Overview

A very short Intro to Hadoop

photo by: exfordy, flickr

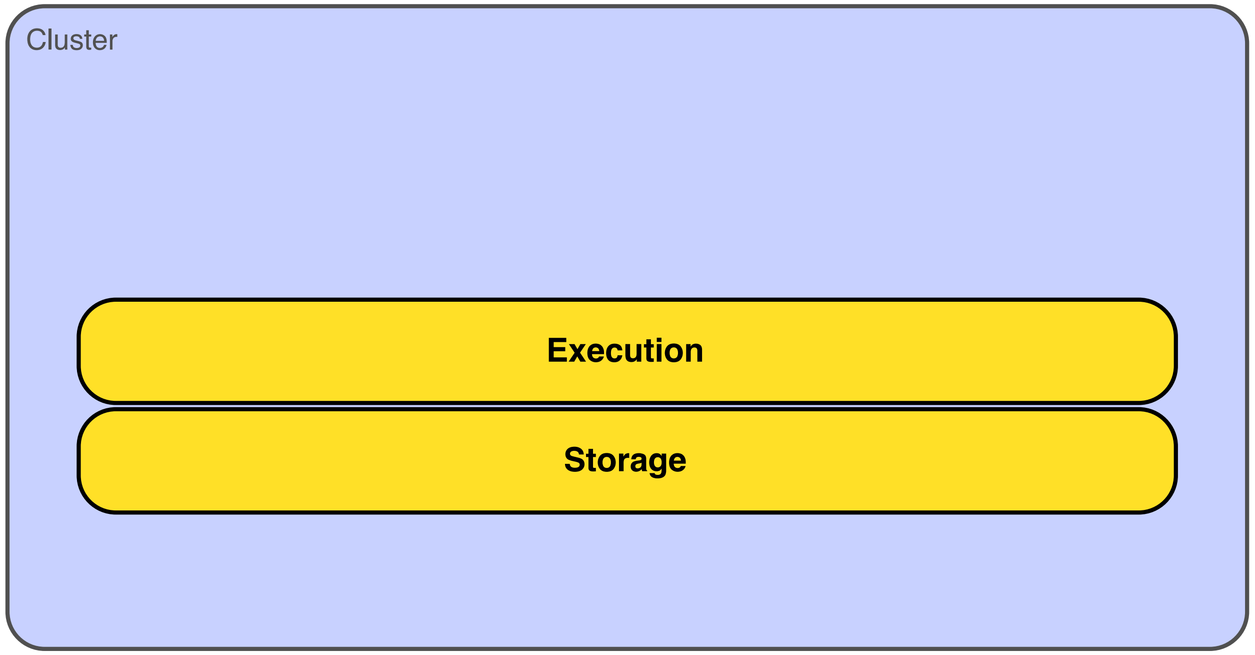
How to Crunch a Petabyte?

- Lots of disks, spinning all the time
- Redundancy, since disks die
- Lots of CPU cores, working all the time
- Retry, since network errors happen

Hadoop to the Rescue

- Scalable - many servers with lots of cores and spindles
- Reliable - detect failures, redundant storage
- Fault-tolerant - auto-retry, self-healing
- Simple - use many servers as one really big computer

Logical Architecture



- Logically, Hadoop is simply a computing cluster that provides:
  - a Storage layer, and
  - an Execution layer

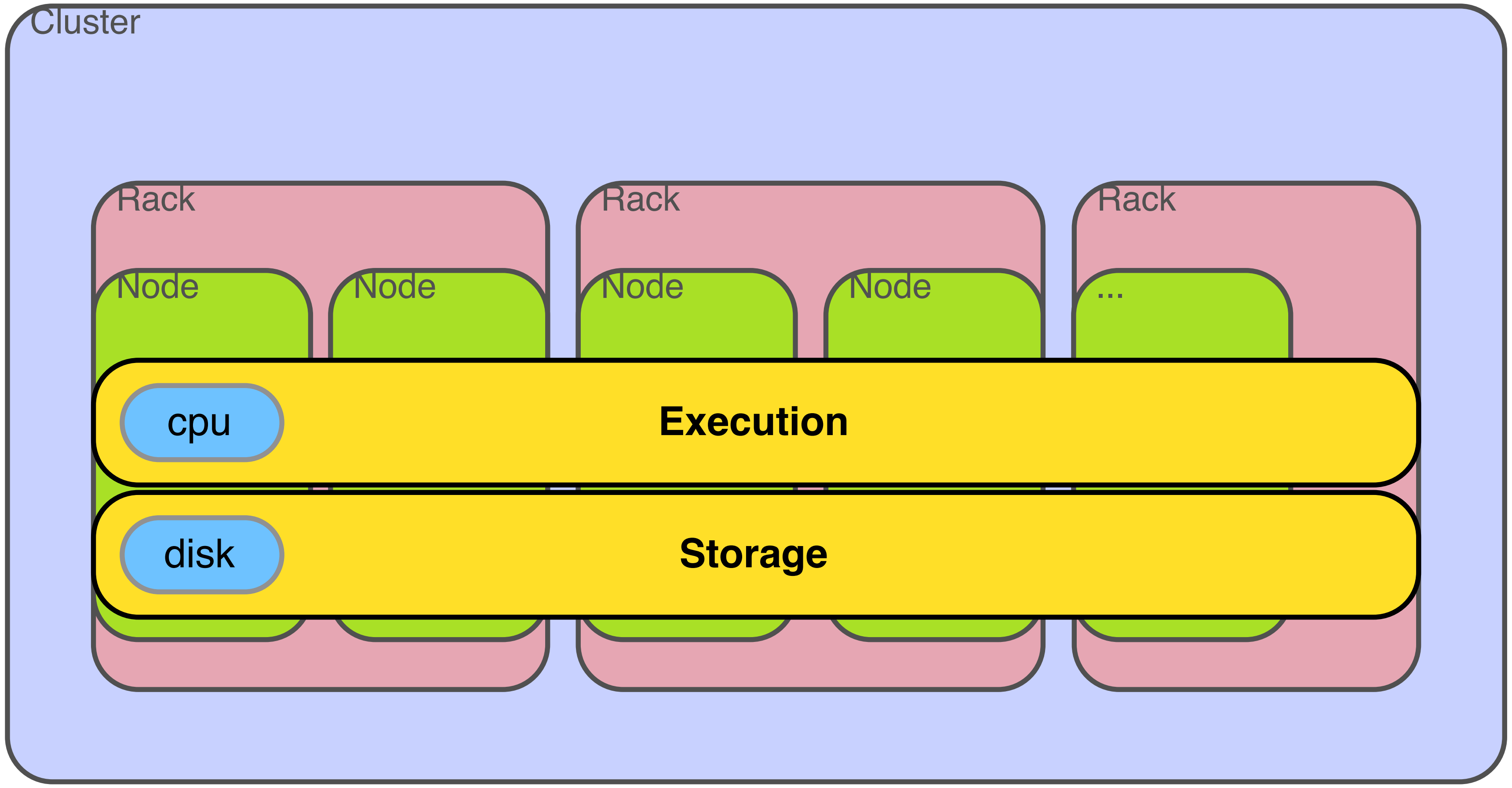
Storage Layer

- Hadoop Distributed File System (aka HDFS, or Hadoop DFS)
- Runs on top of regular OS file system, typically Linux ext3
- Fixed-size blocks (64MB by default) that are replicated
- Write once, read many; optimized for streaming in and out

Execution Layer

- Hadoop Map-Reduce
- Responsible for running a job in parallel on many servers
- Handles re-trying a task that fails, validating complete results
- Jobs consist of special “map” and “reduce” operations

Scalable



- Virtual execution and storage layers span many nodes (servers)
- Scales linearly (sort of) with cores and disks.

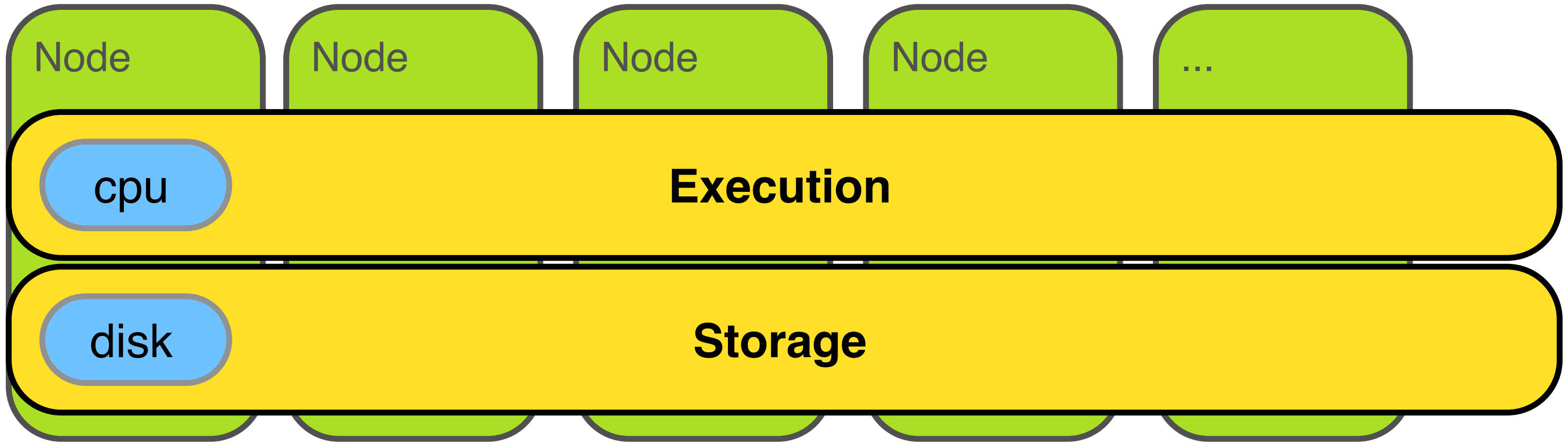
Reliable

- Each block is replicated, typically three times.
- Each task **must** succeed, or the job fails

Fault-tolerant

- Failed tasks are automatically retried.
- Failed data transfers are automatically retried.
- Servers can join and leave the cluster at any time.

Simple



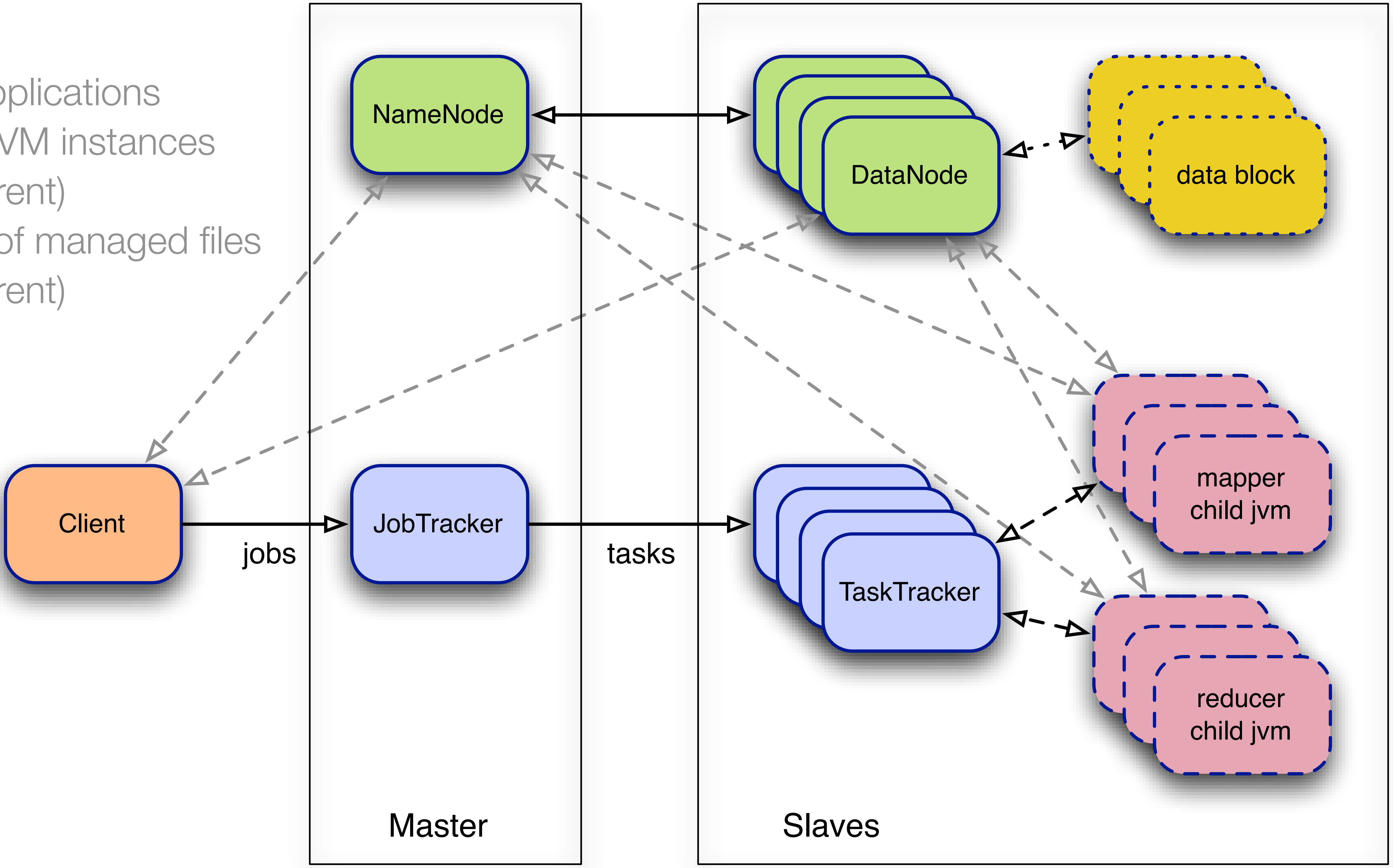
- Reduces complexity
- Conceptual “operating system” that spans many CPUs & disks

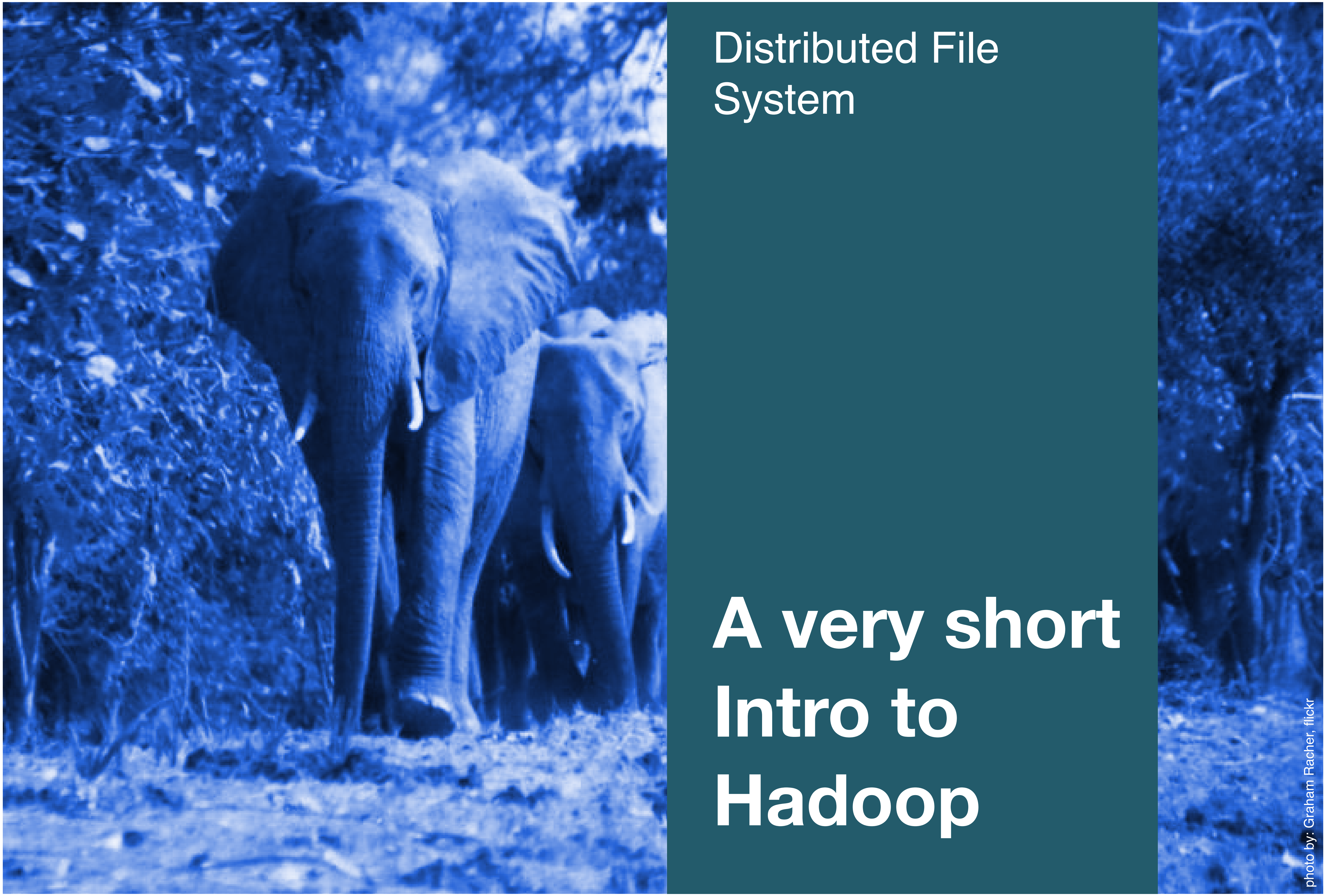
Typical Hadoop Cluster

- Has one “master” server - high quality, beefy box.
  - NameNode process - manages file system
  - JobTracker process - manages tasks
- Has multiple “slave” servers - commodity hardware.
  - DataNode process - manages file system blocks on local drives
  - TaskTracker process - runs tasks on server
- Uses high speed network between all servers

Architectural Components

- Solid boxes are unique applications
- Dashed boxes are child JVM instances
  - (on same node as parent)
- Dotted boxes are blocks of managed files
  - (on same node as parent)





Distributed File System

A very short Intro to Hadoop

photo by: Graham Racher, flickr

Virtual File System

- Treats many disks on many servers as one huge, logical volume
- Data is stored in 1...n blocks
- The “DataNode” process manages blocks of data on a slave.
- The “NameNode” process keeps track of file metadata on the master.

Replication

- Each block is stored on several different disks (default is 3)
- Hadoop tries to copy blocks to different servers and racks.
- Protects data against disk, server, rack failures.
- Reduces the need to move data to code.

Error Recovery

- Slaves constantly “check in” with the master.
- Data is automatically replicated if a disk or server “goes away”.

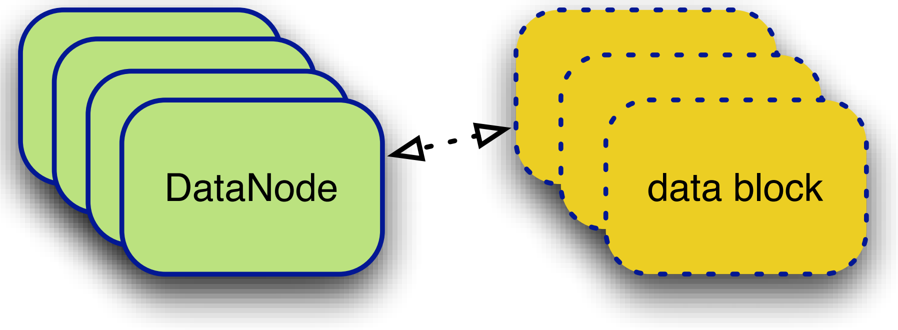
Limitations

- Optimized for streaming data in/out
  - So no random access to data in a file
  - Data rates ≈ 30% - 50% of max raw disk rate
- No append support currently
  - Write once, read many

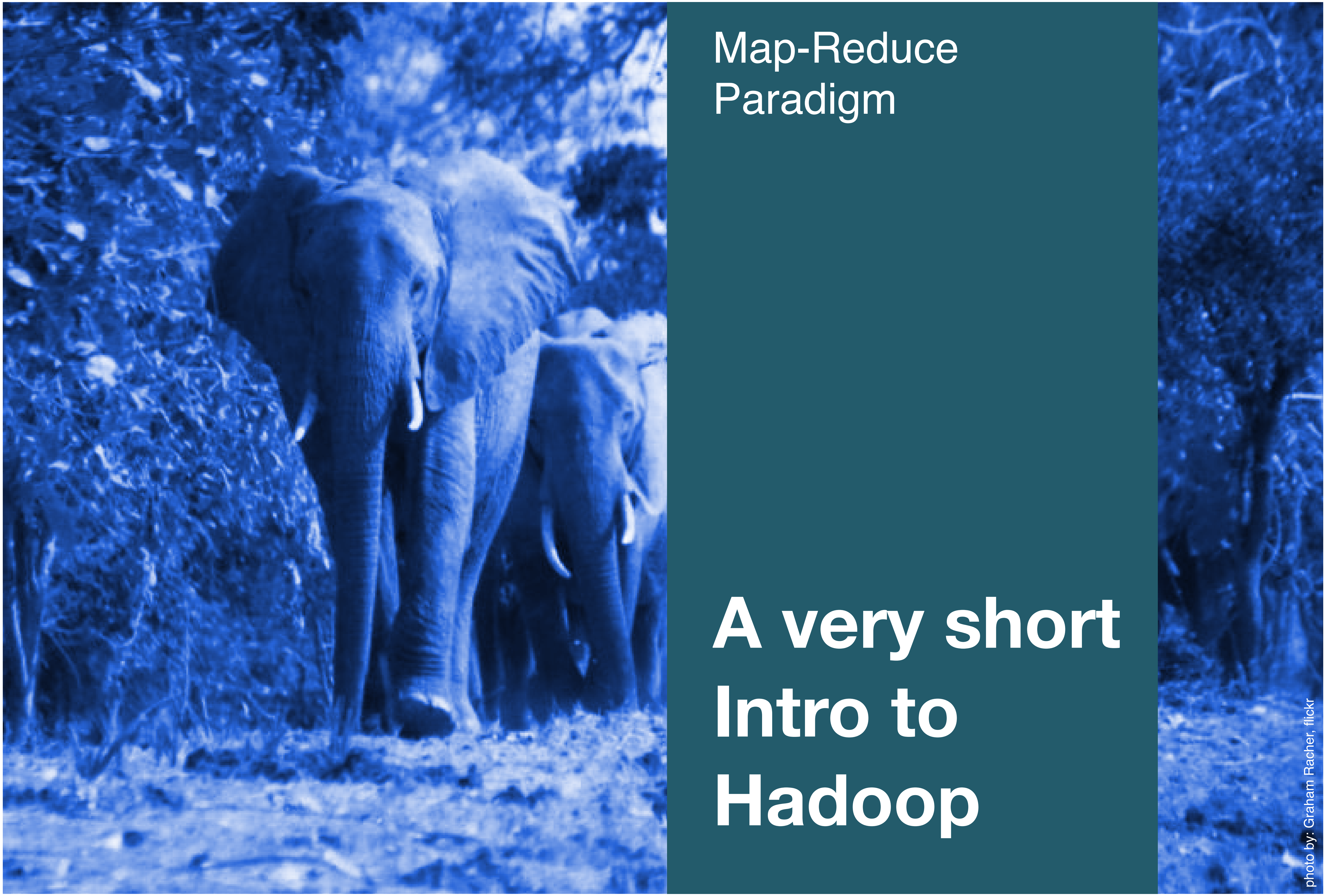
NameNode

- Runs on master node
  - Is a single point of failure
  - There are no built-in software hot failover mechanisms
- Maintains filesystem metadata, the “Namespace”
  - files and hierarchical directories

DataNodes



- Files stored on HDFS are chunked and stored as blocks on DataNode
- Manages storage attached to the nodes that they run on
- Data never flows through NameNode, only DataNodes



Map-Reduce
Paradigm

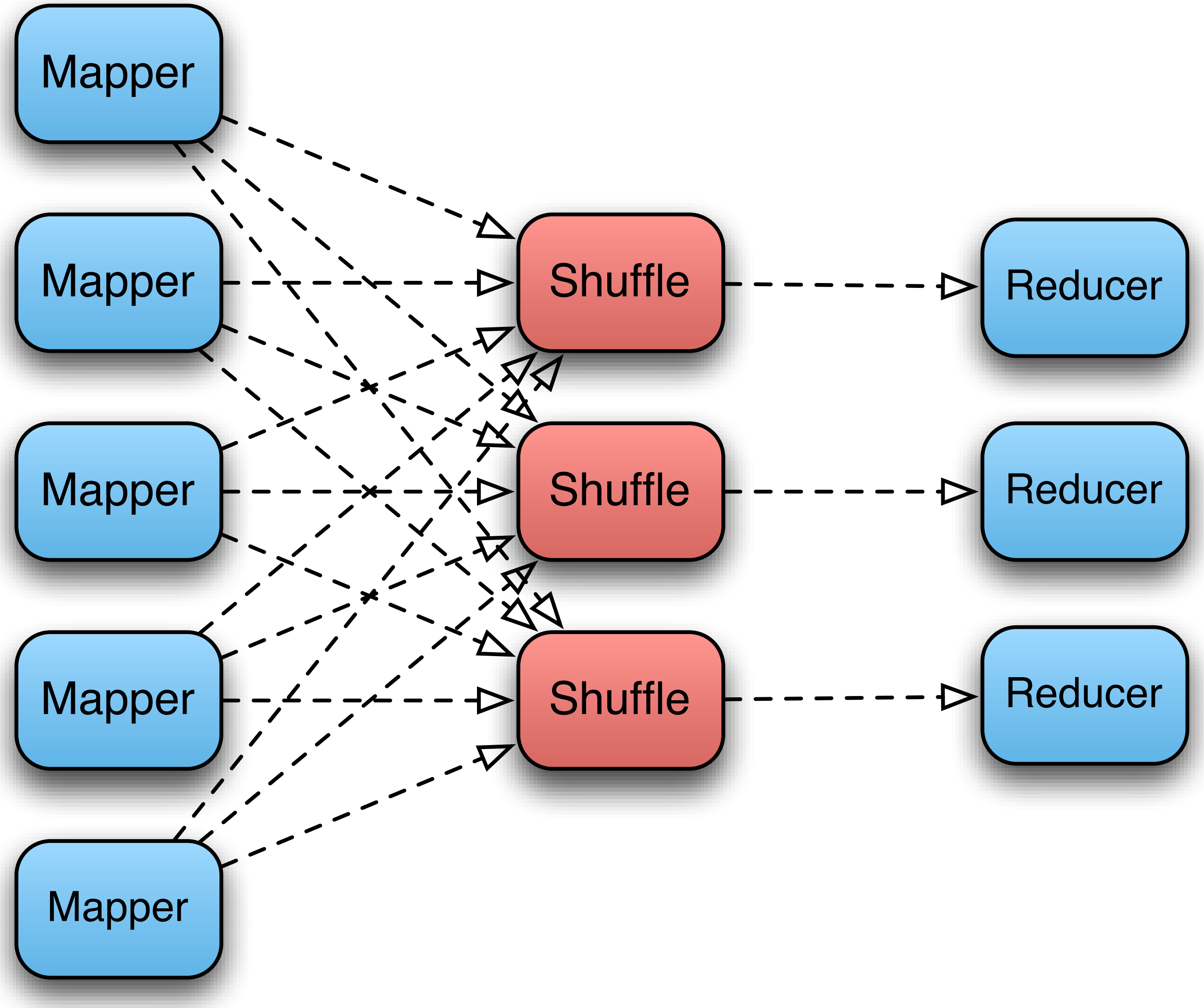
A very short Intro to Hadoop

photo by: Graham Racher, flickr

Definitions

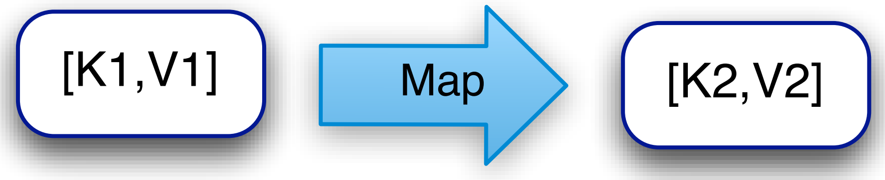
- Key Value Pair -> two units of data, exchanged between Map & Reduce
- Map -> The ‘map’ function in the MapReduce algorithm
    - user defined
    - converts each input Key Value Pair to 0...n output Key Value Pairs
- Reduce -> The ‘reduce’ function in the MapReduce algorithm
    - user defined
    - converts each input Key + all Values to 0...n output Key Value Pairs
- Group -> A built-in operation that happens between Map and Reduce
    - ensures each Key passed to Reduce includes all Values

All Together



How MapReduce Works

- Map translates input to keys and values to new keys and values

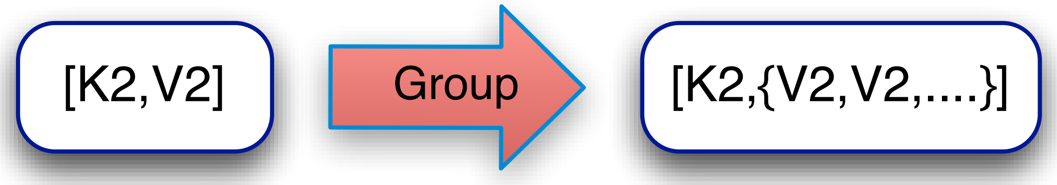


[K1,V1]

Map

[K2,V2]

- System Groups each unique key with all its values

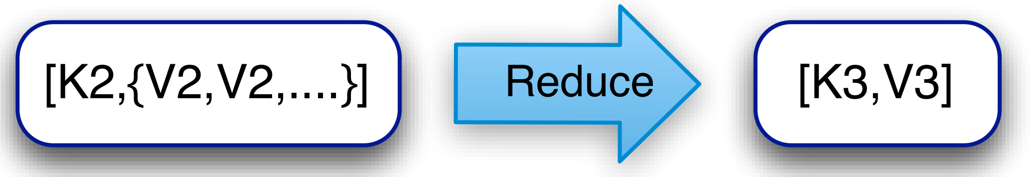


[K2,V2]

Group

[K2,{V2,V2,....}]

- Reduce translates the values of each unique key to new keys and values



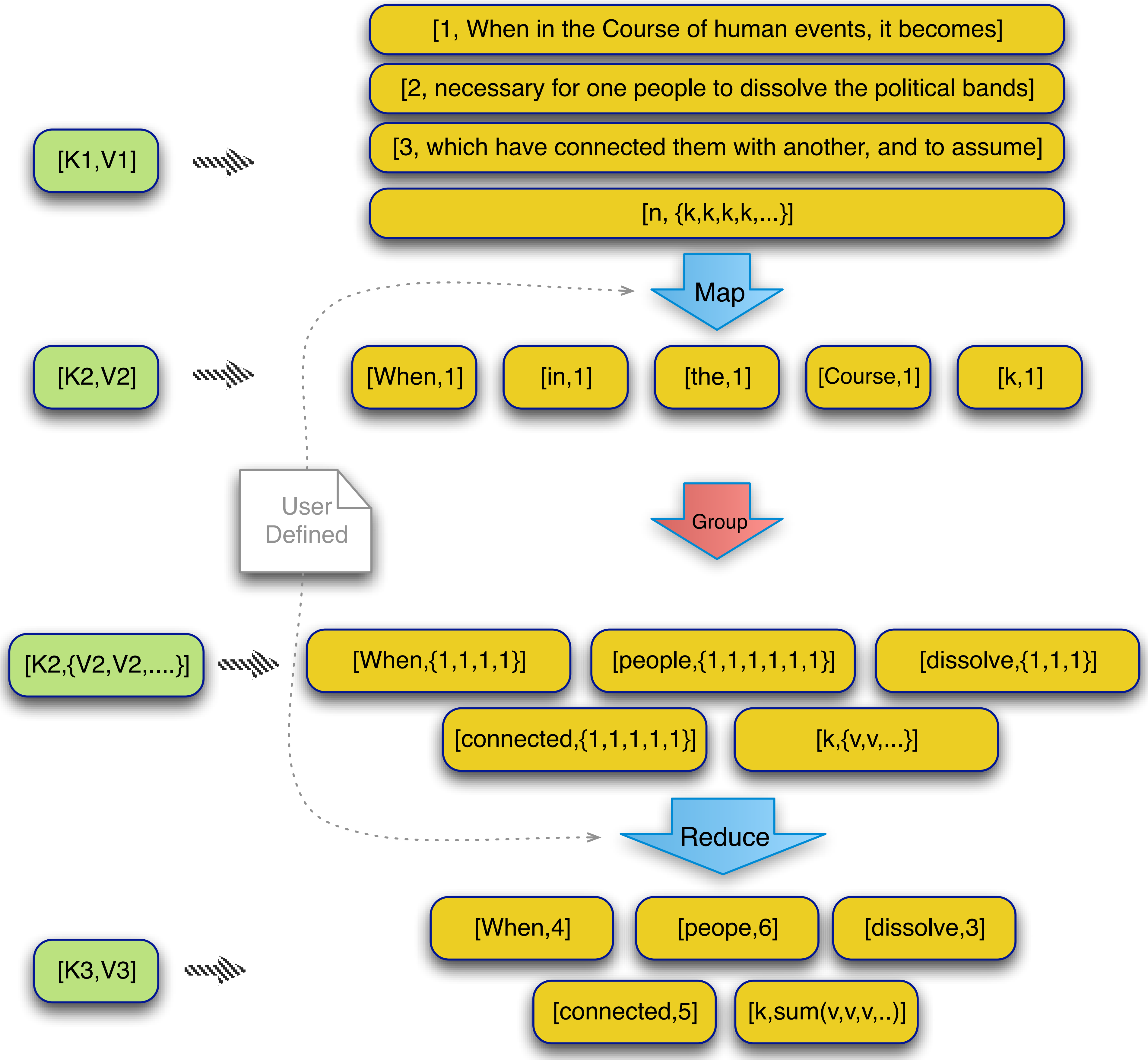
[K2,{V2,V2,....}]

Reduce

[K3,V3]

Canonical Example - Word Count

- Word Count
  - Read a document, parse out the words, count the frequency of each word
- Specifically, in MapReduce
  - With a document consisting of lines of text
  - Translate each line of text into key = word and value = 1
    - e.g. <"the",1> <"quick",1> <"brown",1> <"fox",1> <"jumped",1>
  - Sum the values (ones) for each unique word



[K1,V1]

[1, When in the Course of human events, it becomes]

[2, necessary for one people to dissolve the political bands]

[3, which have connected them with another, and to assume]

[n, {k,k,k,k,...}]

Map

[K2,V2]

[When,1] [in,1] [the,1] [Course,1] [k,1]

User Defined

Group

[K2,{V2,V2,....}]

[When,{1,1,1,1}] [people,{1,1,1,1,1,1}] [dissolve,{1,1,1}]

[connected,{1,1,1,1,1}] [k,{v,v,...}]

Reduce

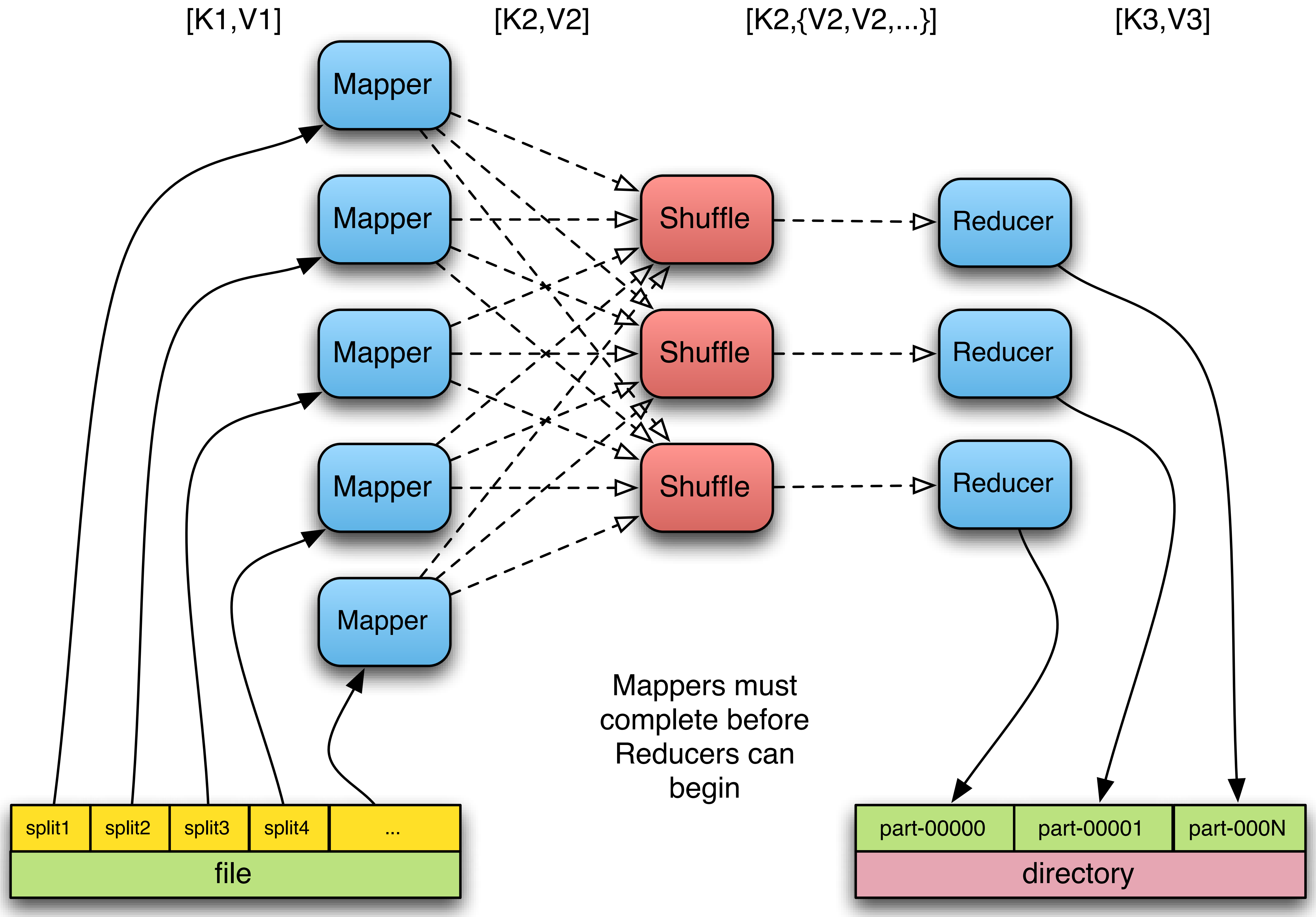
[K3,V3]

[When,4] [peope,6] [dissolve,3]

[connected,5] [k,sum(v,v,v,..)]

Divide & Conquer (splitting data)

- Because
  - The Map function only cares about the current key and value, and
  - The Reduce function only cares about the current key and its values
- Then
  - A Mapper can invoke Map on an arbitrary number of input keys and values
    - or just some fraction of the input data set
  - A Reducer can invoke Reduce on an arbitrary number of the unique keys
    - but all the values for that key

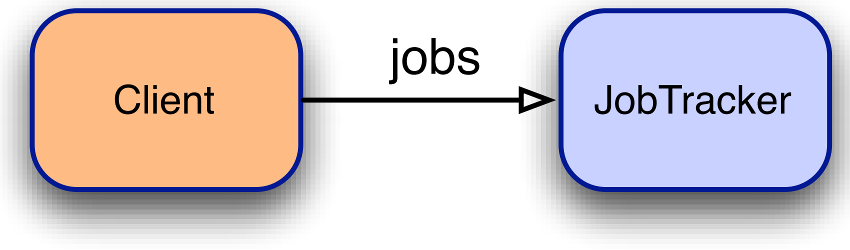


Mappers must complete before Reducers can begin

Divide & Conquer (parallelizable)

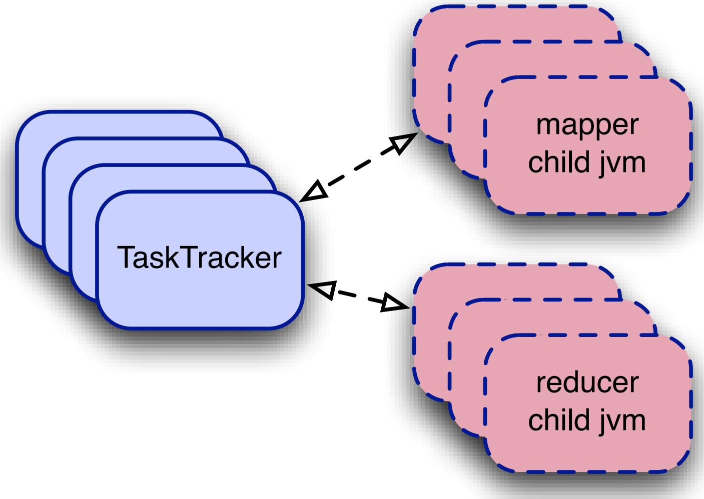
- Because
  - Each Mapper is independent and processes part of the whole, and
  - Each Reducer is independent and processes part of the whole
- Then
  - Any number of Mappers can run on each node, and
  - Any number of Reducers can run on each node, and
  - The cluster can contain any number of nodes

JobTracker



- Is a single point of failure
- Determines # Mapper Tasks from file splits via InputFormat
- Uses predefined value for # Reducer Tasks
- Client applications use JobClient to submit jobs and query status
- Command line use **`hadoop job <commands>`**
- Web status console use http://jobtracker-server:50030/

TaskTracker



- Spawns each Task as a new child JVM
- Max # mapper and reducer tasks set independently
- Can pass child JVM opts via `mapred.child.java.opts`
- Can re-use JVM to avoid overhead of task initialization

Hadoop (sort of)
Deep Thoughts

A very short Intro to Hadoop

photo by: Graham Racher, flickr

Avoiding Hadoop

- Hadoop is a big hammer - but not every problem is a nail
  - Small data
  - Real-time data
  - Beware the Hadoopaphile

Avoiding Map-Reduce

- Writing Hadoop code is painful and error prone
- Hive & Pig are good solutions for People Who Like SQL
- Cascading is a good solution for complex, stable workflows

Leveraging the Eco-System

- Many open source projects built on top of Hadoop
  - HBase - scalable NoSQL data store
  - Sqoop - getting SQL data in/out of Hadoop
- Other projects work well with Hadoop
  - Kafka/Scribe - getting log data in/out of Hadoop
  - Avro - data serialization/storage format

Get Involved

- Join the mailing list - http://hadoop.apache.org/mailing_lists.html
- Go to user group meetings - e.g. http://hadoop.meetup.com/

Learn More

- Buy the book - Hadoop: The Definitive Guide, 2nd edition
- Try the tutorials
  - http://hadoop.apache.org/common/docs/current/mapred_tutorial.html
- Get training (danger, personal plug)
  - http://www.scaleunlimited.com/training
  - http://www.cloudera.com/hadoop-training

Resources

- Scale Unlimited Alumni list - scale-unlimited-alumni@googlegroups.com
- Hadoop mailing lists - http://hadoop.apache.org/mailing_lists.html
- Users groups - e.g. http://hadoop.meetup.com/
- Hadoop API - http://hadoop.apache.org/common/docs/current/api/
- Hadoop: The Definitive Guide, 2nd edition by Tom White
- Cascading: http://www.cascading.org
- Datameer: http://www.datameer.com