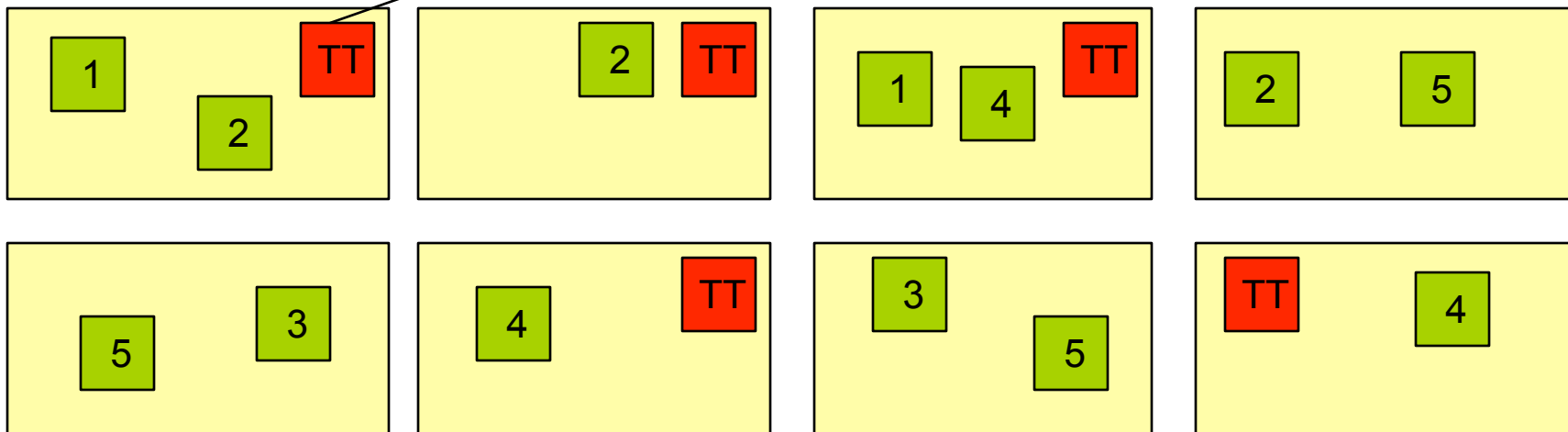
Namenode

file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks

ask for task



Map-Reduce + HDFS

Namenode

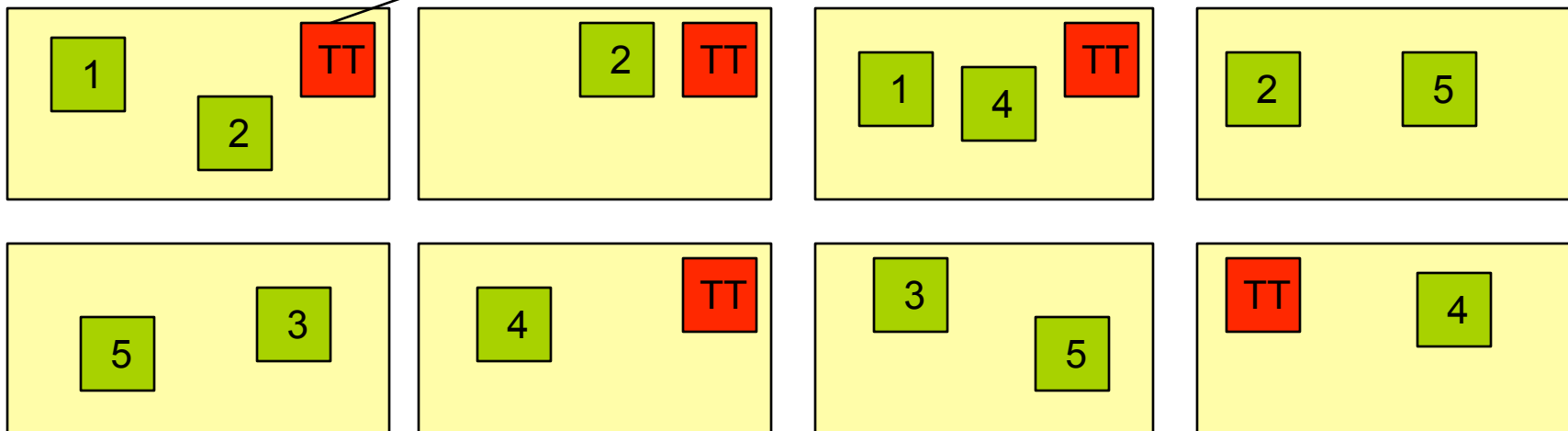
file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks

ask for task

Block 1



Map-Reduce + HDFS

Namenode

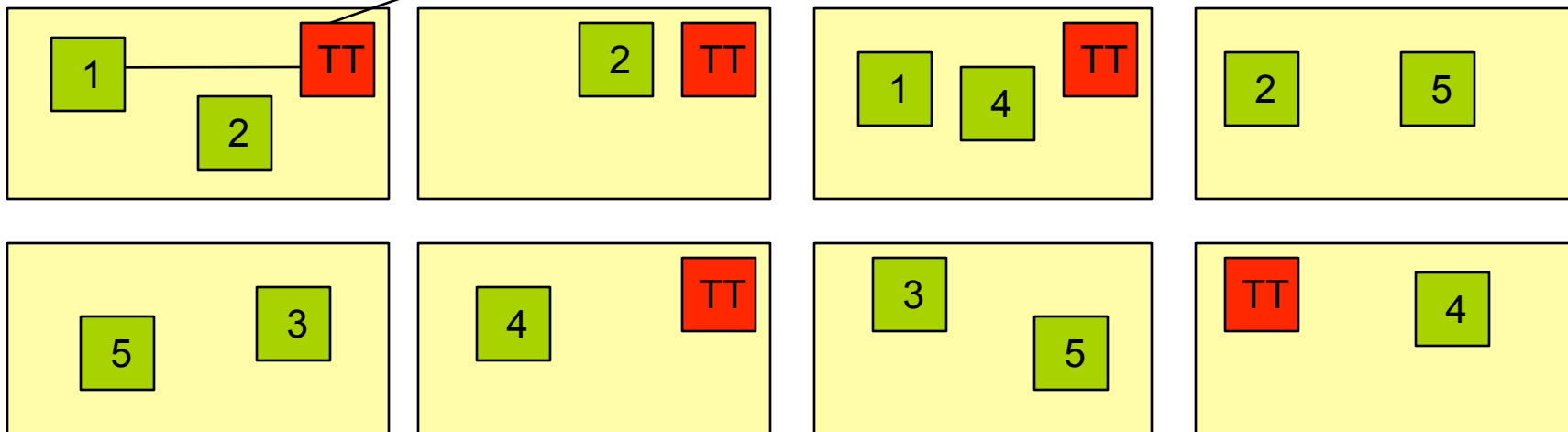
file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks

ask for task

Block 1



Map-Reduce + HDFS

Namenode

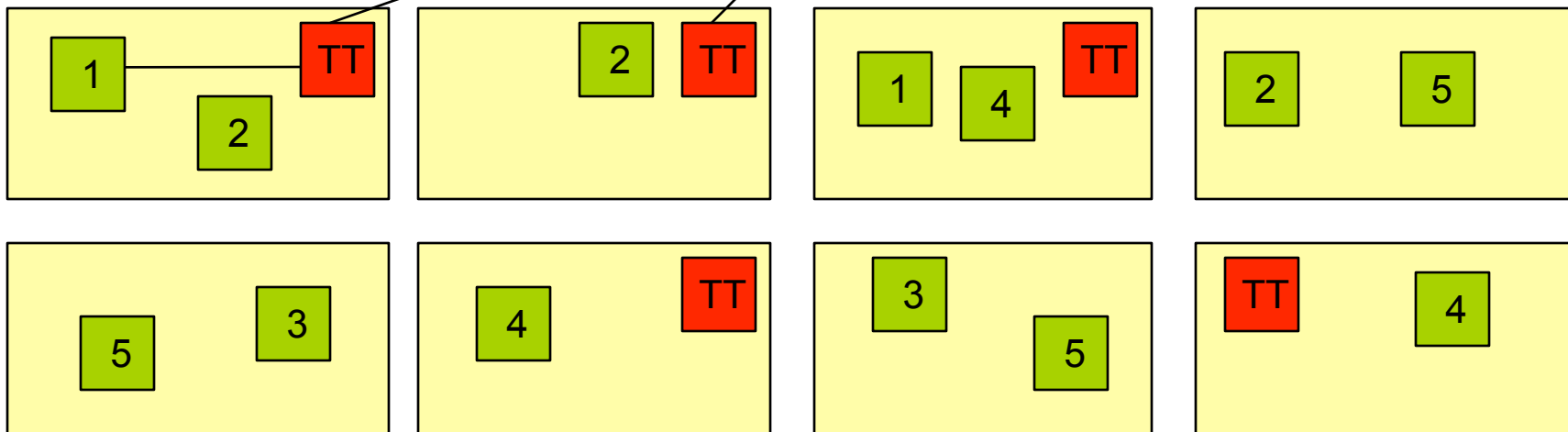
file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks

ask for task

Block 1



Map-Reduce + HDFS

Namenode

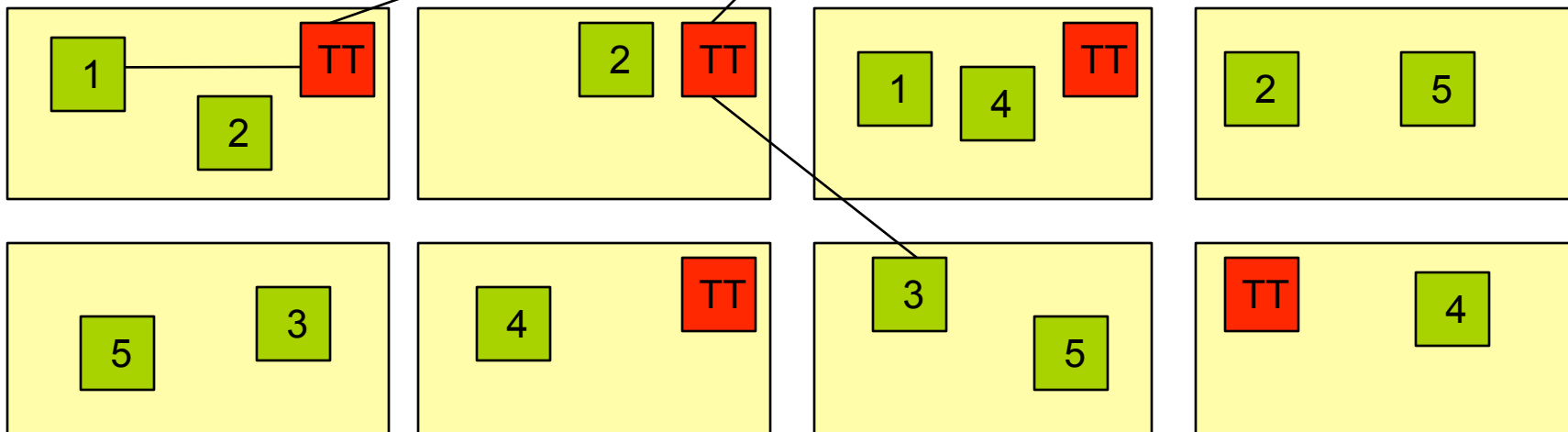
file1 (1,3)
file2 (2,4,5)

JobTracker

Map tasks
Reduce tasks

ask for task

Block 1



Summary, so far

- **Map-Reduce: programming model for distributed processing**
- **Deals with data as keys & values**
- **Apply a map function to each key-value pair, then apply a reduce function to all values for a key**
- **Natural model for lots of data processing tasks**

Yahoo's Hadoop clusters

Yahoo's Hadoop clusters



Yahoo's Hadoop clusters

- **10,000+ machines (and growing fast) running Hadoop**



Yahoo's Hadoop clusters

- **10,000+ machines (and growing fast) running Hadoop**
- **Largest cluster is currently 2000 nodes (scale is increasing)**



Yahoo's Hadoop clusters

- **10,000+ machines (and growing fast) running Hadoop**
- **Largest cluster is currently 2000 nodes (scale is increasing)**
- **Few petabytes of user data (compressed, unreplicated)**



Yahoo's Hadoop clusters

- **10,000+ machines (and growing fast) running Hadoop**
- **Largest cluster is currently 2000 nodes (scale is increasing)**
- **Few petabytes of user data (compressed, unreplicated)**
- **~10,000 research jobs / week**



Hadoop's appeal

Hadoop's appeal

- **For users:**

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.
 - Can run on a single node – easy to install, configure, run

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.
 - Can run on a single node – easy to install, configure, run
 - Part of University coursework for distributed computing

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.
 - Can run on a single node – easy to install, configure, run
 - Part of University coursework for distributed computing
- **For developers:**

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.
 - Can run on a single node – easy to install, configure, run
 - Part of University coursework for distributed computing
- **For developers:**
 - Open Source. Contribute to development, testing, documentation, performance.

Hadoop's appeal

- **For users:**
 - Use Hadoop for large-scale data crunching to analyze logs, study behavior patterns, data mining, etc.
 - Runs on Amazon's EC2/S3. Combination used by several startups.
 - Can run on a single node – easy to install, configure, run
 - Part of University coursework for distributed computing
- **For developers:**
 - Open Source. Contribute to development, testing, documentation, performance.
 - Be part of something big

Engineering challenges

Engineering challenges

- **Scaling to more than 2000 nodes**

Engineering challenges

- **Scaling to more than 2000 nodes**
- **Performance**
 - how to make use of multi-CPU machines
 - optimize for CPU, IO, network bandwidth
 - how to handle lots of communication
- **Provisioning (scheduling)**
 - how to run many jobs simultaneously on a Grid
 - which machines do you give to which job?
- **Easier programming models than Map-Reduce**
- **Apps over Hadoop**

Engineering challenges

- **Scaling to more than 2000 nodes**
- **Performance**
 - how to make use of multi-CPU machines
 - optimize for CPU, IO, network bandwidth
 - how to handle lots of communication
- **Provisioning (scheduling)**
 - how to run many jobs simultaneously on a Grid
 - which machines do you give to which job?
- **Easier programming models than Map-Reduce**
- **Apps over Hadoop**
- **You can help!**

Thank you

- Hadoop: <http://hadoop.apache.org/>
- Hadoop Wiki (for developers): <http://wiki.apache.org/hadoop/>
- Yahoo's Hadoop blog: <http://developer.yahoo.com/blogs/hadoop/>
- My email: vivekr@yahoo-inc.com
- **Disclaimer:** This is an event specific public presentation and the Q&A session is to focus on the content delivered by the speaker. Please contact the Yahoo! PR team or pr-india@yahoo-inc.com for any questions that you may have outside the purview of this presentation.

Backup slides

Q&A (win prizes)

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**
 - 2 yrs

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**
 - 2 yrs
- **How big is the largest Hadoop production installation?**
 - a. 1,000 processors
 - b. 10,000 processors
 - c. 100,00 processors

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**
 - 2 yrs
- **How big is the largest Hadoop production installation?**
 - a. 1,000 processors
 - b. 10,000 processors
 - c. 100,00 processors
 - (B) – 10,00 processors

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**
 - 2 yrs
- **How big is the largest Hadoop production installation?**
 - a. 1,000 processors
 - b. 10,000 processors
 - c. 100,00 processors
 - (B) – 10,00 processors
- **Why is Hadoop called 'Hadoop'?**

Q&A (win prizes)

- **How long has Yahoo been working with Hadoop?**
 - 2 yrs
- **How big is the largest Hadoop production installation?**
 - a. 1,000 processors
 - b. 10,000 processors
 - c. 100,00 processors
 - (B) – 10,00 processors
- **Why is Hadoop called 'Hadoop'?**
- **Which organization hosts the Hadoop project (source code, documentation)?**

Y! Search Webmap

- <http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html>
- **Starts with every Web page crawled by Y!, produces a DB of all known pages & sites.**
- **World's Largest Hadoop Production Application**
 - Number of links between pages in the index: roughly 1 trillion links
 - Size of output: over 300 TB, compressed!
 - Number of cores used to run a single Map-Reduce job: over 10,000
 - Raw disk used in the production cluster: over 5 Petabytes

M45 Supercomputing Project

- <http://research.yahoo.com/node/1884>
- 4K processors, one of 50 most powerful systems
- 3TB of mem, 1.5PB storage
- Currently used by CMU.
- Open to other Universities.

Hadoop and CRL

- <http://biz.yahoo.com/bw/080324/20080324005325.html?.v=1>
- **Yahoo! and Computational Research Laboratories to Collaborate on Cloud Computing Research**
- **CRL runs one of the world's top-5 supercomputers (EKA)**
- **14,400 processors, 28TB mem, 140TB disk.**

Hadoop history

- <http://developer.yahoo.net/blog/archives/2007/07/yahoo-hadoop.html>
- 2004 - Initial versions of what is now Hadoop Distributed File System and Map-Reduce implemented by Doug Cutting & Mike Cafarella
- December 2005 - Nutch ported to the new framework. Hadoop runs reliably on 20 nodes.
- January 2006 - [Doug Cutting joins Yahoo!](#)
- February 2006 - Apache Hadoop project official started to support the standalone development of Map-Reduce and HDFS.
- March 2006 - Formation of the Yahoo! Hadoop team
- May 2006 - Yahoo sets up a Hadoop research cluster - 300 nodes
- April 2006 - Sort benchmark run on 188 nodes in 47.9 hours
- May 2006 - Sort benchmark run on 500 nodes in 42 hours (better hardware than April benchmark)
- October 2006 - Research cluster reaches 600 Nodes
- December 2006 - Sort times 20 nodes in 1.8 hrs, 100 nodes in 3.3 hrs, 500 nodes in 5.2 hrs, 900 nodes in 7.8
- January 2006 - Research cluster reaches 900 node
- April 2007 - Research clusters - 2 clusters of 1000 nodes



Handling HDFS failures

Handling HDFS failures

- **Namenode failure**
 - A single point of failure.
 - Keeps track of operations through a transaction log.
 - Memory image stored as a file.
 - At startup, reads memory image, applies transaction log, rewrites memory image, starts new transaction log
- **Datanode failures**
 - Namenode detects Datanode failures. Re-replicates blocks.
 - When Datanode comes up, it contacts Namenode
- **Checksums (CRC32) to validate data**

Block placement

- **Default is 3 replicas, but settable**
- **Blocks are placed:**
 - One on same rack
 - Others placed randomly across racks
- **Clients read from closest replica**
- **If the replication for a block drops below target, it is automatically re-replicated.**

Data correctness

- **Use checksums (CRC32) to validate data.**
- **File Creation**
 - Client computes checksum per 512 byte
 - DataNode stores the checksum
- **File access**
 - Client retrieves the data and checksum from DataNode
 - If Validation fails, Client tries other replicas

HDFS write-once policy

- If you have a 1TB database with 100 B records and you want to update 1% of them, how long will it take?
- Assume for argument's sake that you have a disk with 100MB/s max bandwidth, 10ms seek time, and 6000 rpm.
- The answer is surprising. If you seek for each update, then you need to do $n = 1\% \times 10^{12} B / 100 B = 10^8$ updates
- Each update will require (roughly) 10ms seek, 10 ms read (one rotation) and 10 ms write for a total of 30 ms.
- The total time for this strategy is $n \times 30 \text{ ms} = 3 \times 10^6$ seconds = 35 days.
- On the other hand, if you read the entire database sequentially and then rewrite it sequentially, it will take $2 \times 10^{12} B / (10^8 B/s) = 2 \times 10^4 \text{ s} = 5.6$ hours.