BDA400 – Lecture 2
Introduction to Hadoop

hadoop

Open Source

The Apache Software
Foundation

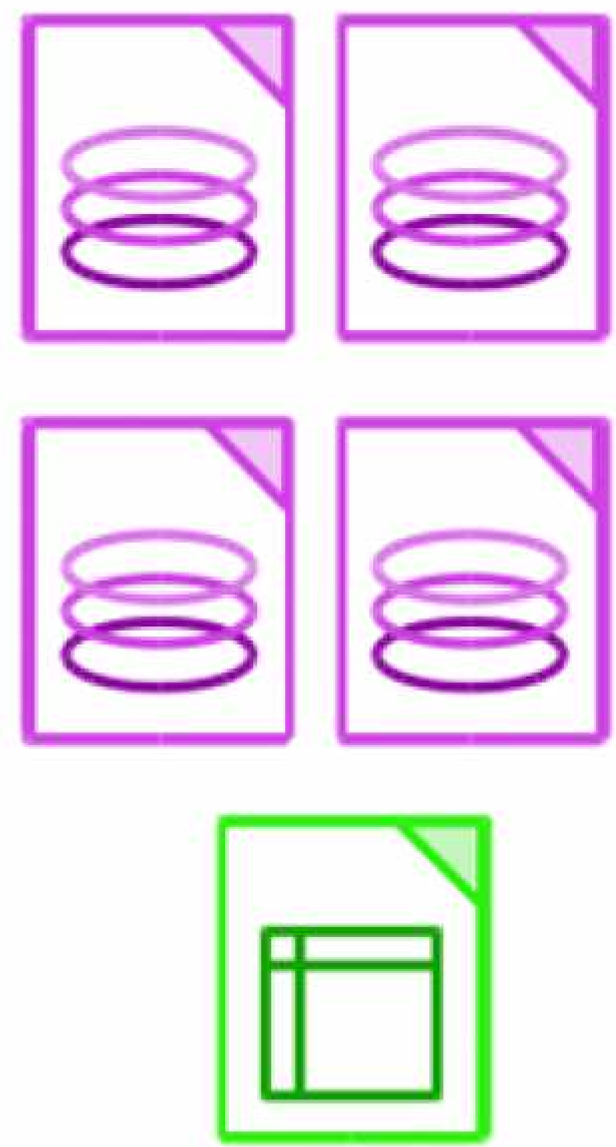
JAVA

SEMI UNSTRUCTURED

Data

80%

Is UnStructured Data



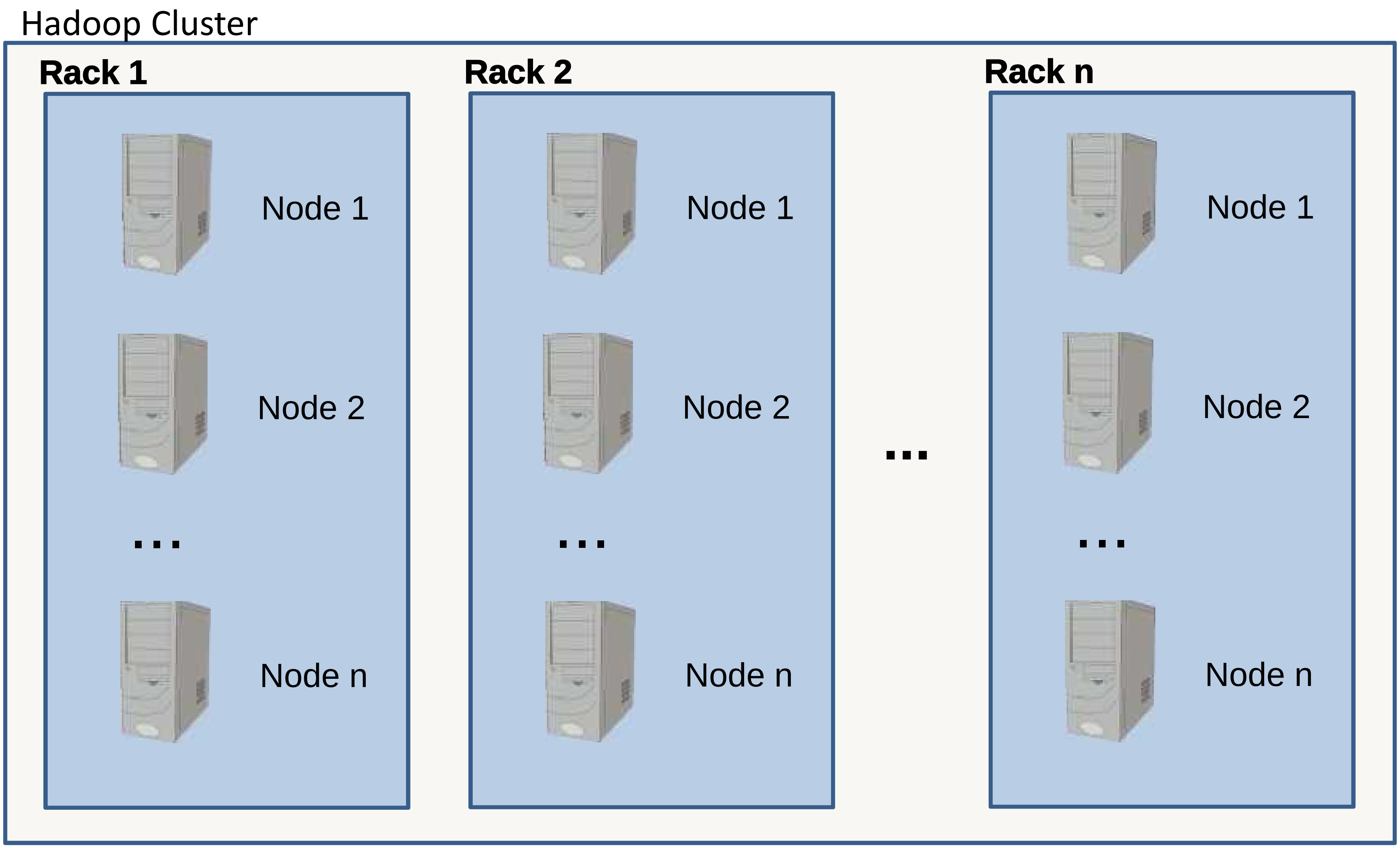
Parallel Processing

Parallel Processing

Parallel Processing

Parallel Processing

Terminology review



Hadoop-related open source projects

PIG

ZooKeeper

Hadoop is not for all types of work

- Not to process transactions (random access)
- Not good when work cannot be parallelized
- Not good for low latency data access
- Not good for processing lots of small files
- Not good for intensive calculations with little data

Big Data solutions and the Cloud

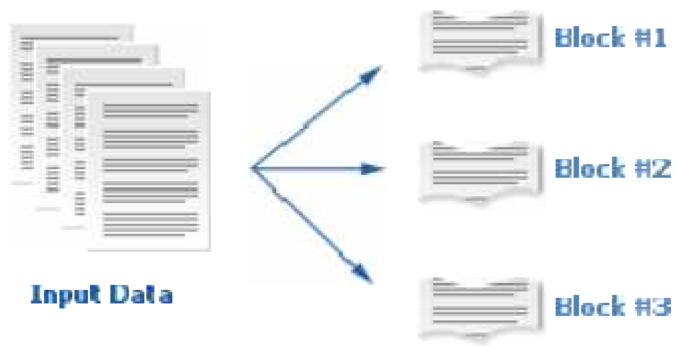
- Big Data solutions are more than just Hadoop
  - Add business intelligence/analytics functionality
  - Derive information of data in motion
- Big Data solutions and the Cloud are a perfect fit
  - The Cloud allows you to set up a cluster of systems in minutes and it's relatively inexpensive

Pre Hadoop 2.2 architecture

- Two main components
  - Distributed File System
    - Hadoop Distributed File System (HDFS)
    - IBM Spectrum Scale
  - MapReduce Engine
    - Framework for performing calculations on the data in the file system
    - Has a built-in resource manager and scheduler

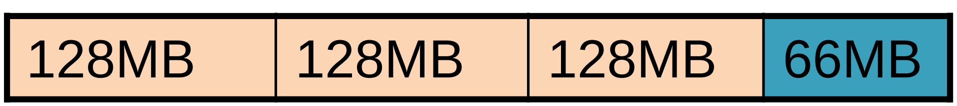
Hadoop Distributed File System (HDFS)

- HDFS runs on top of the existing file system
  - Not POSIX compliant
  - Designed to tolerate high component failure rate
    - Reliability is through replication
- Designed to handle very large files
  - Large streaming data access patterns
    - No random access
- Uses blocks to store a file or parts of a file



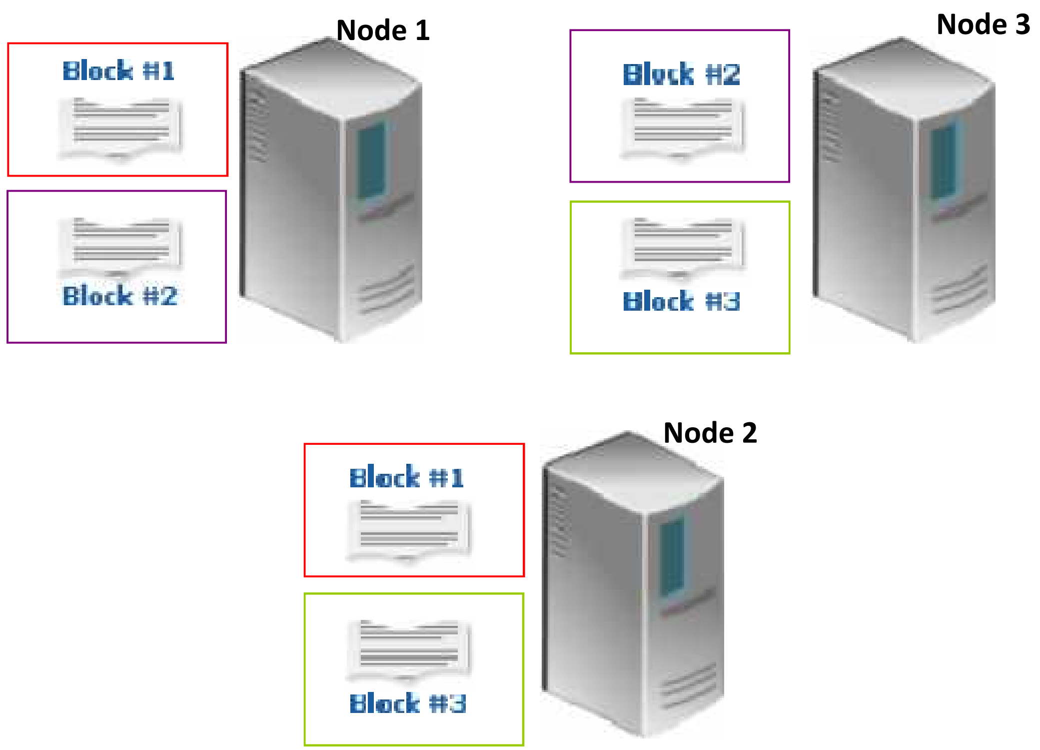
HDFS file blocks

- Not the same as the operating system's file blocks
  - HDFS book made up of multiple operating system blocks
- Default for Hadoop is 64MB
  - Recommended is 128MB (this is the BigInsights default)
- Size of a file can be larger than any single disk in the cluster
  - Blocks for a single file are spread across multiple nodes in the cluster
- If a chunk of the file is smaller than the HDFS block size
  - Only the needed space is used
- Blocks work well with replication

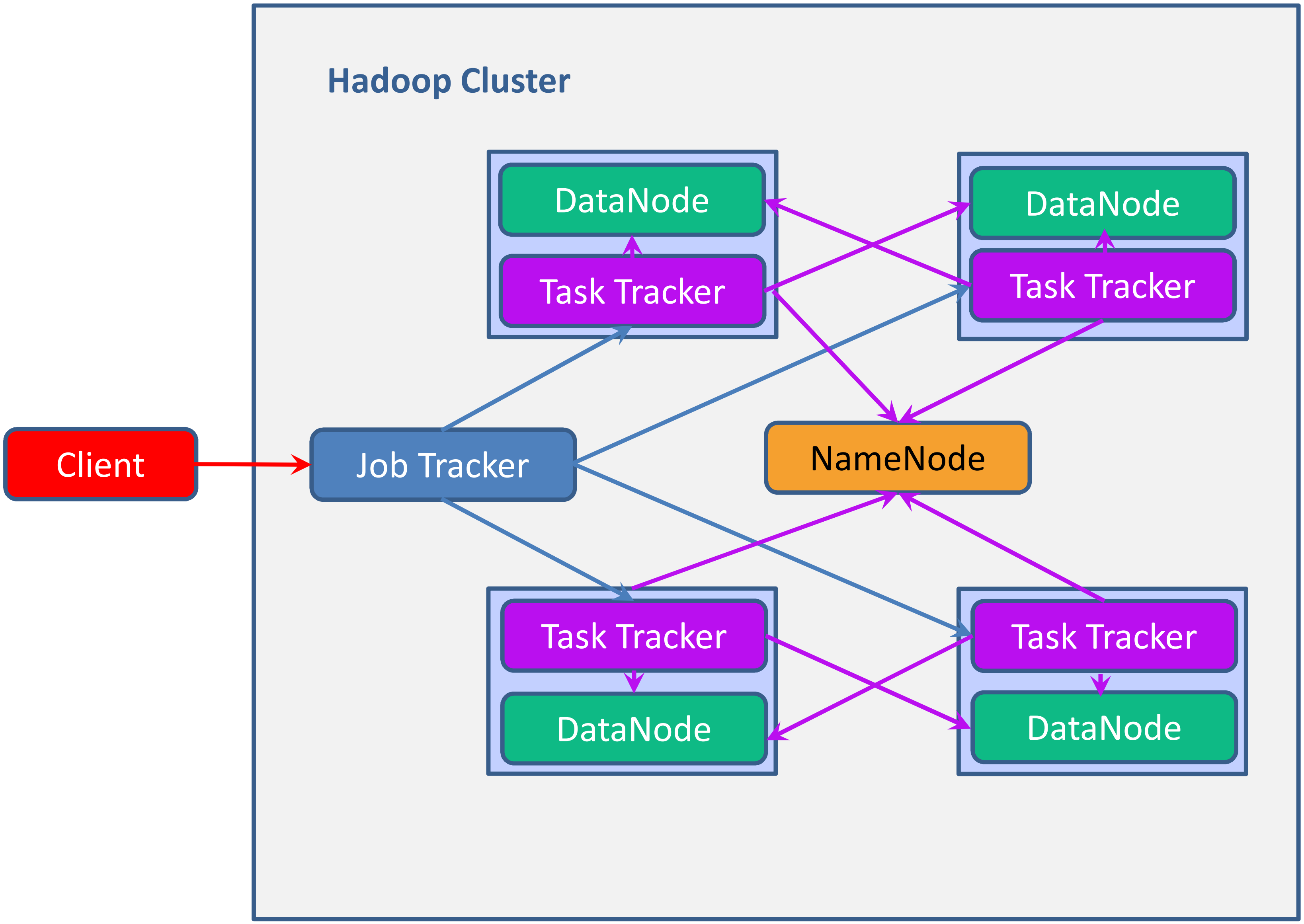


HDFS - Replication

- Blocks with data are replicated to multiple nodes
- Allows for node failure without data loss

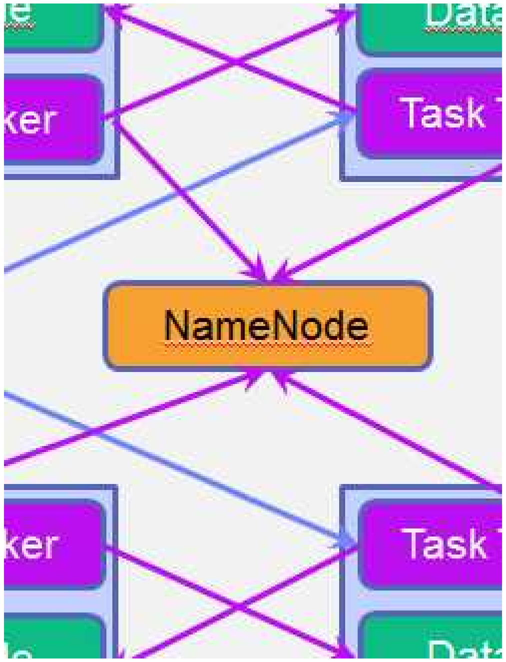


Types of nodes - overview



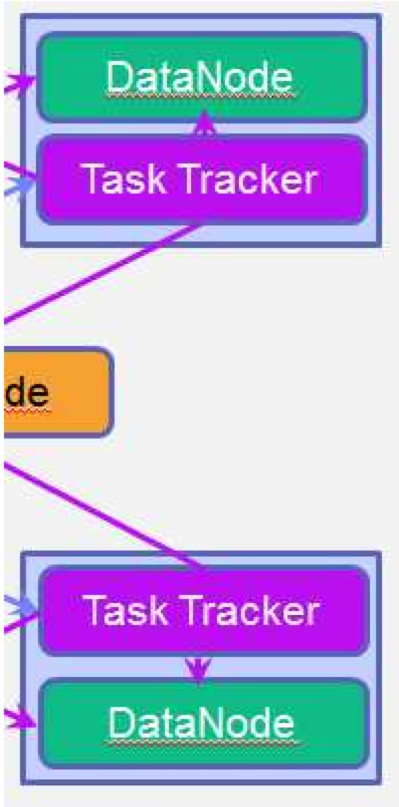
Types of nodes - NameNode

- Only one per Hadoop cluster
- Manages the file system namespace and metadata
  - Data does not go through the NameNode
  - Data is not stored on the NameNode
- Single point of failure
  - Good idea to mirror the NameNode
  - Do not use inexpensive, commodity hardware
- Has large memory requirement
  - File system metadata is maintained in RAM to server read requests



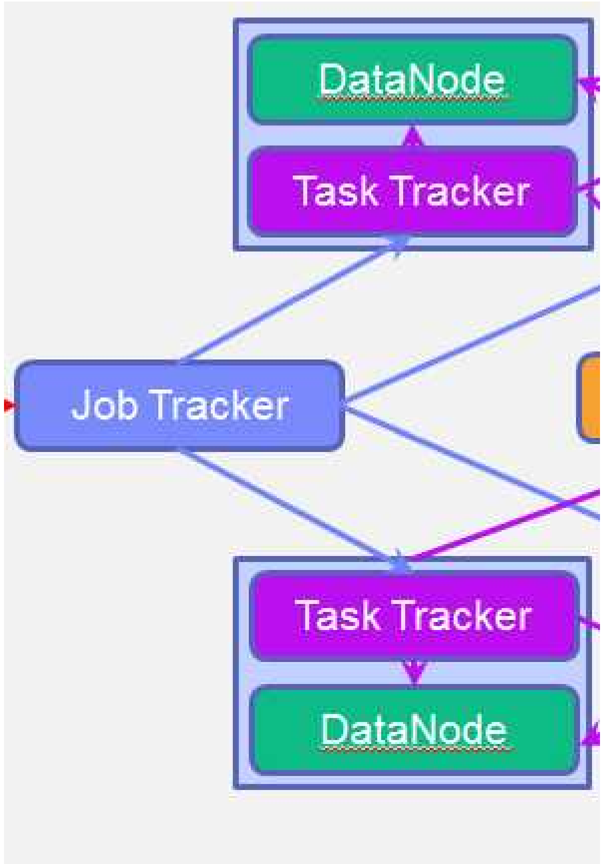
Types of nodes - DataNode

- Many per Hadoop cluster
- Blocks from different files can be stored on the same DataNode
- Manages blocks with data and serves them to clients
- Periodically reports to NameNode the list of blocks it stores
- Suitable for inexpensive, commodity hardware



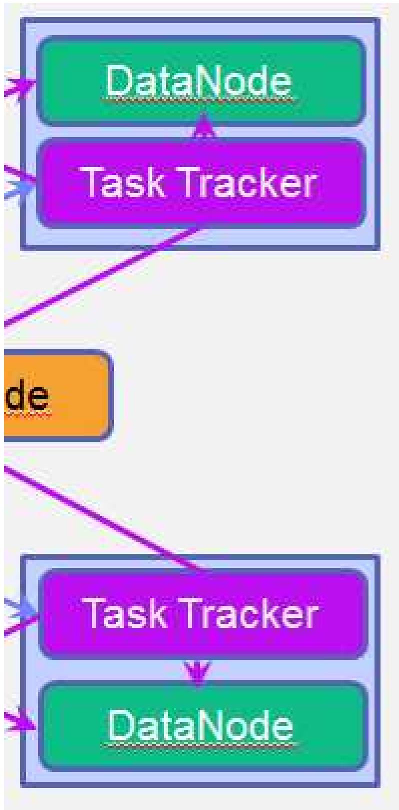
Types of nodes - JobTracker

- Manages the MapReduce jobs in the cluster
- One per Hadoop cluster
- Receives job requests submitted by the client
- Schedules and monitors MapReduce jobs on TaskTrackers
  - Attempts to direct a task to the TaskTracker where the data resides



Types of nodes - TaskTracker

- Many per Hadoop cluster
- Executes the MapReduce operations
  - Runs the MapReduce tasks in JVMs
  - Have a set number of slots used to run tasks
  - Communicates with the JobTracker via heartbeat messages
  - Reads blocks from DataNodes



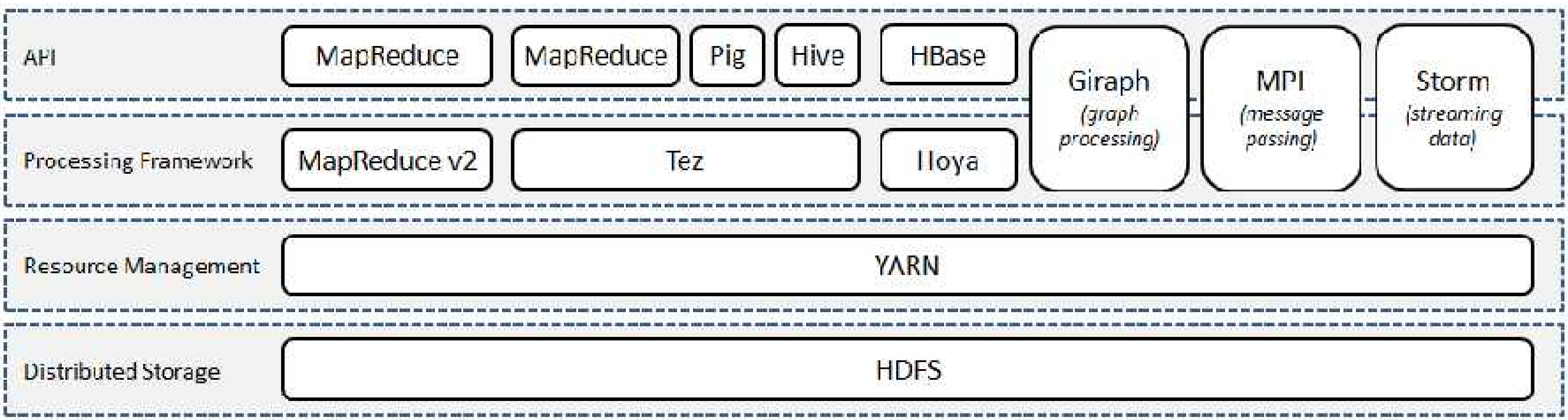
Hadoop 2.2 architecture

- Provides YARN
  - Referred to as MapReduce V2
  - Resource manager and scheduler external to any framework
  - DataNodes still exist
  - JobTracker and TaskTrackers no longer exist

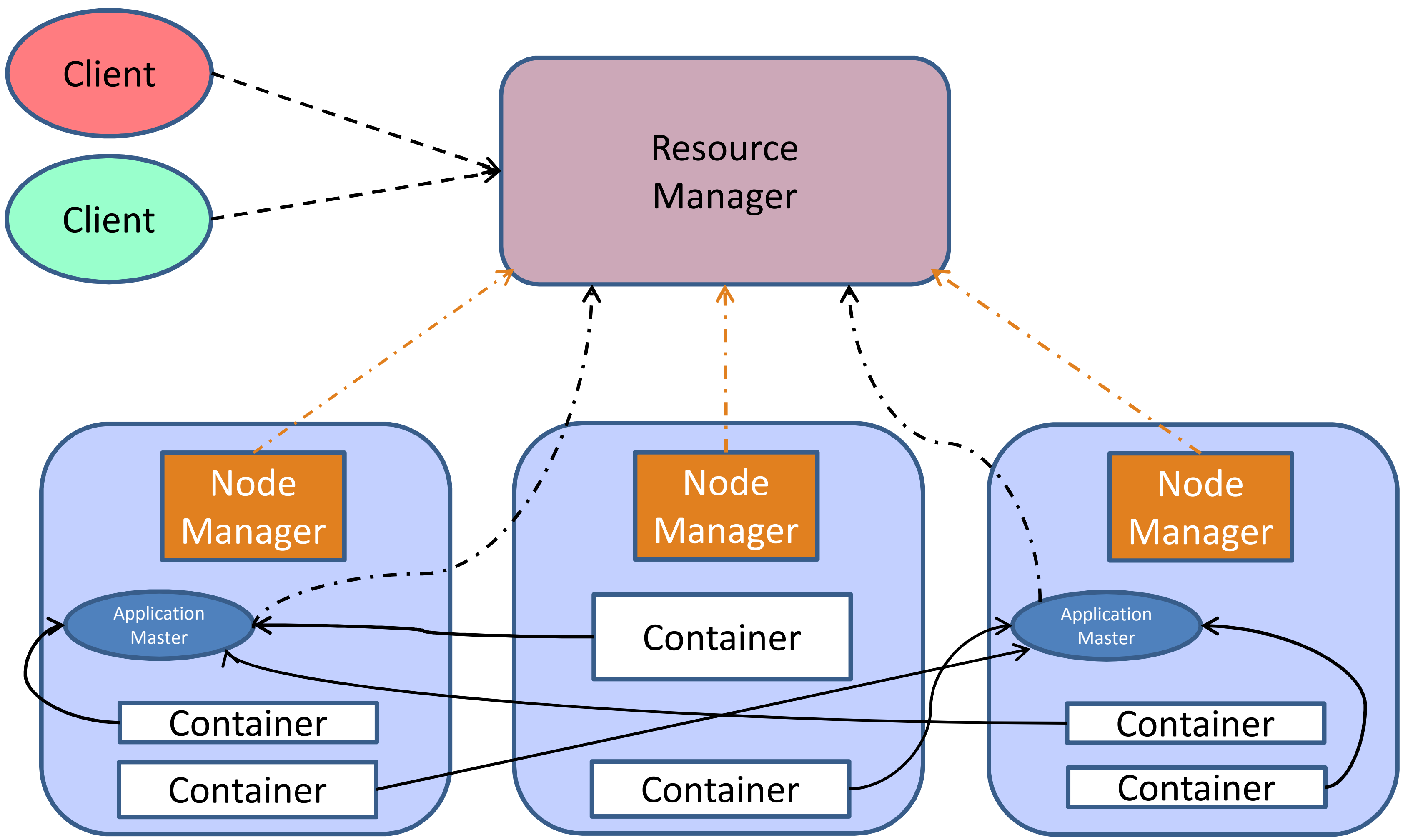
- Not a requirement to run YARN with Hadoop 2.2
  - Still supports MapReduce V1

YARN

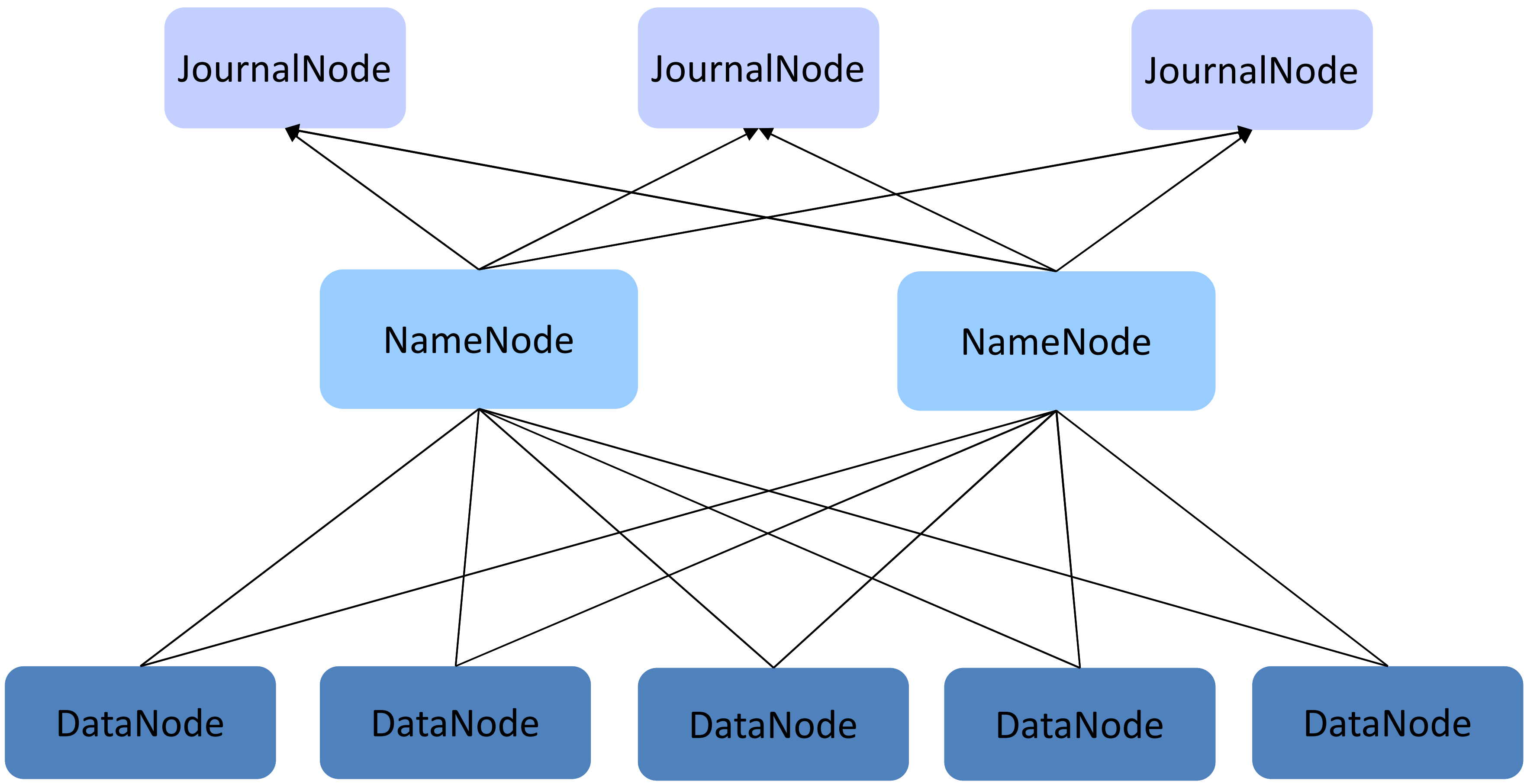
- **Two main ideas**
  - Provide generic scheduling and resource management
    - Support more than just MapReduce
    - Support more than just batch processing
  - More efficient scheduling and workload management
    - No more balancing between map slots and reduce slots!



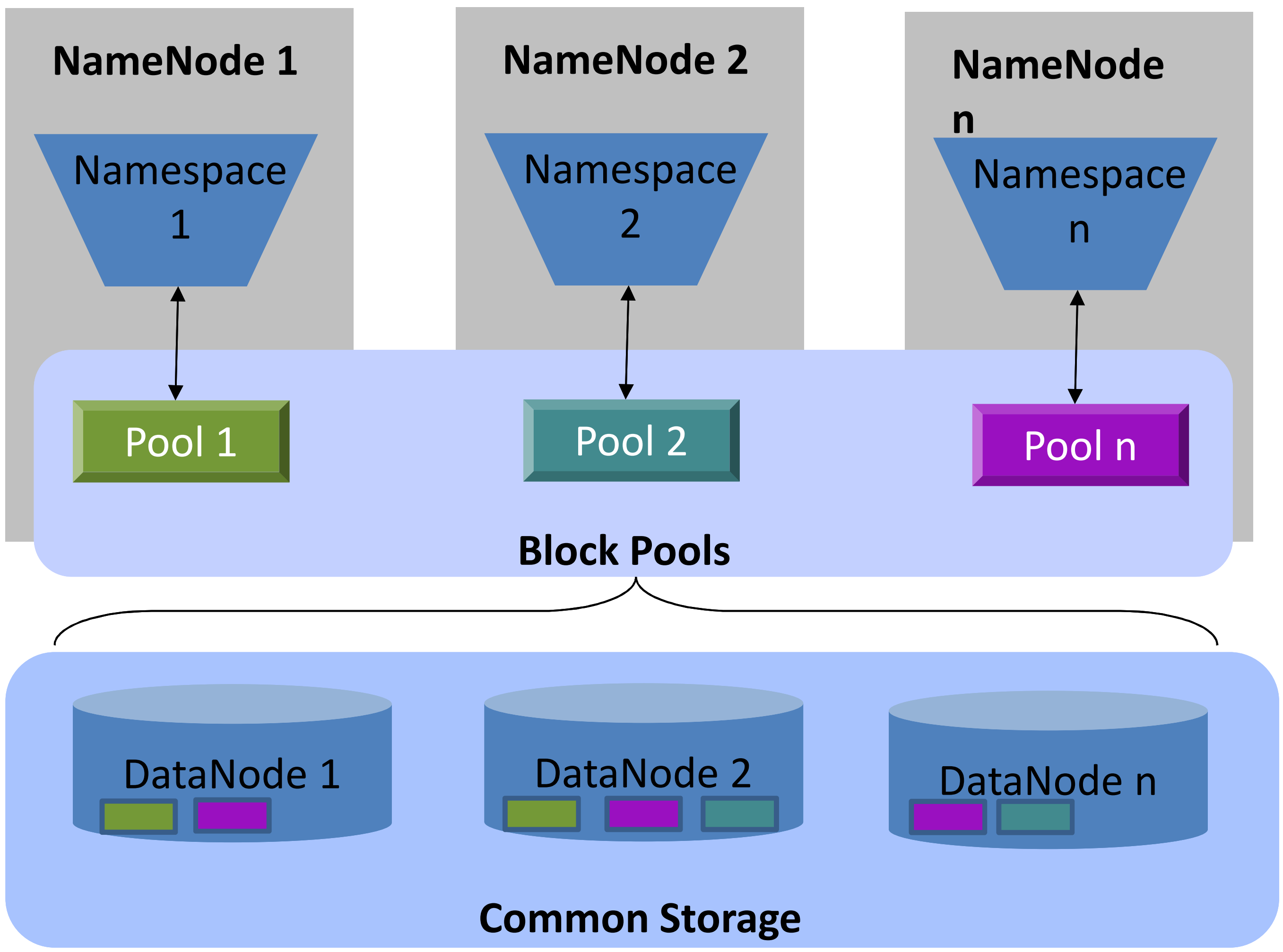
YARN Overview



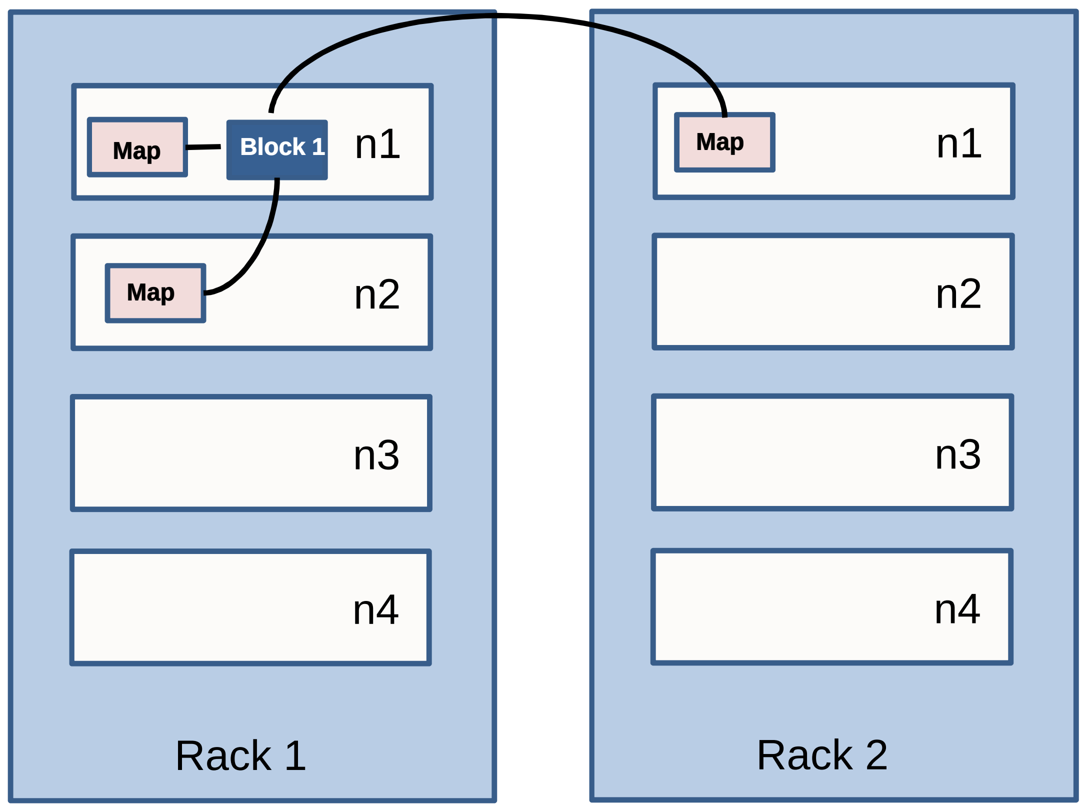
Hadoop High Availability



Hadoop Federation

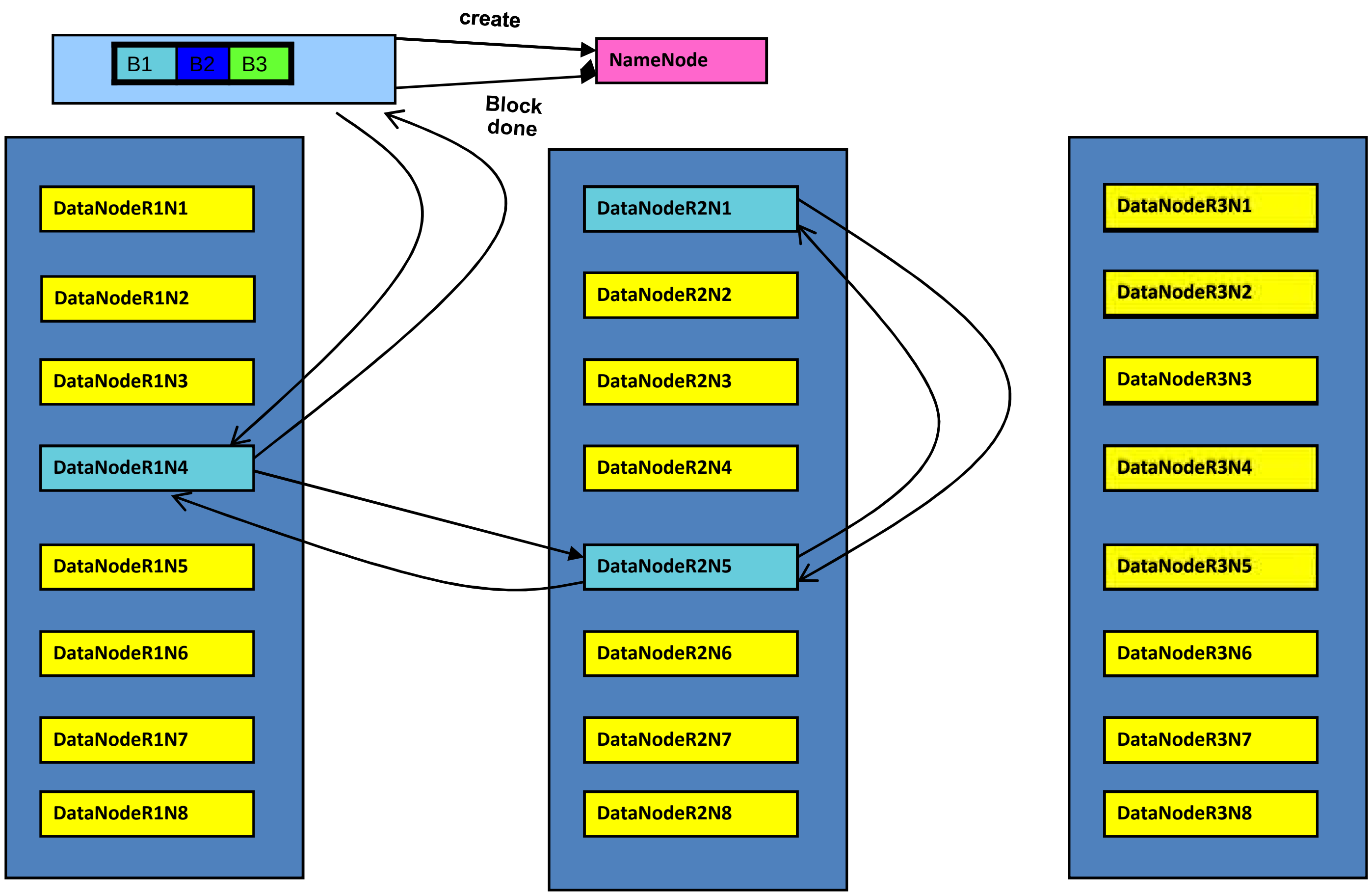


Topology awareness



The administrator defines the topology using the *topology.script.file.name* property in *core-site.xml*

HDFS - replication



HDFS file command interface

- The FileSystem (FS) shell is invoked by

  hdfs dfs <args>

- Example to list the current directory in HDFS

  hdfs dfs -ls .

HDFS file command interface

- All FS shell commands take path URIs as arguments

  scheme://authority/path

- Scheme
  - Scheme for HDFS is *hdfs*
  - Scheme for the local filesystem is *file*

```
hdfs dfs -cp
     file:///sampleData/spark/myfile.txt
     hdfs://rvm.svl.ibm.com:8020/user/spark/test/myfile.txt
```

- Scheme and authority are optional
  - Defaults are taken from the *core-site.xml* configuration file
- Most of the FS shell commands behave like corresponding UNIX commands

HDFS file commands

- A number of POSIX-like commands
  - cat, chgrp, chmod, chown, cp, du, ls, mkdir, mv, rm, stat, tail
- Some HDFS-specific commands
  - copyFromLocal, copyToLocal, get, getmerge, put, setrep

HDFS - specific commands

- copyFromLocal / put
  - Copy files from the local filesystem to HDFS
- copyToLocal / get
  - Copies files from HDFS to the local filesystem
- getMerge
  - Gets all files in the directories that match the source pattern
  - Merges and sorts them to only one file on local filesystem
- setRep
  - Sets the replication factor of a file
  - Can be executed recursively to change an entire tree
  - Can specify to wait until the replication level is achieved

Adding and removing nodes from the cluster

- Can be performed from Ambari Web Console
  - Need IP address or hostname of node to add
  - Node must be reachable

    Eg: `ssh 192.168.44.15`

  - BigInsights on node to add must NOT be installed
  - /etc/hosts on both master and child nodes should be updated prior to adding child nodes

Verifying cluster health – Disk space

- Performed DFS Disk Check by running DFS Report
  - Helps determine if there is low disk storage
  - Can be viewed from the Ambari Web Console
- Run DFS Report using:

  `hdfs dfsadmin –report`

```
[hdfs@rvm /]$ hdfs dfsadmin -report
Configured Capacity: 18433347584 (17.17 GB)
Present Capacity: 14663487488 (13.66 GB)
DFS Remaining: 14081208320 (13.11 GB)
DFS Used: 582279168 (555.30 MB)
DFS Used%: 3.97%
Under replicated blocks: 22
Blocks with corrupt replicas: 0
Missing blocks: 0

-------------------------------------------------
Live datanodes (1):

Name: 172.17.0.7:50010 (rvm.svl.ibm.com)
Hostname: rvm.svl.ibm.com
Decommission Status : Normal
Configured Capacity: 18433347584 (17.17 GB)
DFS Used: 582279168 (555.30 MB)
Non DFS Used: 3769860096 (3.51 GB)
DFS Remaining: 14081208320 (13.11 GB)
DFS Used%: 3.16%
DFS Remaining%: 76.39%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 4
Last contact: Wed Jul 15 16:39:59 UTC 2015


[hdfs@rvm /]$ _
```

Configuring Hadoop - Example

- Stop appropriate services before making the change
- Change to the conf directory, look for hdfs-site.xml:

cd /usr/iop/current/hadoop-client/conf

vi hdfs-site.xml

```
<! Fri Jan  8 12:06:28 2016  >
  <configuration>

  <property>
    <name>dfs.block.access.token.enable</name>
    <value>true</value>
  </property>

  <property>
    <name>dfs.blockreport.initialDelay</name>
    <value>120</value>
  </property>

  <property>
    <name>dfs.blocksize</name>
    <value>134217728</value>
  </property>

  <property>
    <name>dfs.client.file.block.storage.locations.timeout.millis</name>
    <value>3000</value>
  </property>
```

Setting Rack Topology (Rack Awareness)

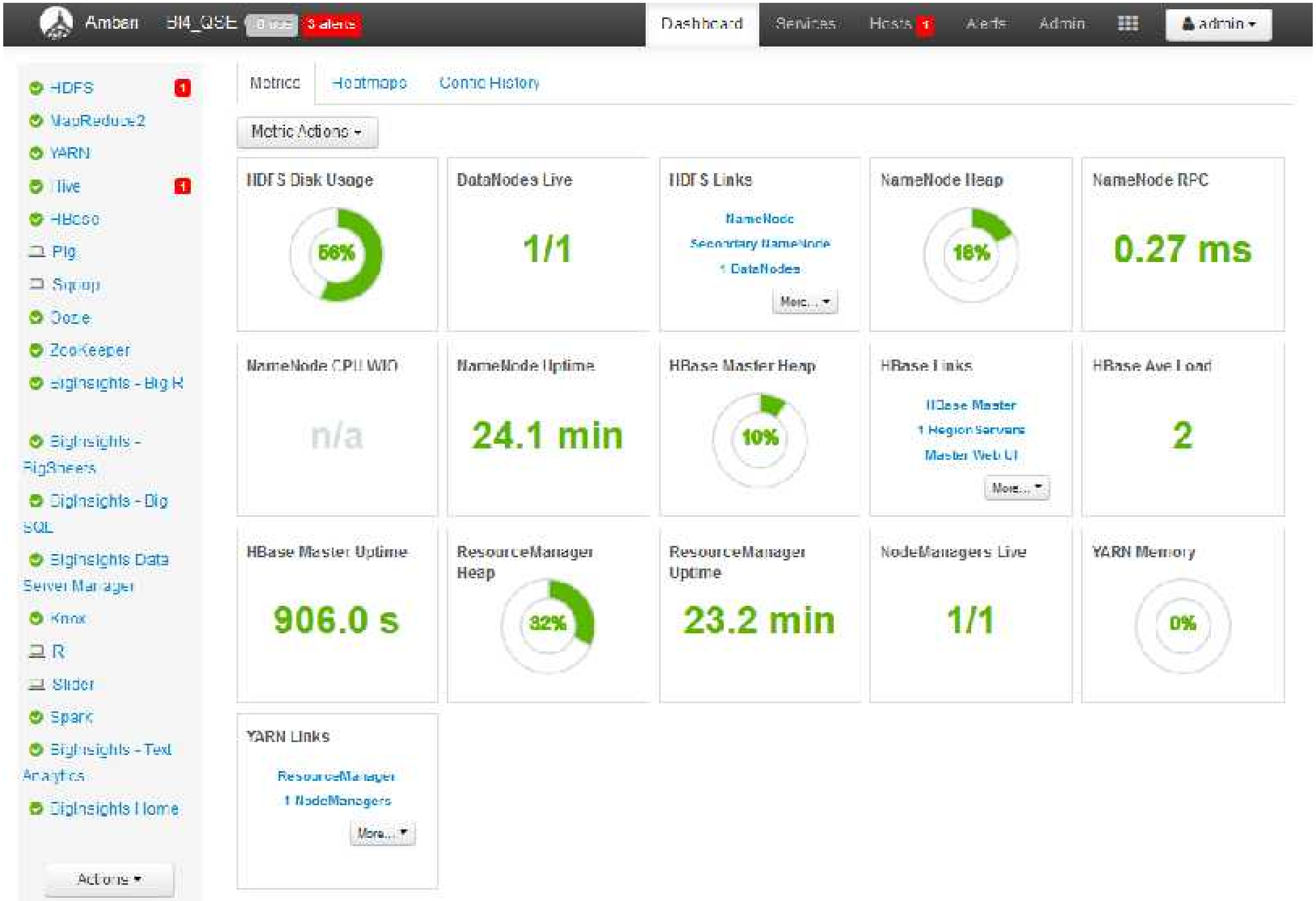
- Can be defined by script which specifies which node is on which rack.
- Script is referenced in **topology.script.file.name** property in **core-site.xml**.
  - Example of property:

```
<property>
    <name>topology.script.file.name</name>
    <value>/opt/ibm/biginsights/hadoop-conf/rack-aware.sh</value>
<property>
```

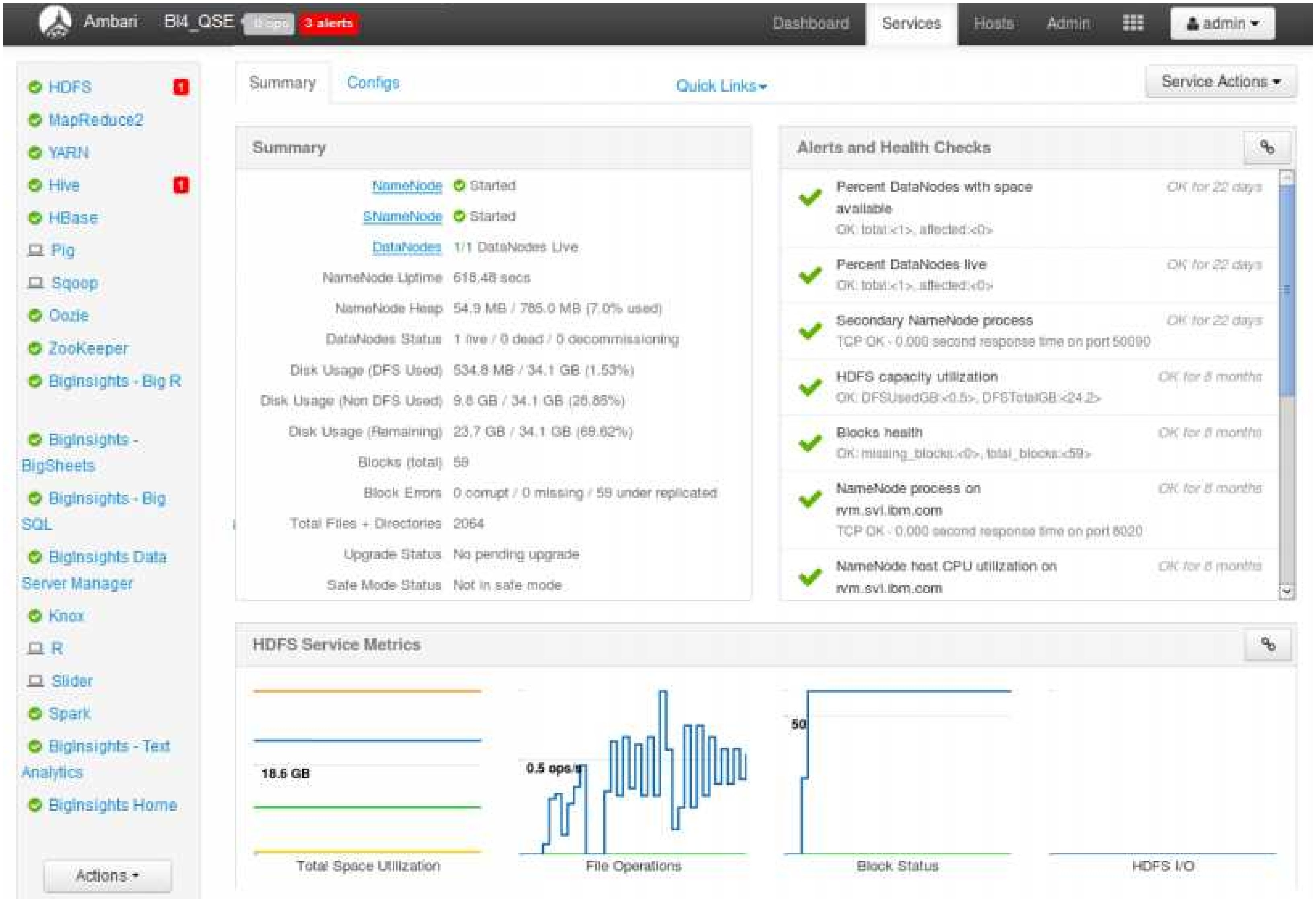
- The *network topology script* (**topology.script.file.name** in the above example) receives as arguments one or more IP addresses of nodes in the cluster. It returns on stdout a list of rack names, one for each input. The input and output order must be consistent.

- Example: http://wiki.apache.org/hadoop/topology_rack_awareness_scripts

Ambari Console



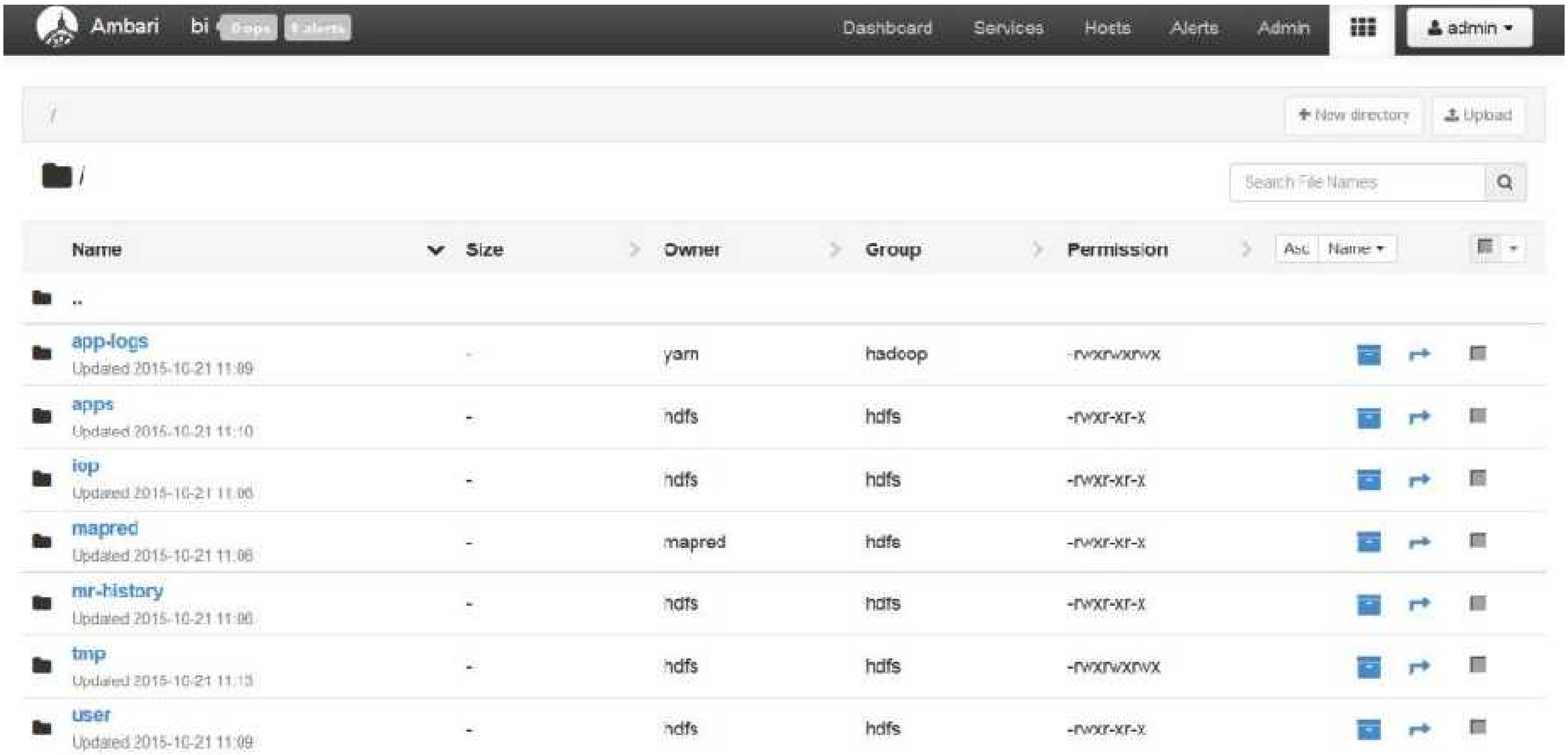
Services status



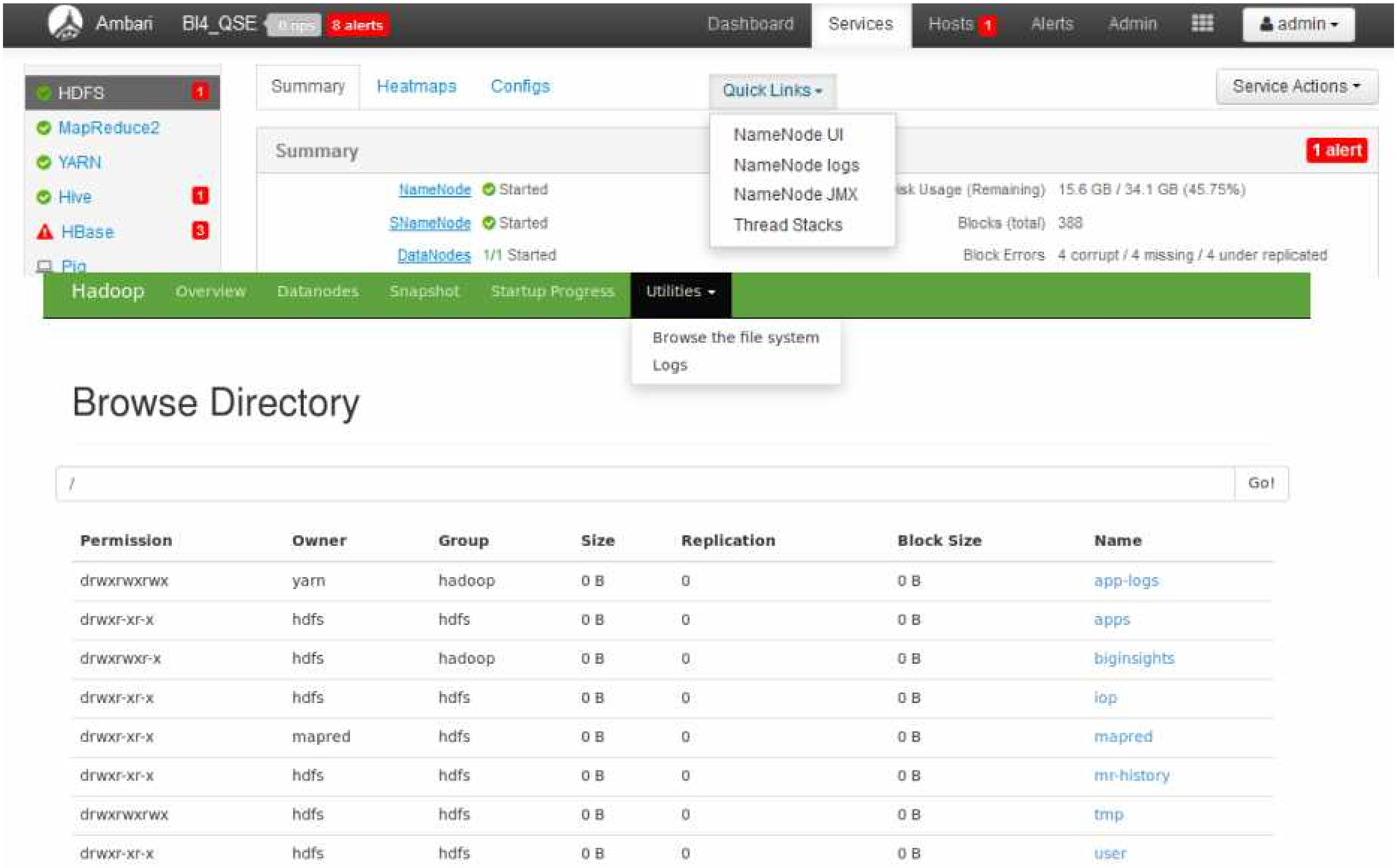
Starting / stopping components

- Not all components may need to be running
  - Stopping some can save resources

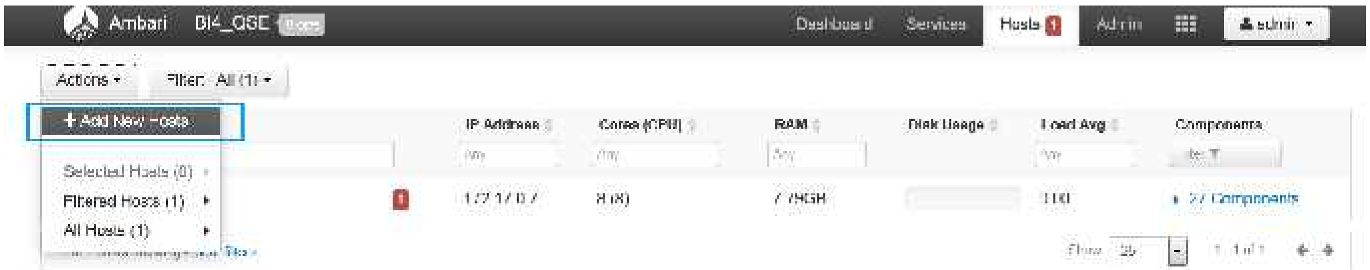
Working with directories and files

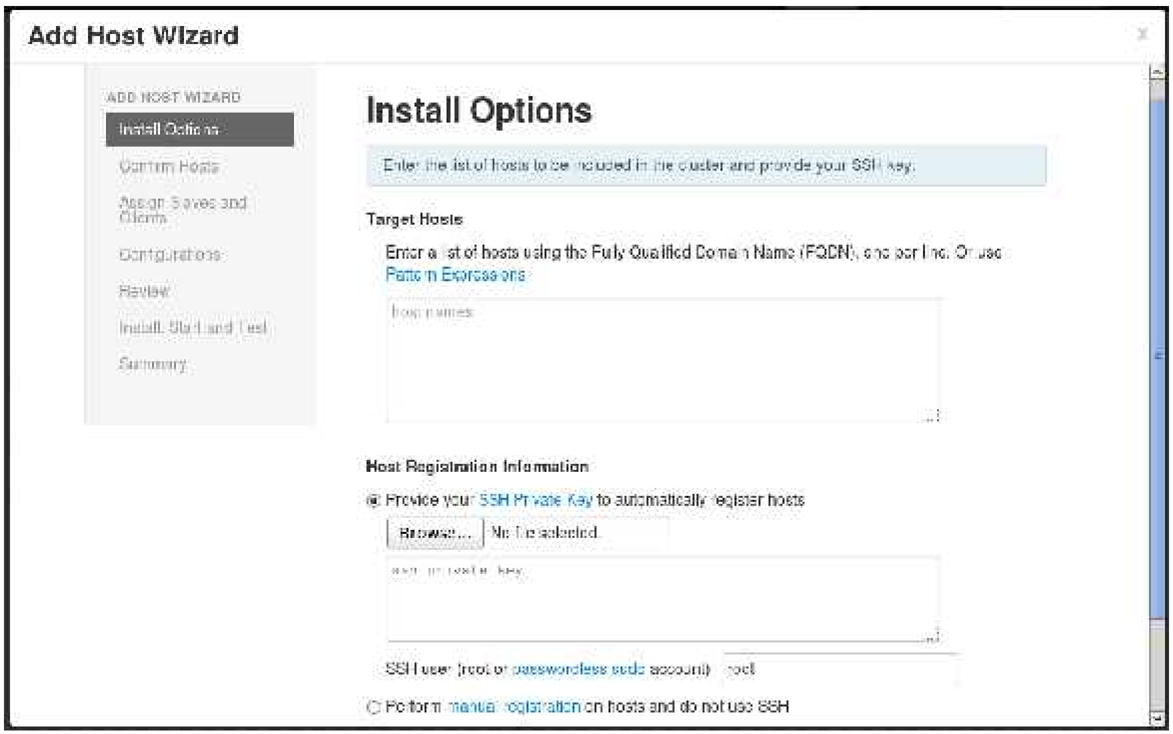


Working with directories and files



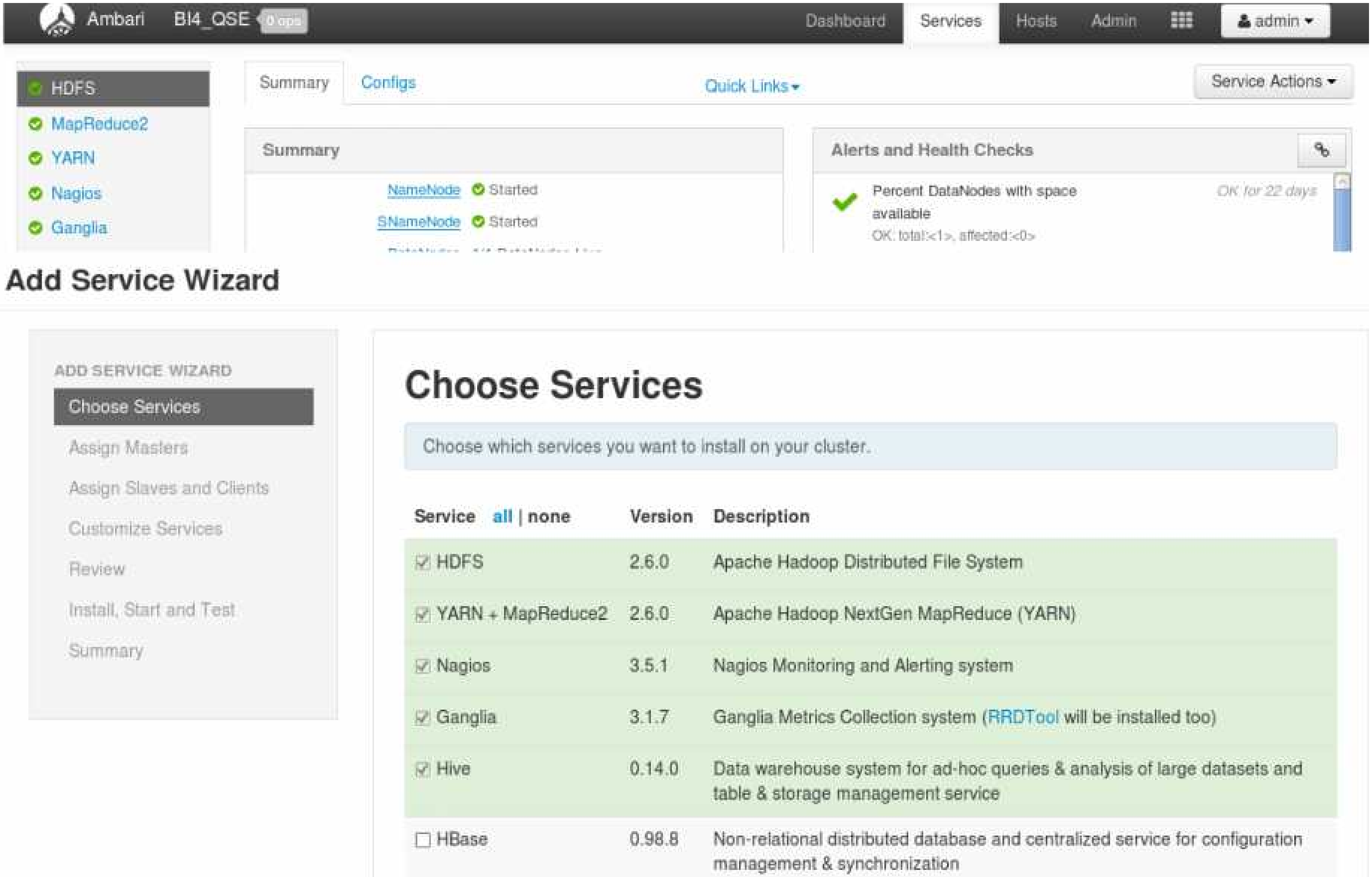
Using Web Console to add nodes





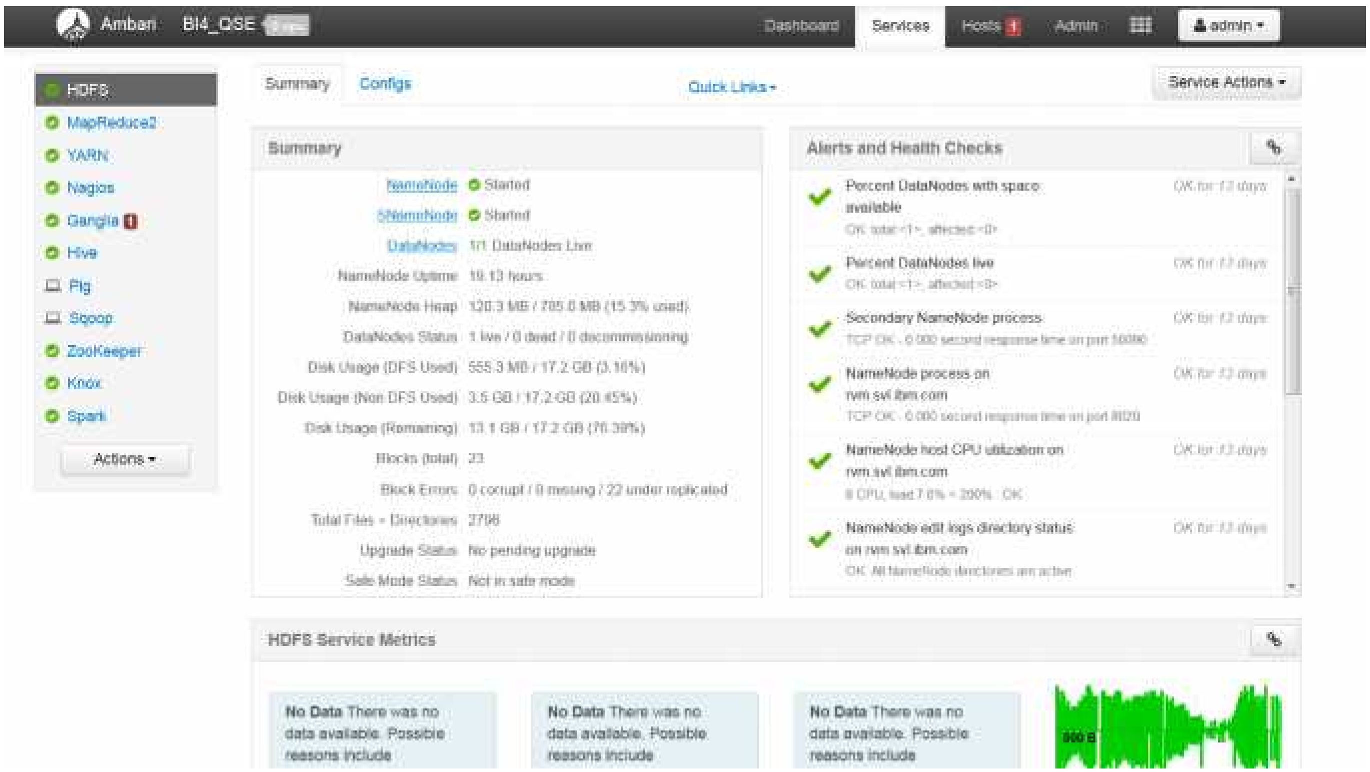
Using Web Console to remove nodes

Using Web Console to add services



Verifying cluster health

- Perform visual health check from Web console Services tab



MapReduce

- Processes huge datasets for certain kinds of distributable problems using a large number of nodes
- Map
  - Master node partitions the input into smaller sub-problems
  - Distributes the sub-problems to the worker nodes
- Reduce
  - Master node then takes the answers to all the sub-problems
  - Combines them in some way to get the output
- Allows for distributed processing of the map and reduce operations

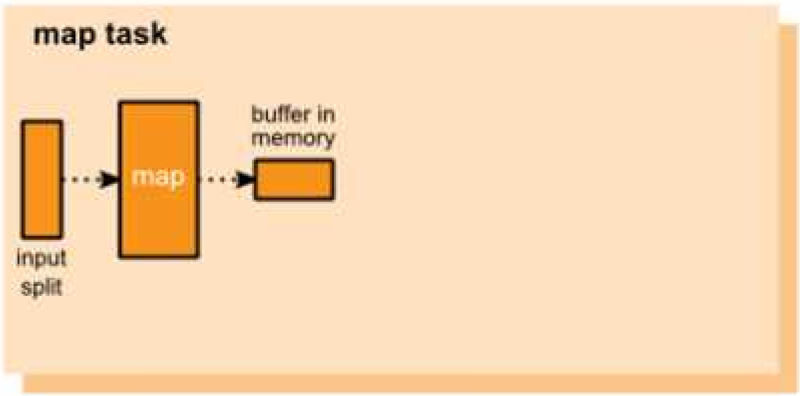
MapReduce framework

- Based on technology from Google
- Processes huge datasets for certain kinds of distributable problems using a large number of nodes
- A MapReduce program consists of map and reduce functions
- Allows for distributed processing of the map and reduce operations
  - Tasks run in parallel

MapReduce – Distributed Merge Sort Engine

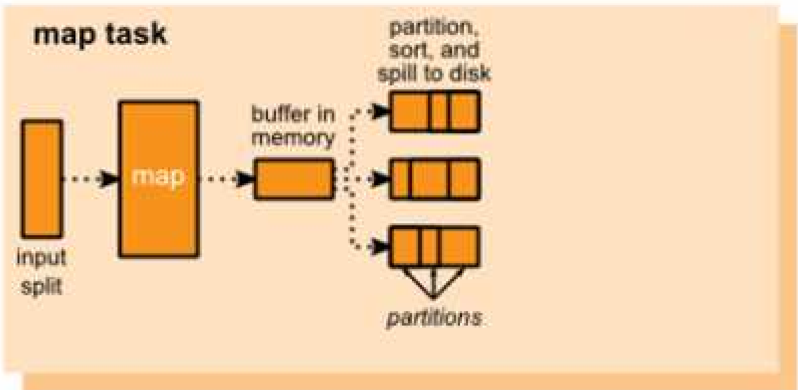
In this case, we will have a job with a single map step and a single reduce step. The first step is the map step. It takes a subset of the full dataset called an input split and applies to each row in the input split an operation that you have written, such as parsing each character string.

MapReduce – Distributed Merge Sort Engine



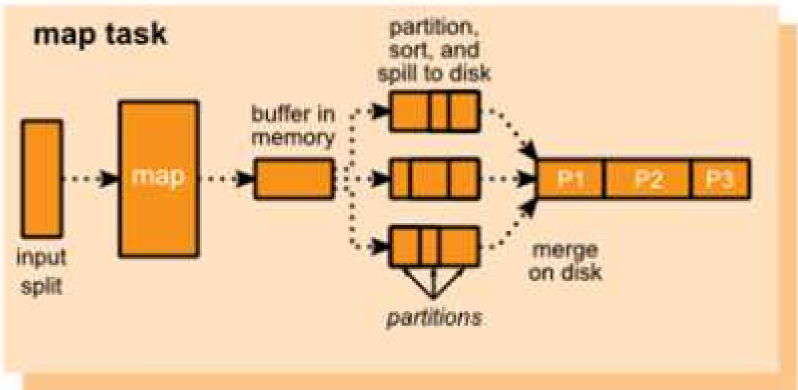
The output data is buffered in memory and spills to disk.

MapReduce – Distributed Merge Sort Engine



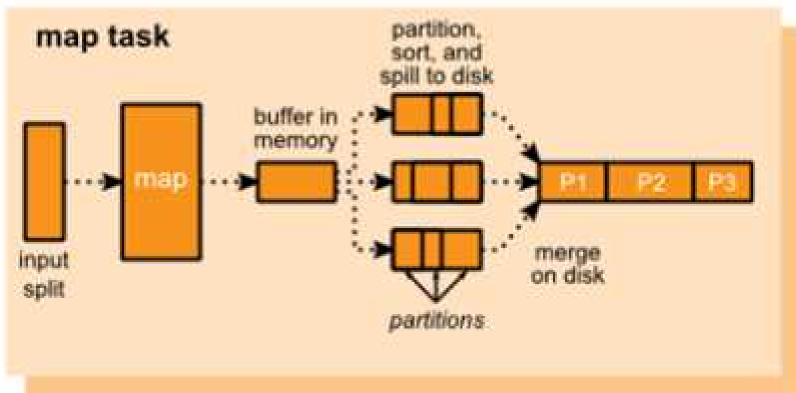
It is sorted and partitioned by key using the default partitioner.

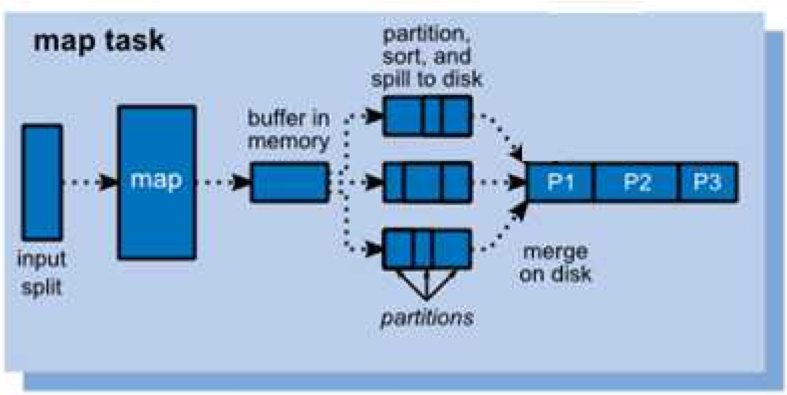
MapReduce – Distributed Merge Sort Engine



A merge sort sorts each partition.

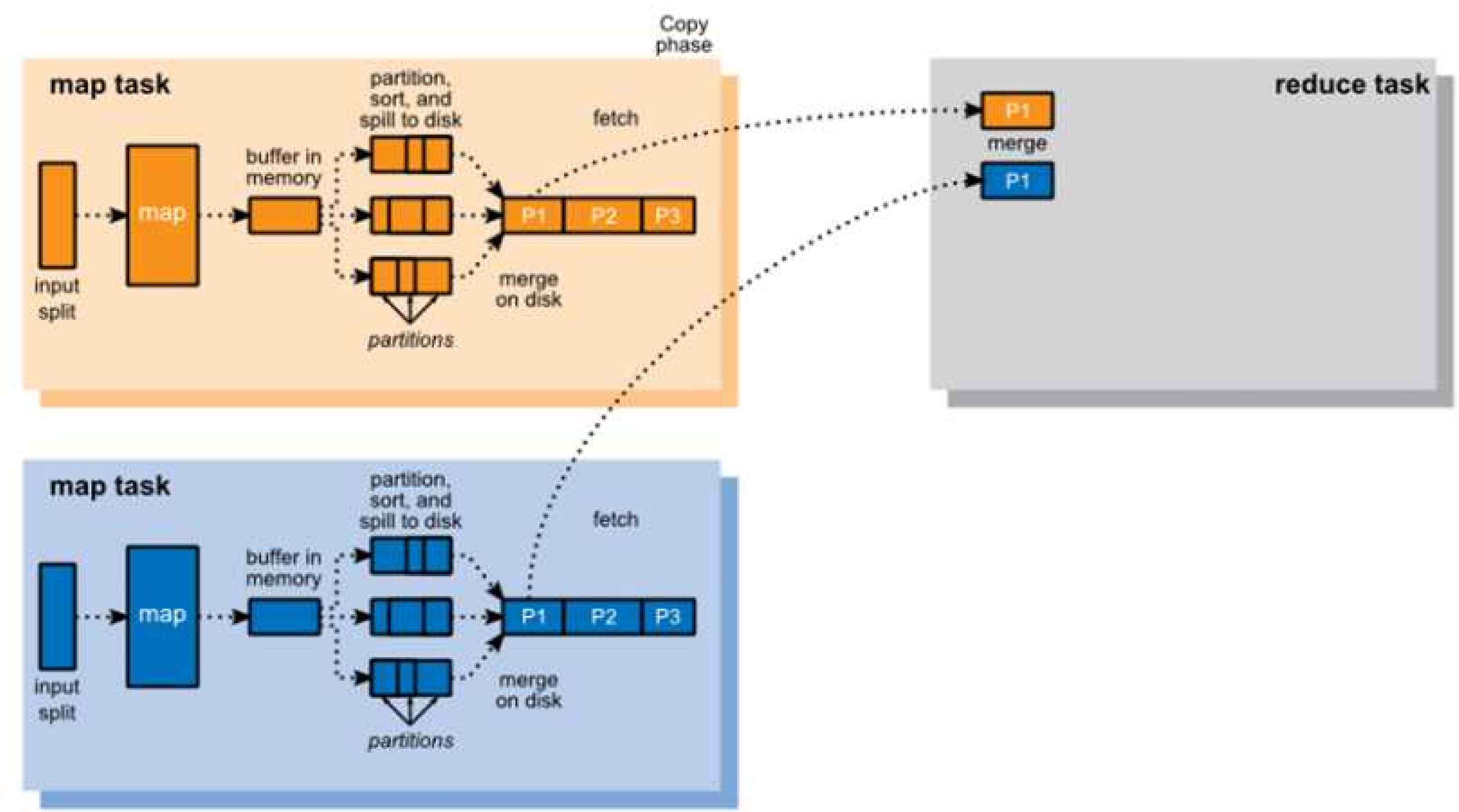
MapReduce – Distributed Merge Sort Engine





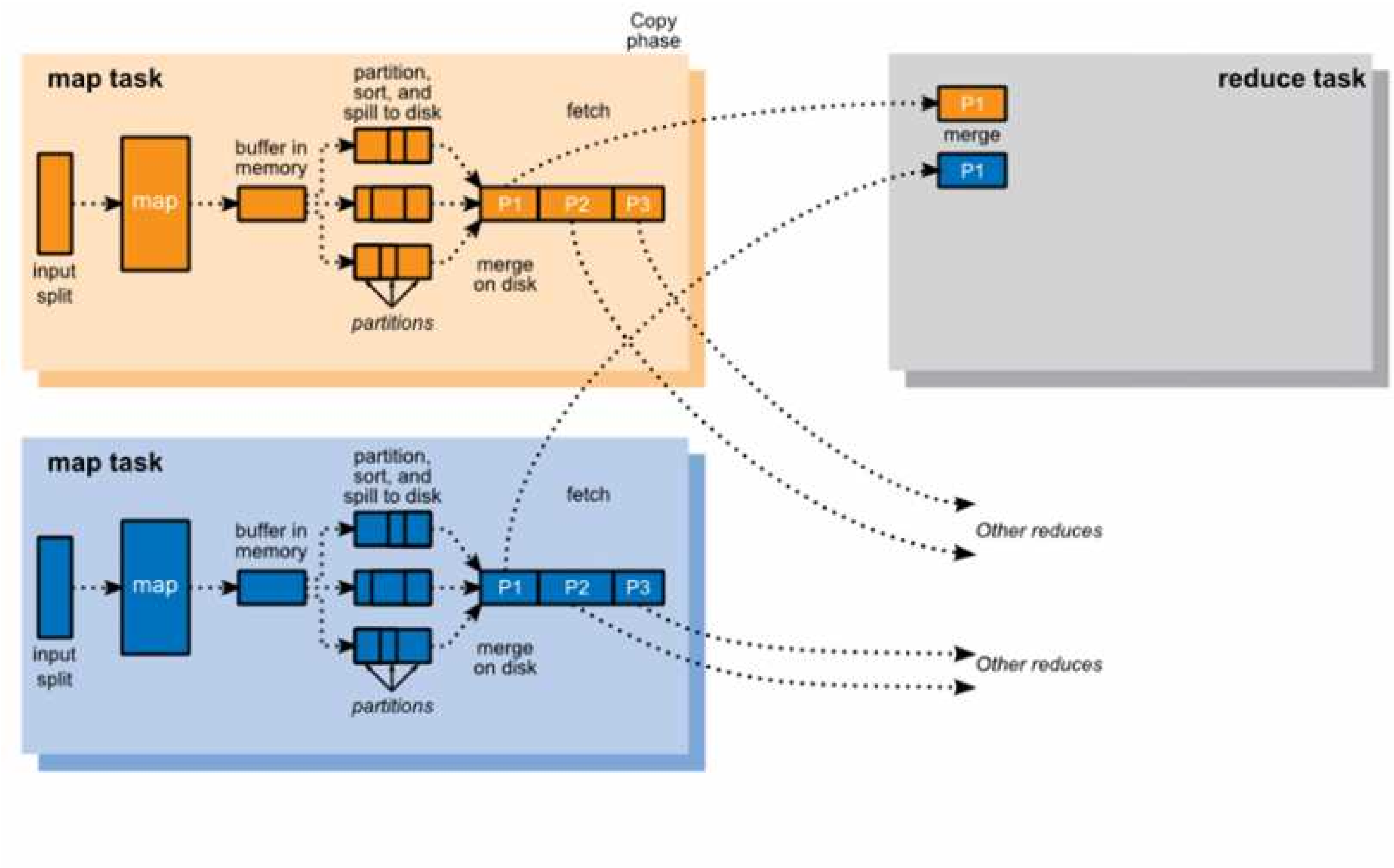
A merge sort sorts each partition.

MapReduce – Distributed Merge Sort Engine



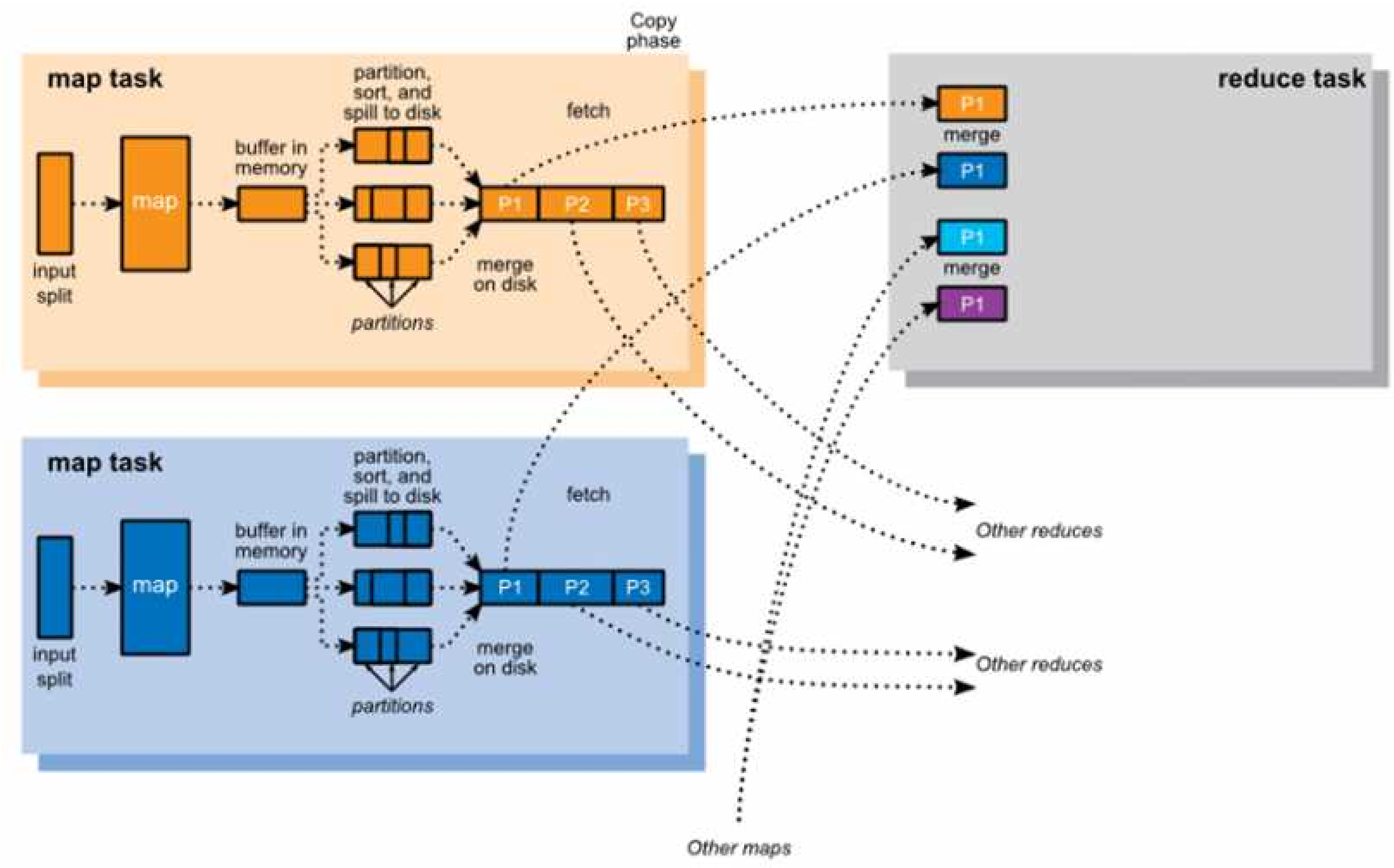
The partitions are shuffled among the reducers. For example, partition 1 goes to reducer 1. The second map task also sends its partition 1 to reducer 1.

MapReduce – Distributed Merge Sort Engine

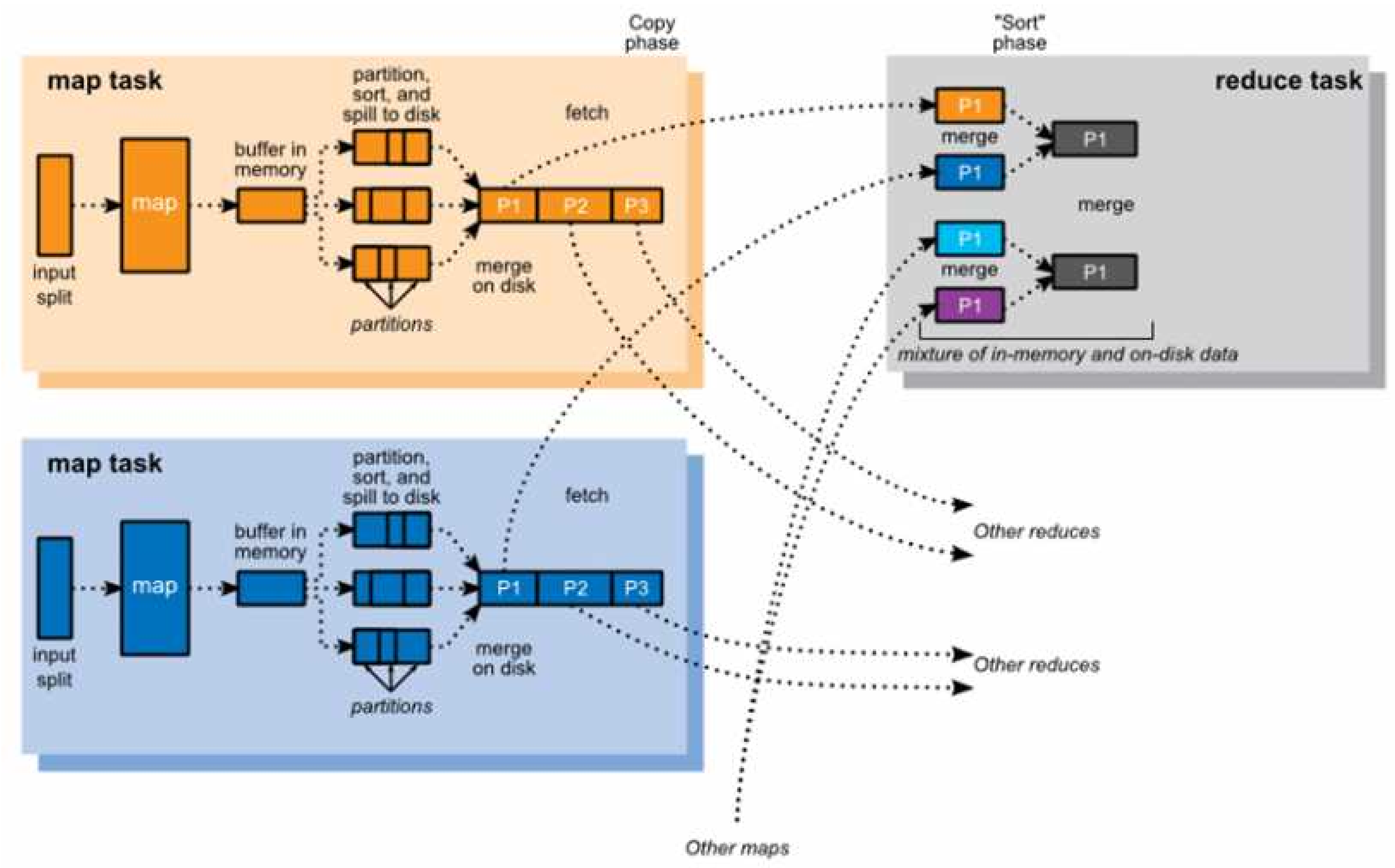


Partition 2 goes to another reducer.

MapReduce – Distributed Merge Sort Engine

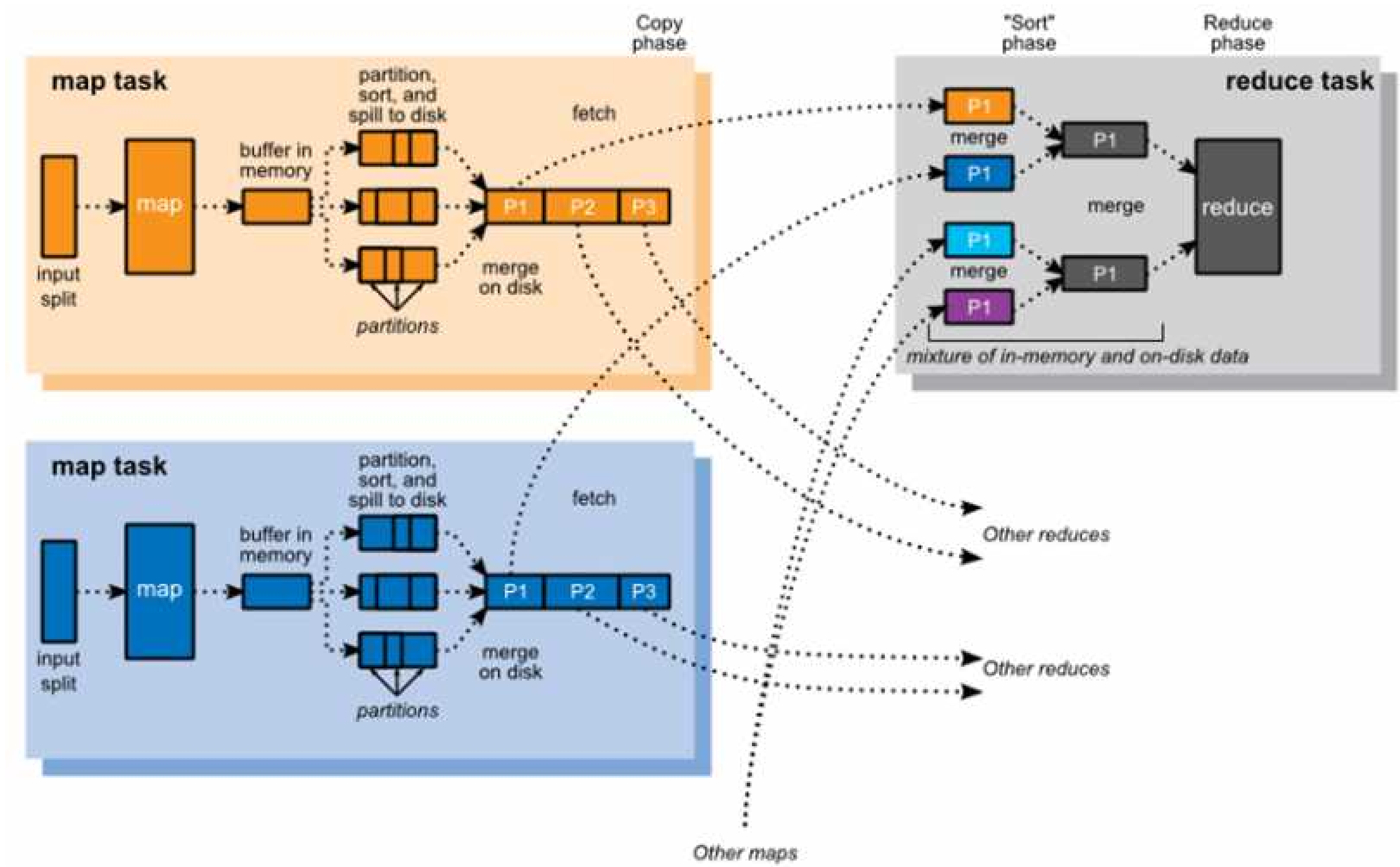


MapReduce – Distributed Merge Sort Engine



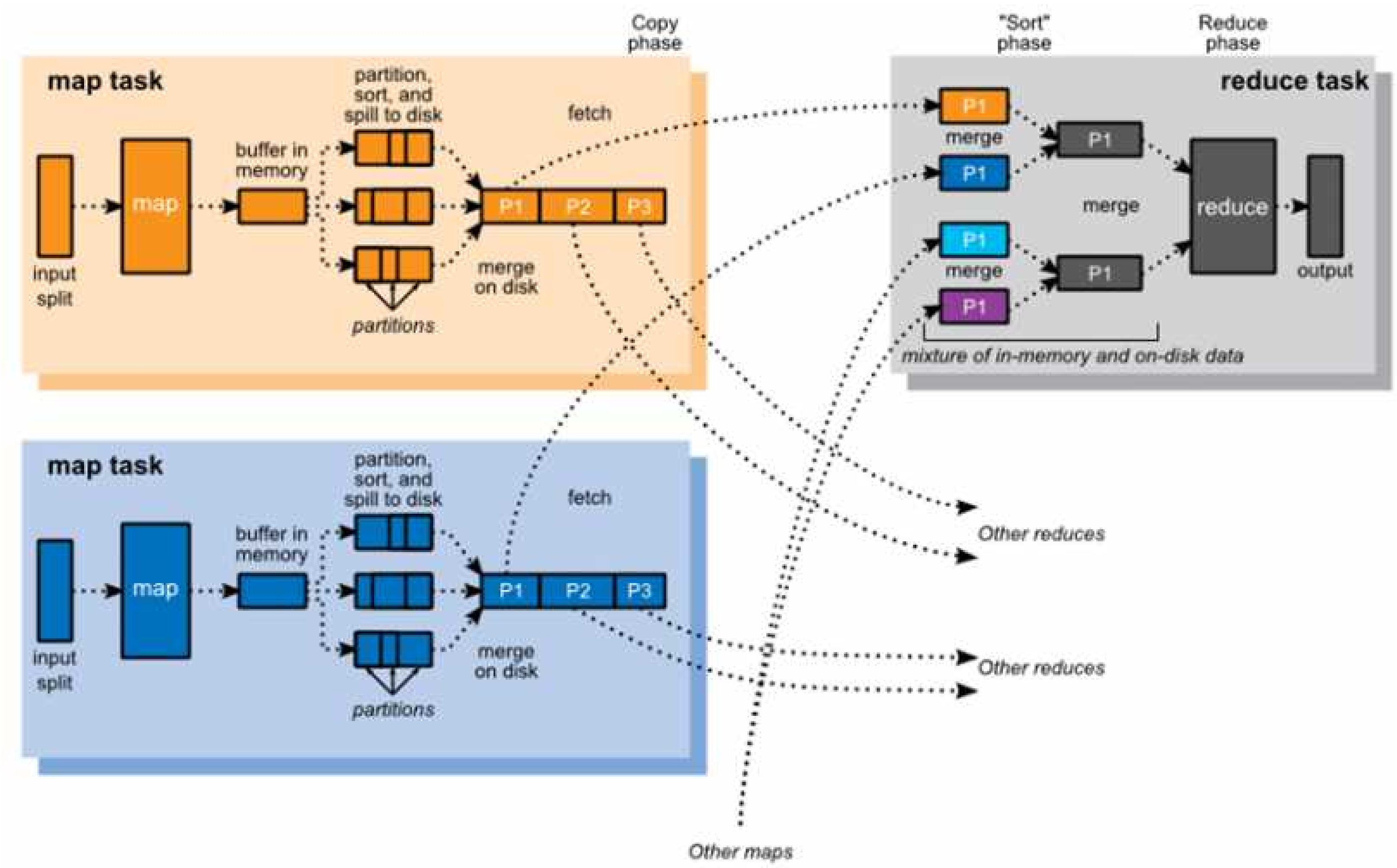
Additional map tasks would act in the same way

MapReduce – Distributed Merge Sort Engine



Each reducer does its own merge steps and executes the code of your reduce task.
For example, it could do a sum on the number of occurrences of a particular character string.

MapReduce – Distributed Merge Sort Engine



This produces sorted output at each reducer

Two Fundamental data types

- Key/value pairs
- Lists

	Input	Output
map	<k1, v1>	list(<k2, v2>)
reduce	<k2, list(v2)>	list(<k3, v3>)

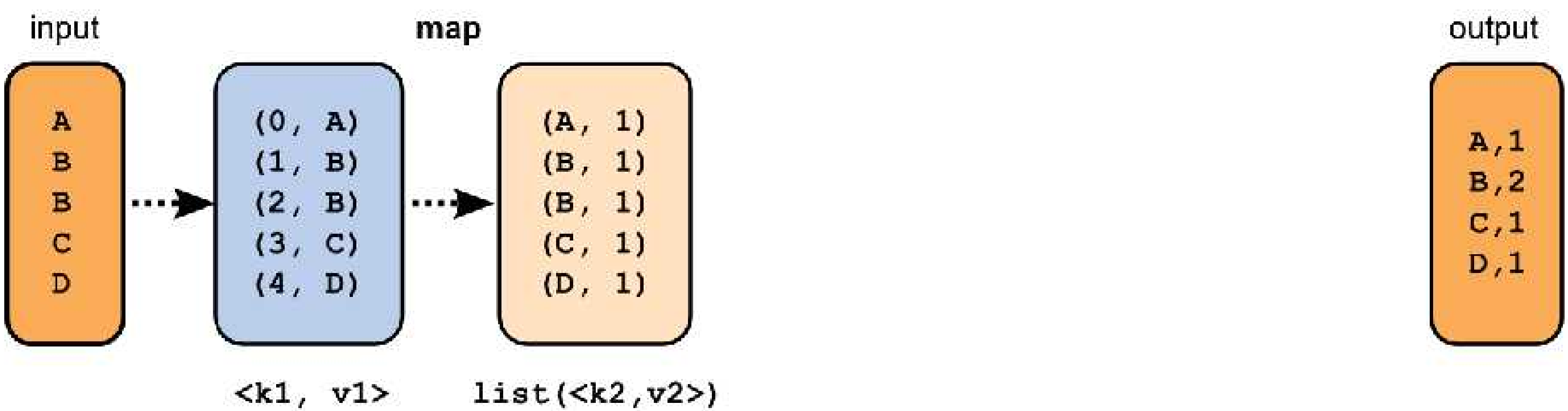
Simple data flow example

Say we want to transform the input on the left to the output on the right. On the left, we just have letters. On the right, we have counts of the number of occurrences of each letter in the input.

Simple data flow example

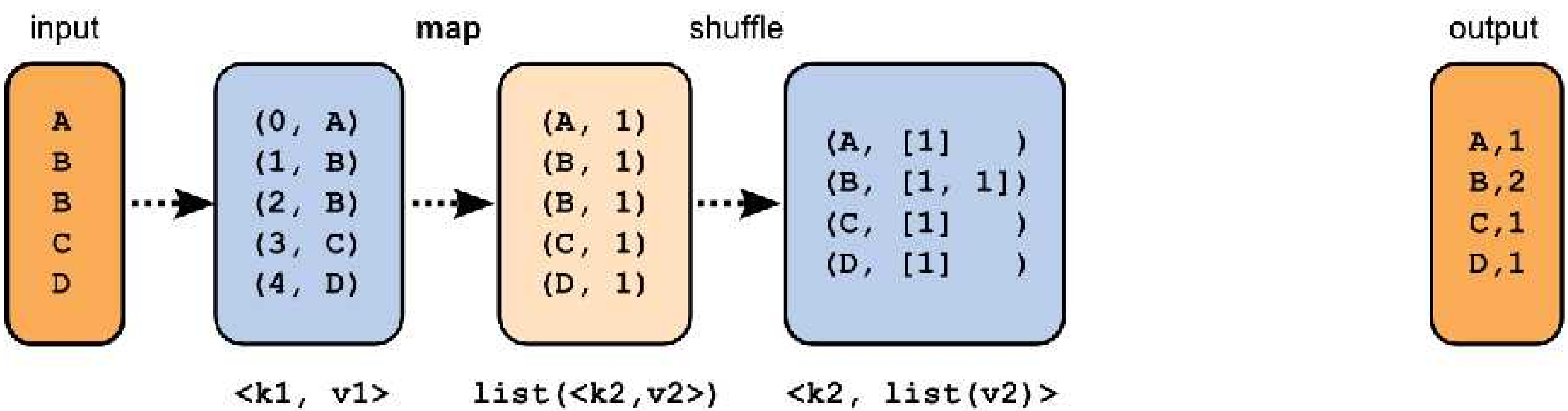
Hadoop does the first step for us. It turns the input data into key-value pairs and supplies its own key: an increasing sequence number.

Simple data flow example



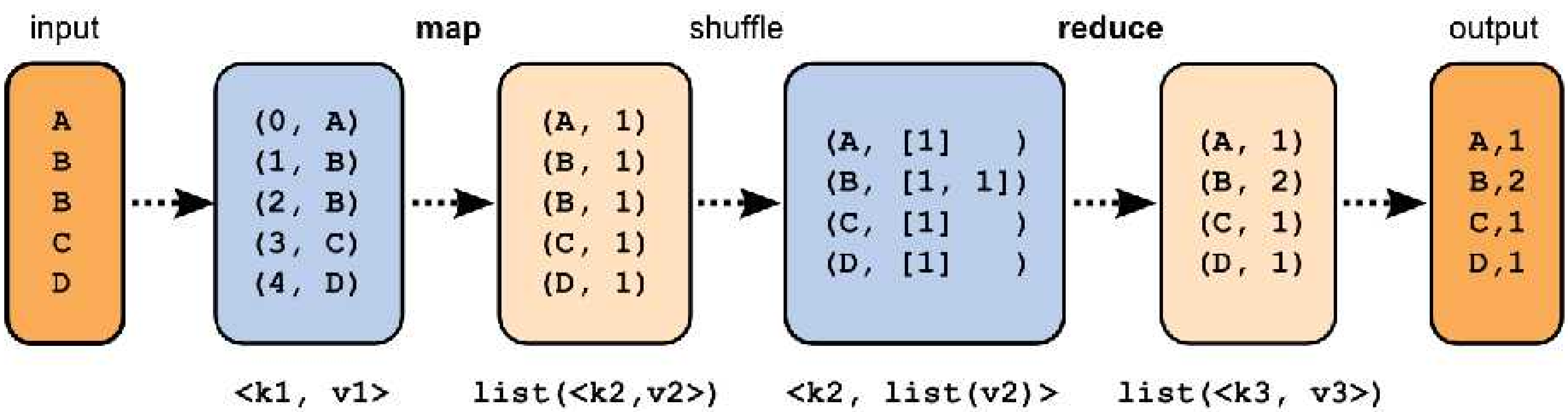
The function we write for the mapper needs to take these key-value pairs and produce something that the reduce step can use to count occurrences. The simplest solution is make each letter a key and make every value a 1.

Simple data flow example



The shuffle groups records having the same key together, so we see B now has two values, both 1, associated with it.

Simple data flow example



The reduce is simple: it just sums the values it is given to produce a sum for each key.

Languages - overview

- **Pig and Hive**
  - Similarities
    - All translate high-level languages to MapReduce jobs
    - All offer significant reductions in program size over Java
    - All provide points of extension to cover gaps in functionality
    - All provide interoperability with other languages
    - None support random reads/writes or low-latency queries

Languages - Pig

- Developed at Yahoo!
- Data flow language
- Can operate on complex, nested data structures
- Schema optional
- Relationally complete
- Turing complete when extended with Java UDFs

Languages - Hive

- Developed at Facebook
- Declarative language (SQL dialect)
- Schema non-optional but data can have many schemas
- Relationally complete
- Turing complete when extended with Java UDFs

Pig

- **Running Pig**
  - Script
    - pig scriptfile.pig
  - Grunt
    - pig (to launch command line tool)
  - Embedded
    - Call in to Pig from Java
  - Execution environments
    - Local
    - Distributed

Hive

- **Running Hive**
  - Hive Shell
    - Interactive - hive
    - Script - hive -f myscript
    - Inline - hive -e 'SELECT * FROM mytable'

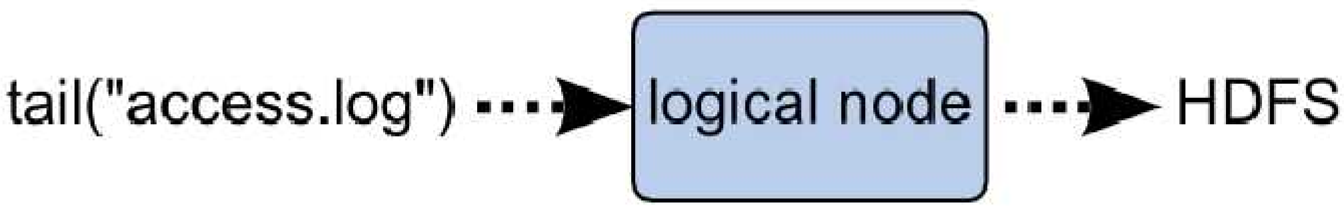
Data movement - overview

- **Flume**
  - A service for moving large amounts of data around a cluster soon after the data is produced
  - Primary use case
    - Gathering log files from every machine in a cluster
    - Transferring the data to a centralized persistent store
      - e.g. HDFS
- **Sqoop**
  - Transfers data between Hadoop and relational databases
  - Uses MapReduce to import and export the data

Flume architecture

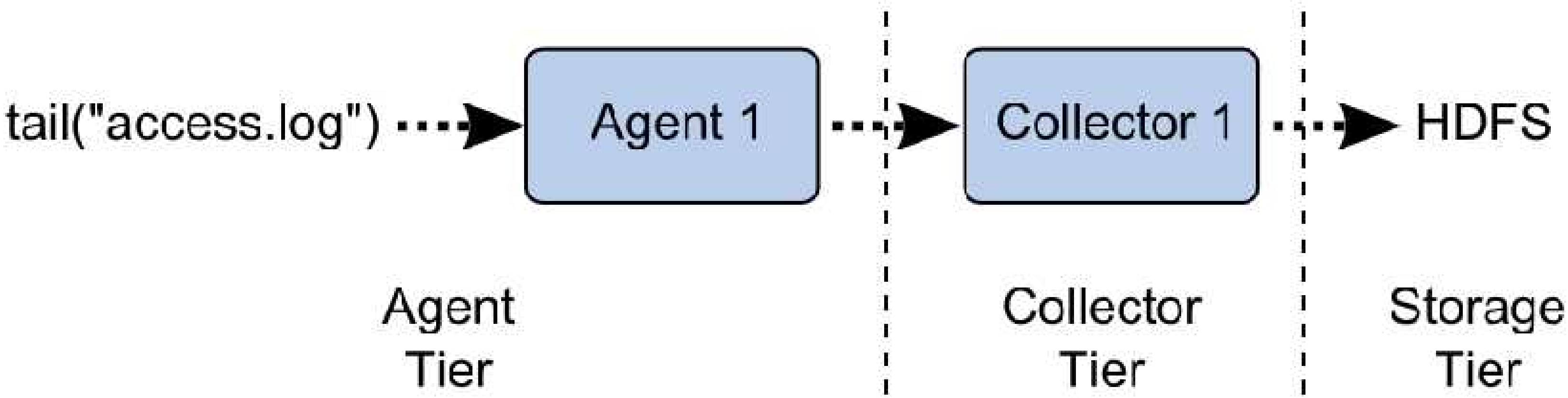
- Stream-oriented data flow
  - Chains of logical nodes

  - Example:

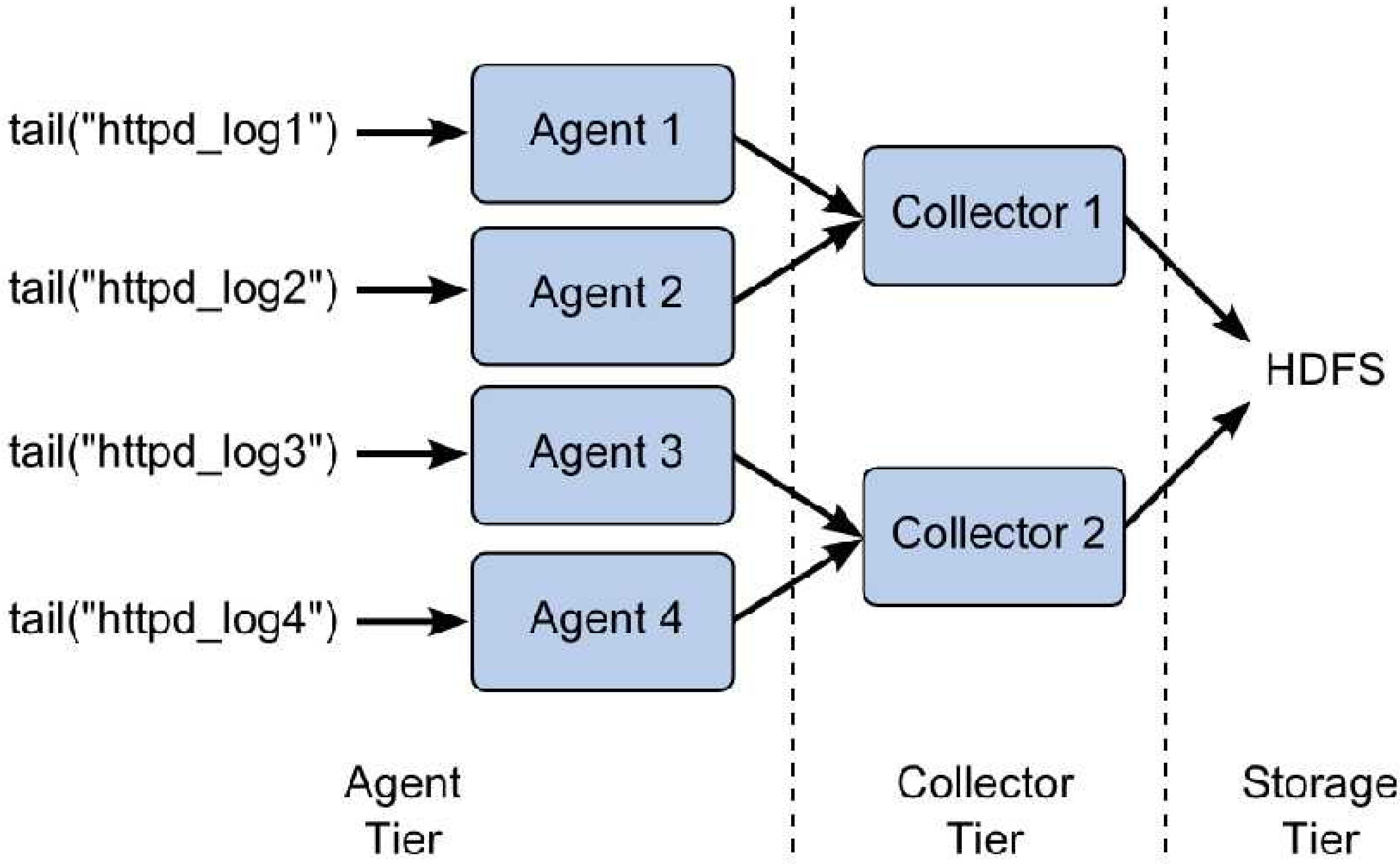


Flume architecture

- Tiers
  - Agent Tier
  - Collector Tier
  - Storage Tier

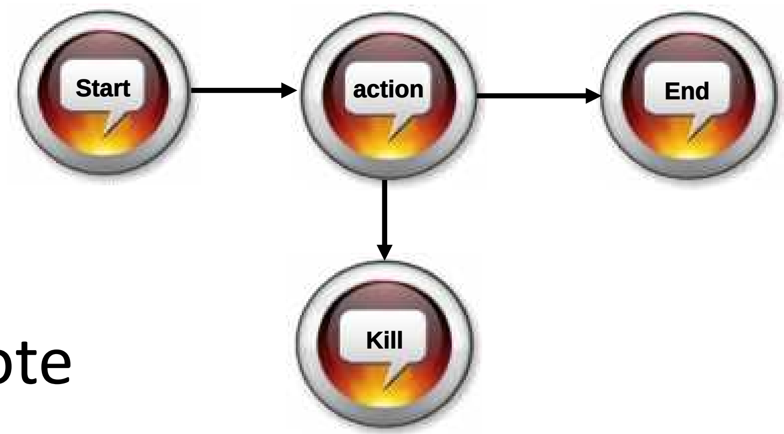


Flume architecture



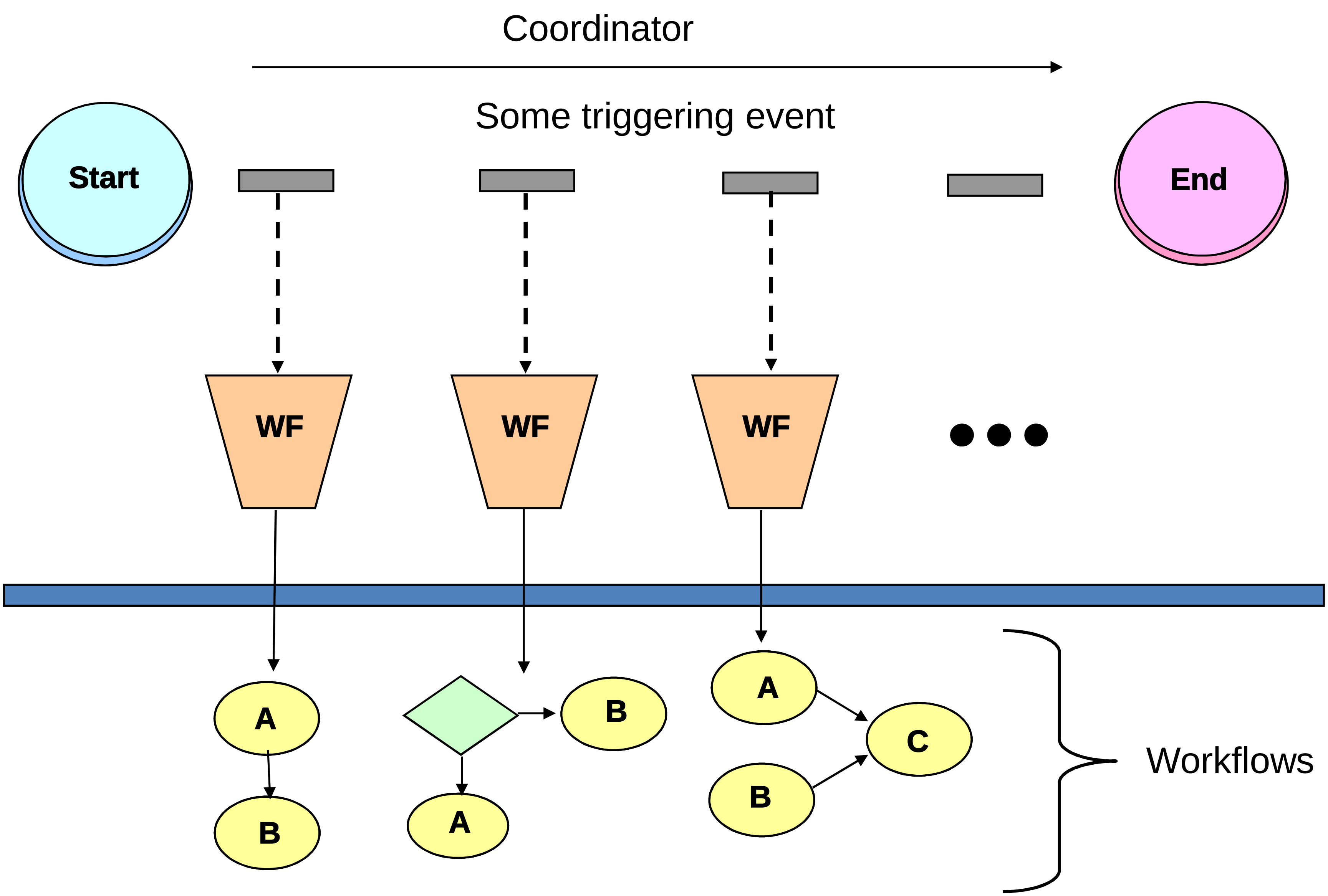
Oozie - workflows

- Workflows
  - Collections of actions arranged in a Direct Acyclic Graph (DAG)
    - There is a control dependency from one action to a second action
    - Second action cannot run until the first action completes
  - Definitions are written in hPDL
    - An XML Process Definition Language



- Workflow actions start jobs in remote systems
  - The remote systems callback Oozie to notify that the action has completed

Oozie - coordinator



Configuration files

- hadoop-env.sh Environment variables that are used in the scripts to run Hadoop.
- core-site.xml Configuration settings for Hadoop Core, such as I/O settings that are common to HDFS and MapReduce
- hdfs-site.xml Configuration settings for HDFS daemons: the name node, secondary name node, and the data nodes.
- mapred-site.xml Configuration settings for MapReduce daemons and jobtracker, and tasktrackers.
- masters A list of machines (one per line) that each run secondary NameNode
- slaves A list of machines (one per line) that each run data node and tasktracker
- hadoop-metrics.properties Properties for controlling how metrics are published in Hadoop.
- log4j.properties Properties for system logfiles, the NameNode audit log, and the task log for the tasktracker child process

BigInsights Configuration Directory: /usr/iop/current/hadoop-client/conf

hadoop-env.sh settings

- Most variables by default and not set
- Only export JAVA_HOME is required and should be set to java JDK
- HADOOP_HOME – Contains code & config files /usr/iop/current/hadoop-client
- HADOOP_LOG_DIR – Keeps logs /var/log/Hadoop/$USER
- HADOOP_HEAPSIZE – heap size used by JVM of each daemon
  - Can be overwritten for each daemon:
    - NameNode - HADOOP_NAMENODE_OPTS
    - DataNode - HADOOP_DATANODE_OPTS
    - Secondary NameNode - HADOOP_SECONDARYNAMENODE_OPTS
    - JobTracker - HADOOP_JOBTRACKER_OPTS
    - TaskTracker - HADOOP_TASKTRACKER_OPTS
- Other environment variables: HADOOP_CLASSPATH, HADOOP_PID_DIR

core-site.xml settings

fs.defaultFS	The name of the default file system. A URI whose scheme and authority determine the FileSystem implementation. The uri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class. The uri's authority is used to determine the host, port, etc. for a filesystem. Default: file:///
hadoop.tmp.dir	A base for other temporary directories. Default: /tmp/hadoop-${user.name}
fs.trash.interval	Number of minutes between trash checkpoints. If zero, the trash feature is disabled (default). When greater than zero erased files will be inserted in .trash in user’s home directory.
io.file.buffer.size	The size of buffer for use in sequence files. The size of this buffer should be a multiple of hardware page size (4096 on Intel x86), and it determines how much data is buffered during read and write operations.

core-site.xml settings (continue)

hadoop.rpc.socket.factory.class.default	Default SocketFactory to use. This parameter is expected to be formatted as package.FactoryClassName".
hadoop.rpc.socket.factory.class.ClientProtocol	SocketFactory to use to connect to a DFS. If null or empty, use hadoop.rpc.socket.class.default. This socket factory is also used by DFSClient to create sockets to DataNodes.

Recommendation:

Leave both of the above parameters empty and mark them as FINAL

hdfs-site.xml settings

dfs.datanode.data.dir	Determines where on the local filesystem a DFS data node should store its blocks.
dfs.namenode.name.dir	Determines where on the local filesystem the DFS namenode should store the name table.
dfs.blocksize	HDFS block size. Default is 64MB. **Recommendation:** Set block size to 128MB or as appropriate for your data.

hdfs-site.xml settings (continue)

dfs.namenode.handler.count	Number of threads the NameNode node will use to handle requests. Default: 10 **Recommendation:** Increase for larger cluster
dfs.replication	The number of time the file block should be replicated in HDFS. Default: 3 **Recommendation:** Set it to 1 when not on the cluster
dfs.hosts	Name of a file containing an approved list of hostnames to access the NameNode.
dfs.hosts.exclude	Name of a file containing a list of hostnames not allowed to access the NameNode
dfs.permissions.enabled	Enables/Disables unix-like permissions on HDFS. Enabling the permissions does usually make things harder to work with while its bringing limited advantages (its not so much for securing things but for prohibiting users to mistakenly mess up others user's data)

mapred-site.xml configuration

mapreduce.jobtracker.hosts	Names a file that contains the list of nodes that may connect to the jobtracker. If the value is empty, all hosts are permitted.
mapreduce.jobtracker.hosts.exclude	Names a file that contains the list of hosts that should be excluded by the jobtracker. If the value is empty, no hosts are excluded.
mapreduce.job.maxtaskfailures.per.tracker	The number of task-failures on a tasktracker of a given job after which new tasks of that job aren't assigned to it. Default is 3
mapreduce.jobtracker.tasktracker.maxblacklists	The number of blacklists for a taskTracker by various jobs after which the task tracker could be blacklisted across all jobs. The tracker will be given a tasks later (after a day). The tracker will become a healthy tracker after a restart. Default is 4.

mapred-site.xml configuration (continue)

mapreduce.job.reduces	The default number of reduce tasks per job. Typically set to 99% of the cluster's reduce capacity, so that if a node fails the reduces can still be executed in a single wave. Ignored when mapred.job.tracker is "local". Default: 1. Recommendation: set it to 90%
mapreduce.map.speculative	If true, then multiple instances of some map tasks may be executed in parallel. Default: true.
mapreduce.reduce.speculative	If true, then multiple instances of some reduce tasks may be executed in parallel. Default: true. Recommended: false.
mapreduce.tasktracker.map.tasks.maximum	The maximum number of map tasks that will be run simultaneously by a task tracker. Default: 2. Recommendations: set relevant to number of CPUs and amount of memory on each data node.
mapreduce.tasktracker.reduce.tasks.maximum	The maximum number of reduce tasks that will be run simultaneously by a task tracker. Default: 2. Recommendations: set relevant to number of CPUs and amount of memory on each data node.

mapred-site configuration (continue)

mapreduce.jobtracker.taskscheduler	The class responsible for scheduling the tasks. Default points to FIFO scheduler. Recommendation: Use Job Queue Task - org.apache.hadoop.mapred.JobQueueTaskScheduler
mapreduce.jobtracker.restart.recover	Recover failed job when JobTracker restarts. For production clusters recommended to be set to TRUE
mapreduce.cluster.local.dir	The local directory where MapReduce stores intermediate data files. May be a comma-separated list of directories on different devices in order to spread disk i/o. Directories that do not exist are ignored. Default: ${hadoop.tmp.dir}/mapred/local