Hadoop Core components and Daemons

2 components of Hadoop

1.HDFS(storage)

2.YARN/MRv2(processing)

Daemons

Namenode
Datanode

ResourceManger
NodeManger

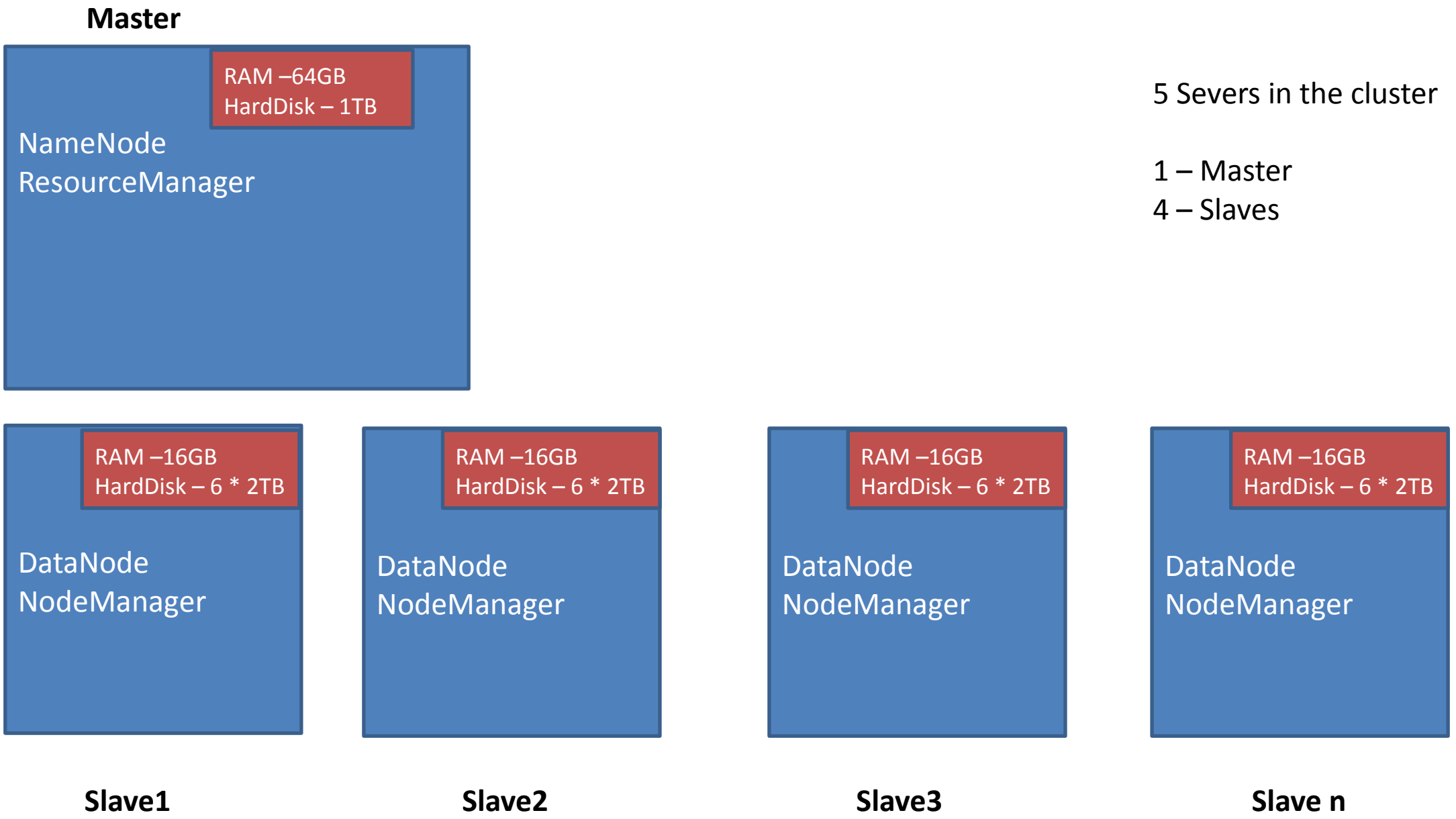
Hadoop Core components and Daemons

2 components of Hadoop
Master-Slave

1.HDFS(storage)
Namenode(Master)
Datanode(Slave)

2.YARN(processing)
ResourceManger(Master)
NodeManager(Slave)

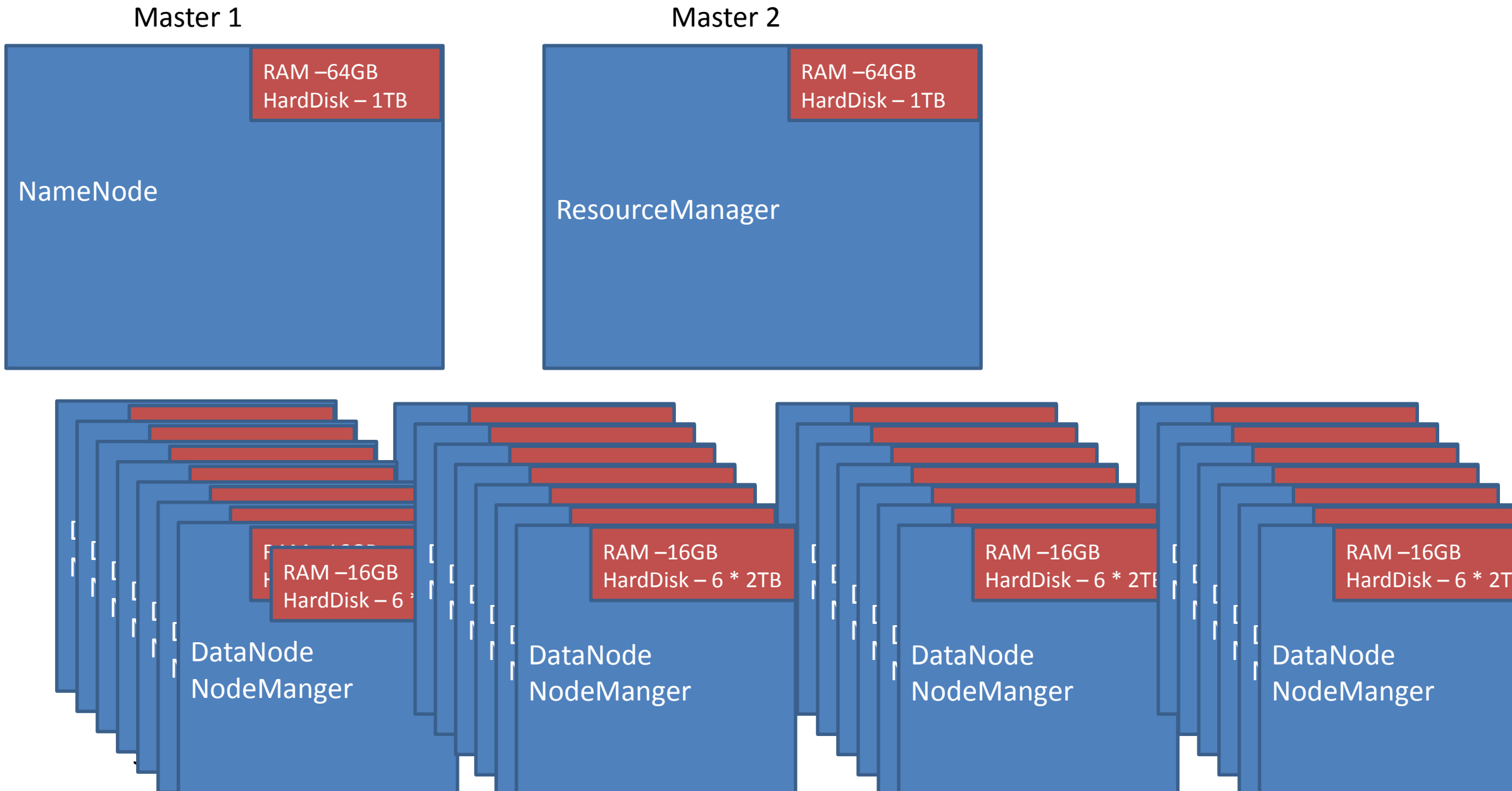
Simple Hadoop cluster with Daemons



5 Severs in the cluster

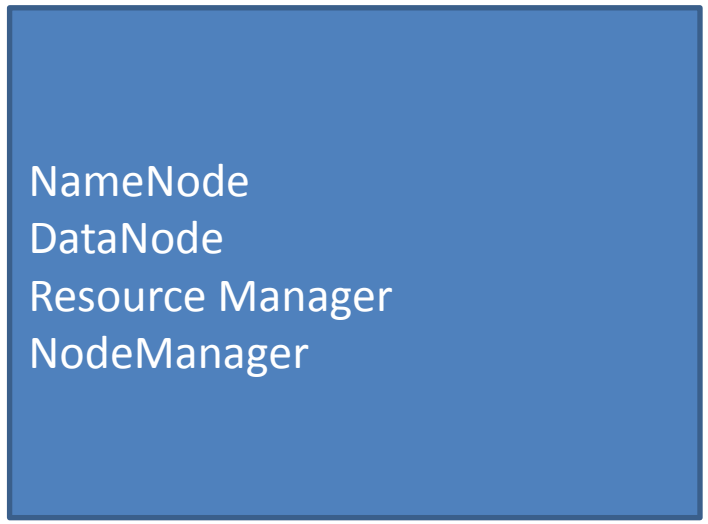
1 – Master
4 – Slaves

Simple cluster with Hadoop Daemons



Pseudo Distributed Mode

Single Server



File Blocks

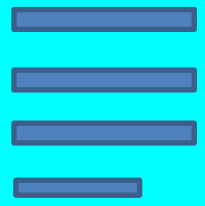
By Default Block cutoff size is 128 MB

200mb – abc.txt

128mb – Block 1
72mb -- Block 2

500mb – weblog.dat

128mb – Block 1
128mb -- Block 2
128mb – Block 3
116mb – Block 4



Files with less than 128MB just takes one block of exact size.

100mb -- books.xml

Block1

100 kb – flower.jpg

Block1

File Blocks

First of all, if the block size were too small it would overwhelm the metadata server with too many blocks to track.

Second, HDFS is designed to enable high throughput so that the parallel processing of these large data sets happens as quickly as possible.

On one hand, the block size needs to be large enough to warrant the resources dedicated to an individual unit of data processing (for instance, a map or reduce task)

You want to find a balance where each task is able to process a reasonable amount of data while still getting the benefits of parallelism.

The smaller the block size, the more tasks you get, the more scheduling activity occurs.

Data Storage in Slaves with replication

Master 1

NameNode

200mb – Sample.txt

128mb – Block 1

b1r1

b1r2

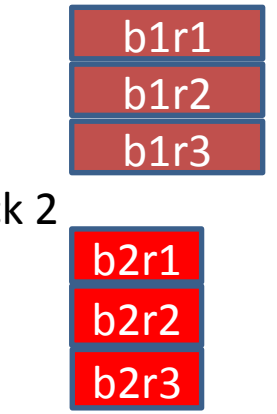
b1r3

72mb -- Block 2

b2r1

b2r2

b2r3



Namespace

b1
r1
sl
b1
r2
s2
b1
r3
s4

DataNode

b1r1

b2r3

Slave1

DataNode

b1r2

Slave2

DataNode

b2r2

Slave3

DataNode

b1r3

b2r1

Slave4

MapReduce

Master 1

Resource Manager

200mb – Sample.txt

128mb – Block 1

b1r1

b1r2

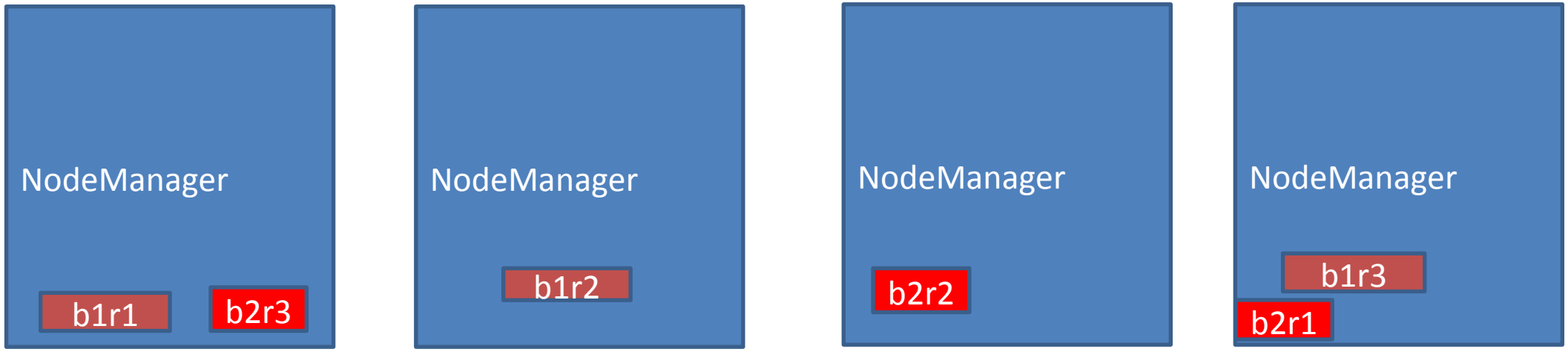
b1r3

72mb -- Block 2

b2r1

b2r2

b2r3



Slave1

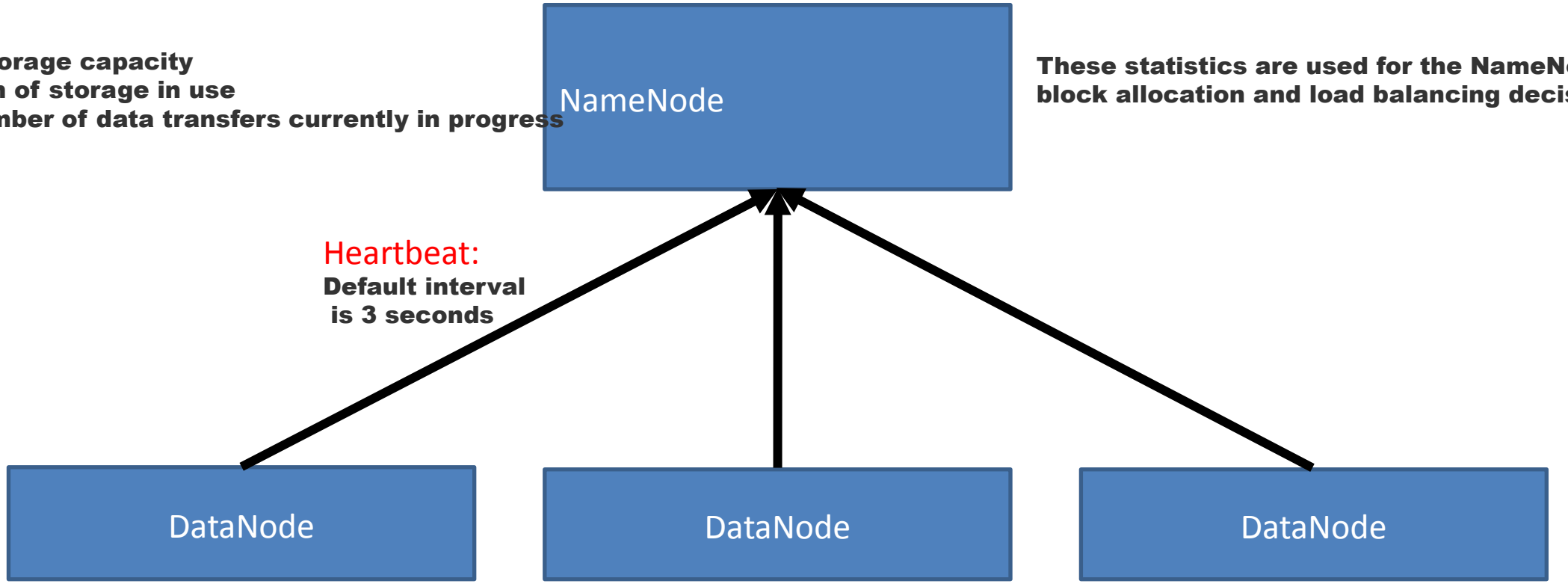
Slave2

Slave3

Slave4

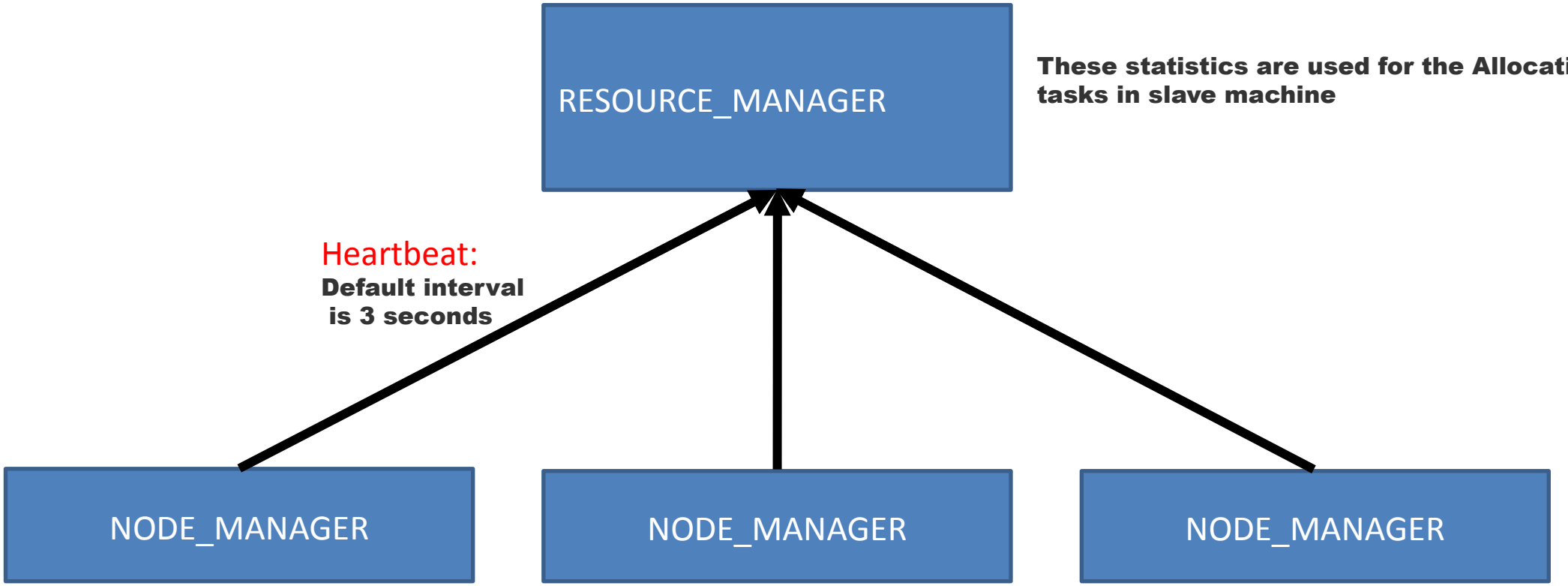
Heartbeat: **Confirm that the DataNode is operating and the block replicas it hosts are available**

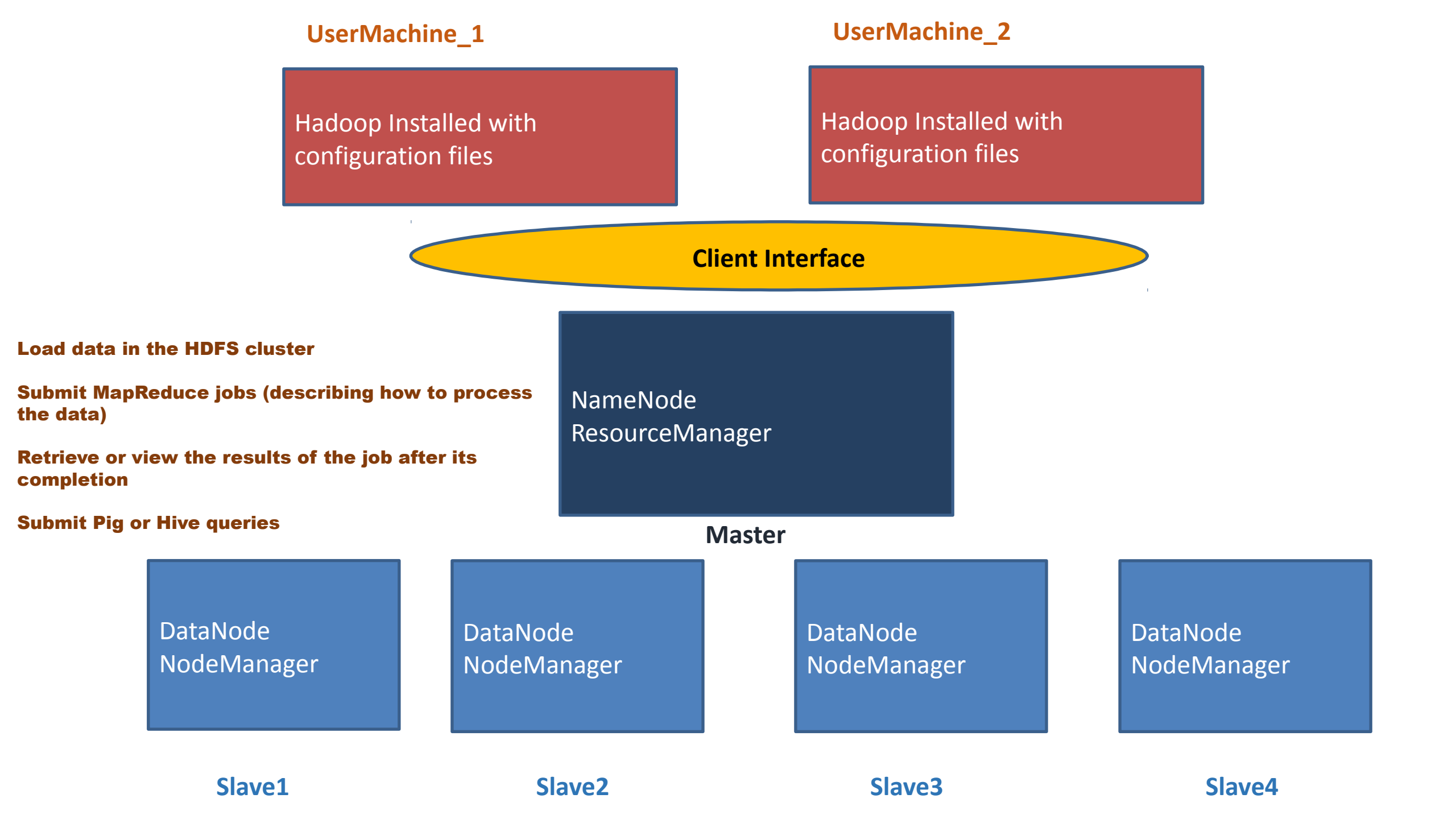
- **Total storage capacity**
- **Fraction of storage in use**
- **The number of data transfers currently in progress**



These statistics are used for the NameNode's block allocation and load balancing decisions.

Heartbeat: **Confirm that the NODEManager is operating**

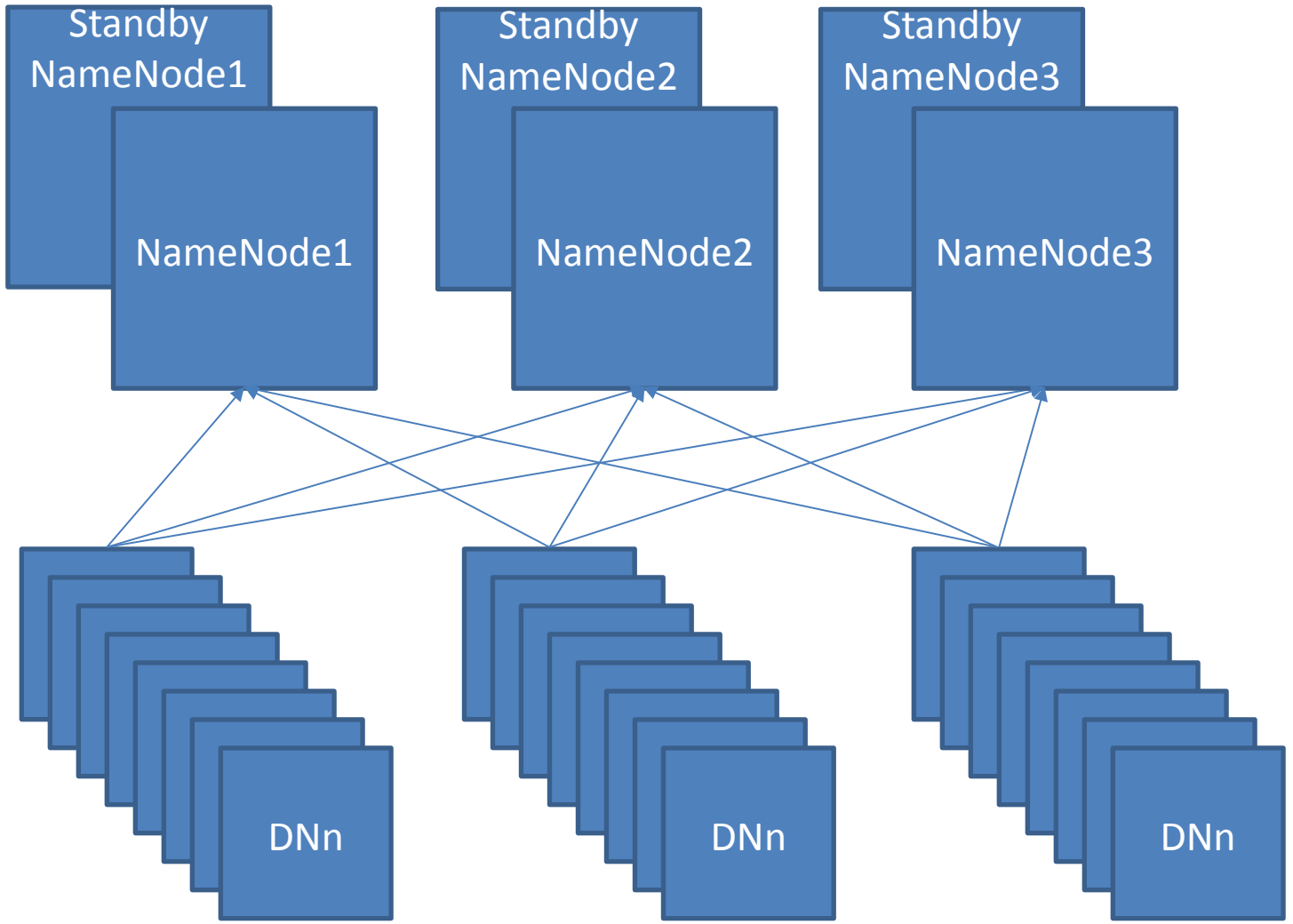



UserMachine_1
UserMachine_2
Hadoop Installed with configuration files
Hadoop Installed with configuration files
Client Interface
Load data in the HDFS cluster
Submit MapReduce jobs (describing how to process the data)
Retrieve or view the results of the job after its completion
Submit Pig or Hive queries
NameNode
ResourceManager
Master
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
Slave1
Slave2
Slave3
Slave4

Client of HDFS uses a specialize plugin called viewFS to view the logical, global namespace as a Single Entity

ViewFS Map

/ dev => NN1
/hbase => NN2
/user => NN3



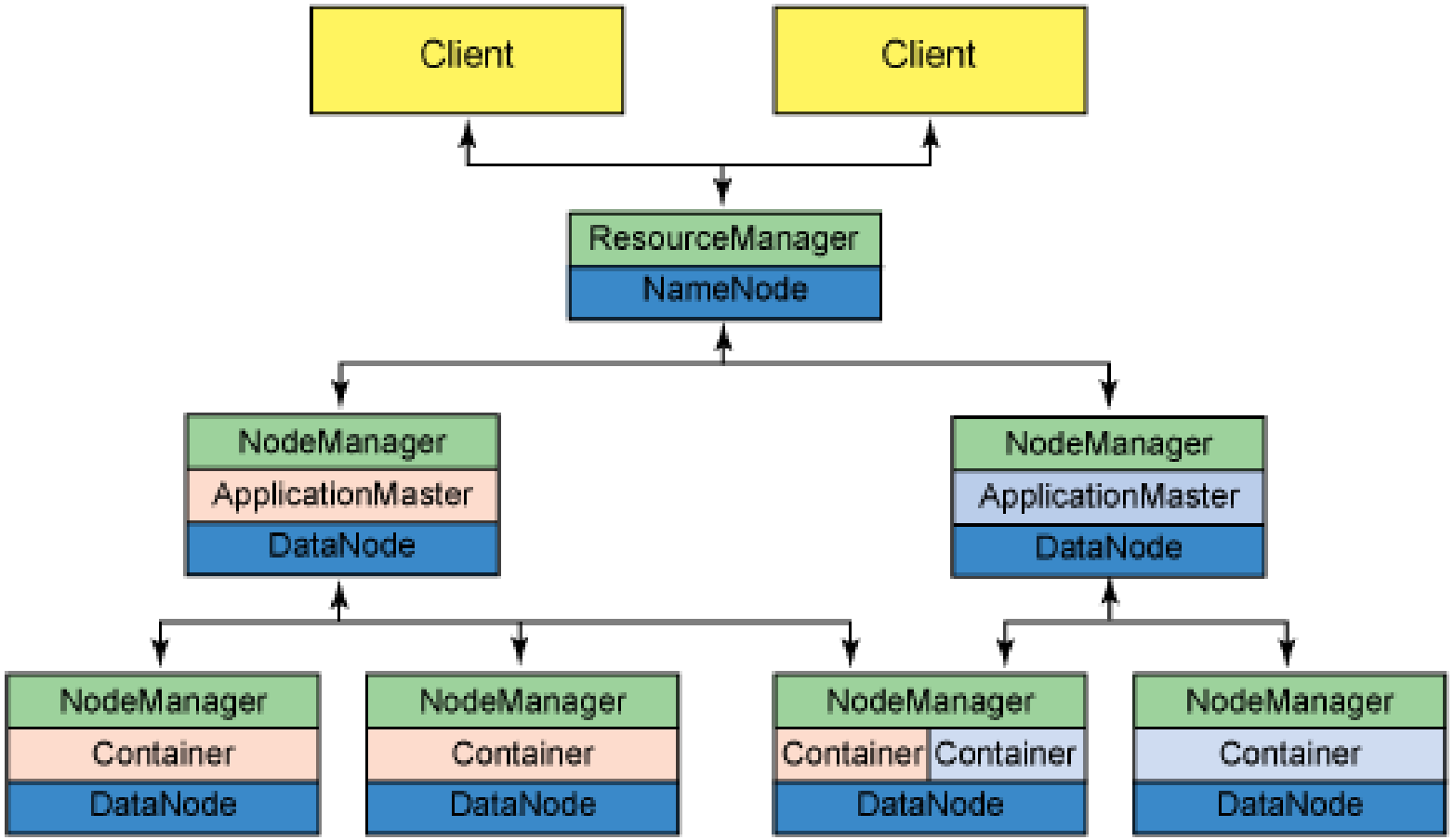
```
<configuration>
<property>
<name>dfs.nameservices</name>
<value>dev,hbase,user</value>
</property>

<property>
<name>dfs.namenode.rpc-address.dev</name>
<value>nn-host1:rpc-port</value>
</property>

<property>
<name>dfs.namenode.rpc-address.hbase</name>
<value>nn-host2:rpc-port</value>
</property>

<property>
<name>dfs.namenode.rpc-address.user</name>
<value>nn-host3:rpc-port</value>
</property>
....
Other common configuration
```

Architecture of YARN

ResourceManager governs an entire cluster and manages the assignment of applications to underlying compute resources.

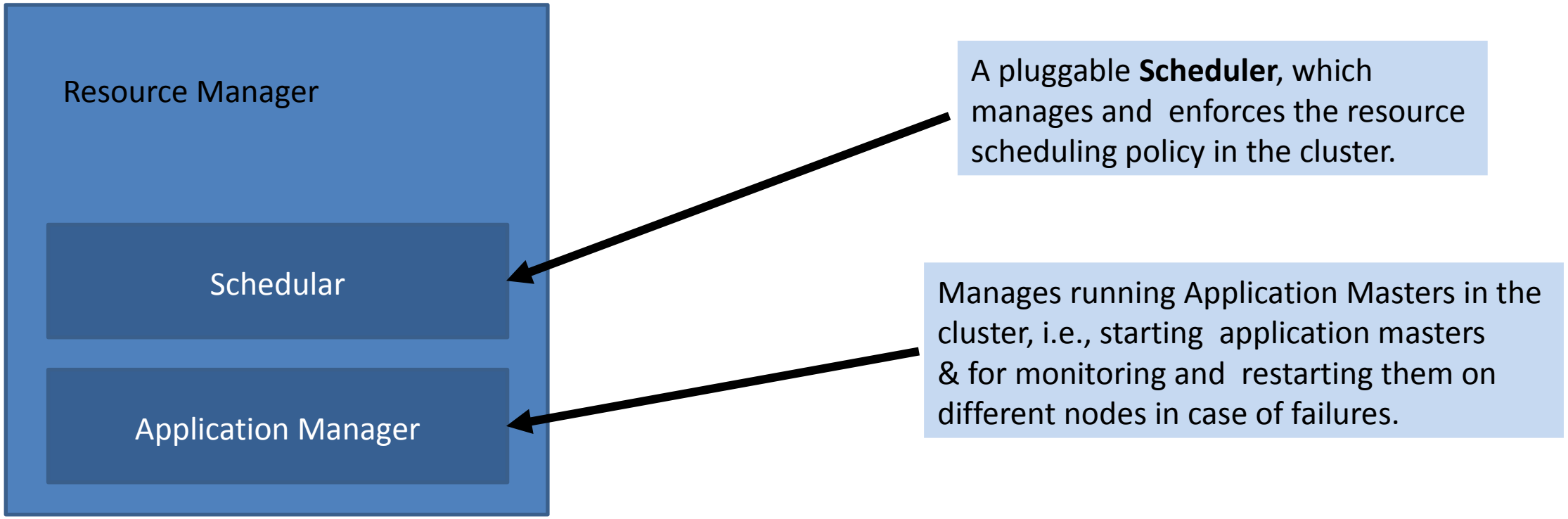
ApplicationMaster manages each instance of an application that runs within YARN.

NodeManager provides per-node services within the cluster, from overseeing the management of a container over its life cycle to monitoring resources and tracking the health of its node.



Resource Manager

There is a single Resource Manager, which has two main services:



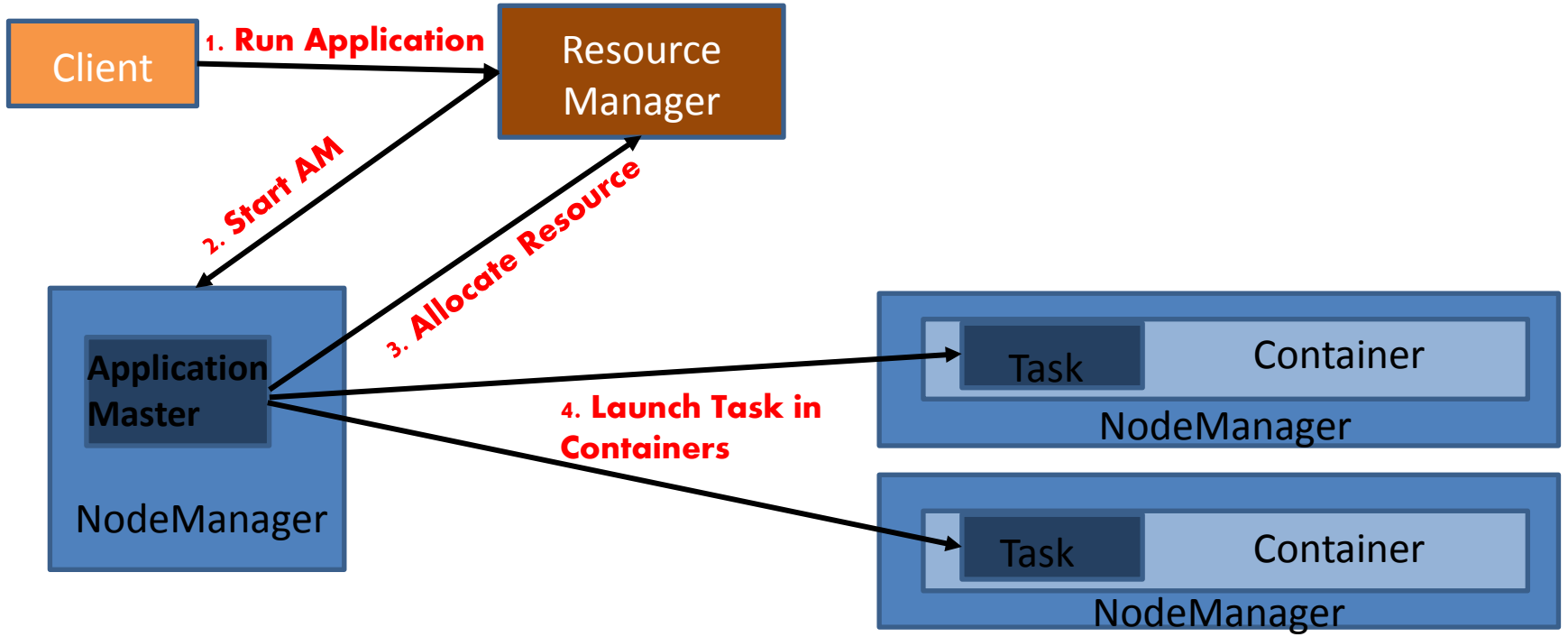
Application Submission in YARN

1. Application Submission Client submits an Application to the YARN Resource Manager. The client needs to provide sufficient information to the ResourceManager in order to launch ApplicationMaster

2. YARN ResourceManager starts ApplicationMaster.

3. The ApplicationMaster then communicates with the ResourceManager to request resource allocation.

4. After a container is allocated to it, the ApplicationMaster communicates with the NodeManager to launch the tasks in the container.



Simple cluster with Hadoop Daemons