Hadoop Core components and Daemons

2 components of Hadoop

1.HDFS(storage)

2.YARN/MRv2(processing)

Daemons

Namenode
Datanode

SecondaryNameNode

ResourceManger
NodeManger
App Master

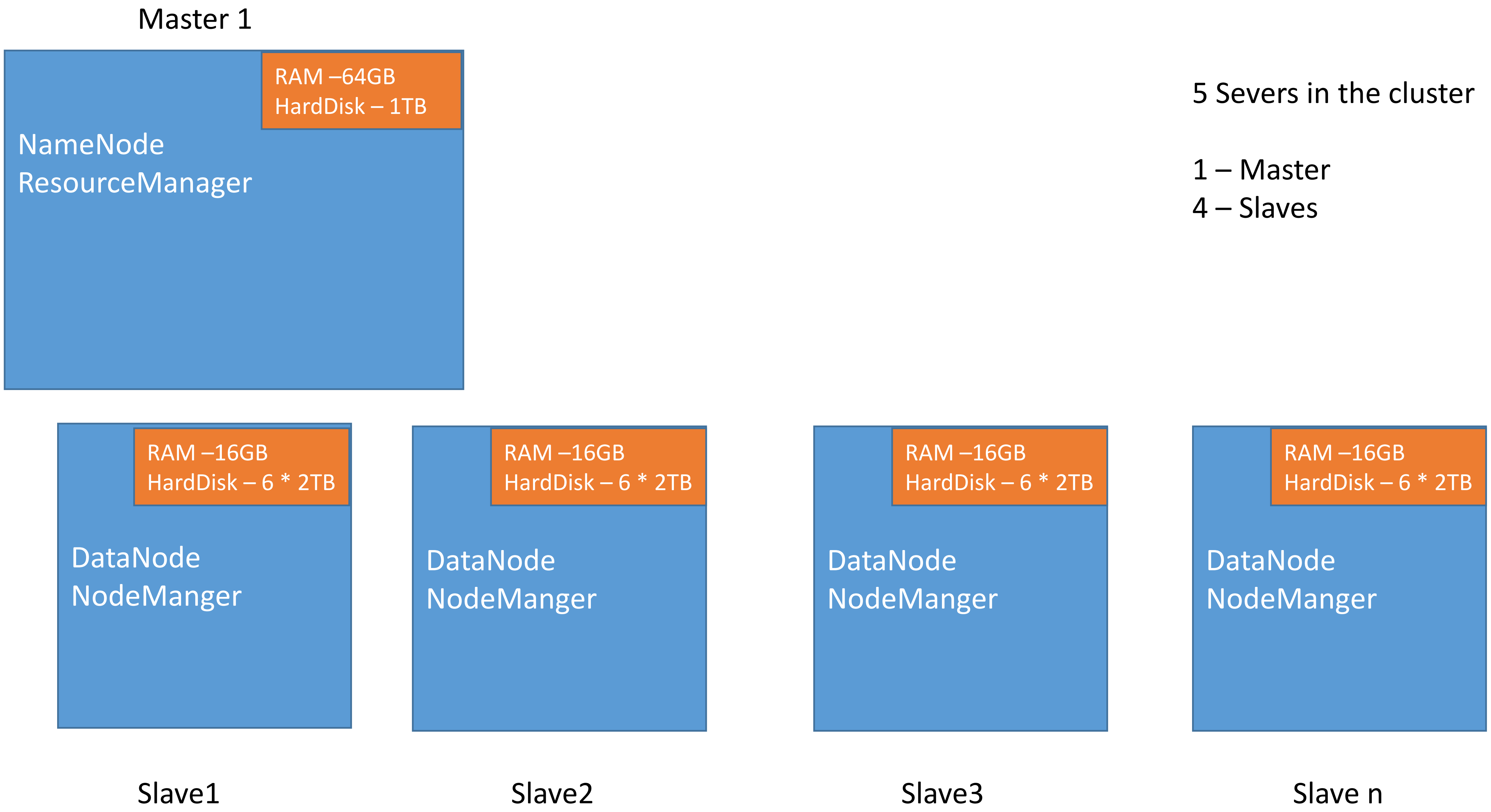
Hadoop Core components and Daemons

2 components of Hadoop
Master-Slave

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

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

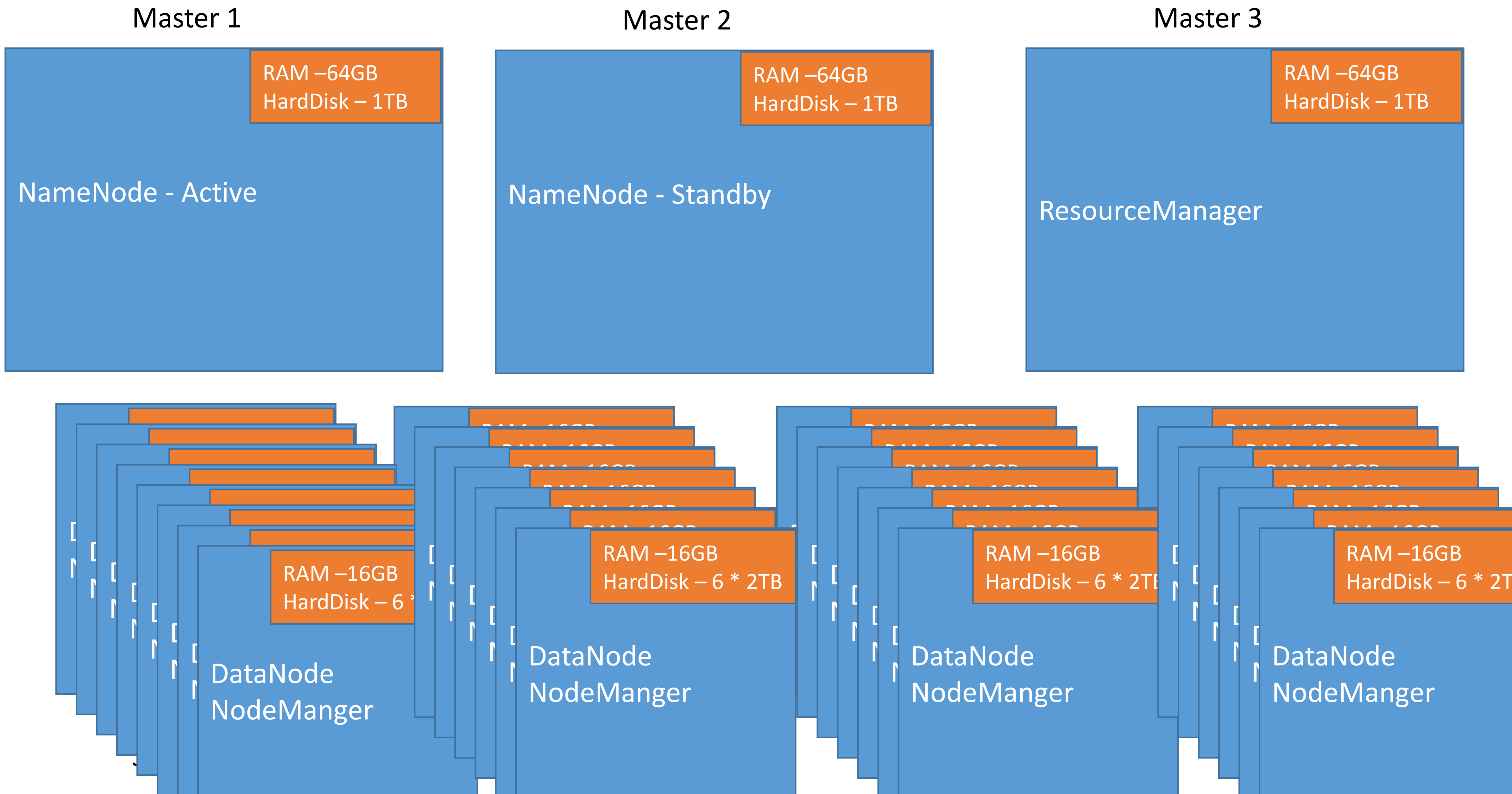
Simple cluster with Hadoop Daemons



5 Severs in the cluster

1 – Master
4 – Slaves

Simple cluster with Hadoop Daemons



File Blocks

By Default 64 MB split

100mb – abc.txt

64mb – Block 1
36mb -- Block 2

200mb – emp.dat

64mb – block1
64mb – block2
64mb – block3
8mb – block4

500mb – weblog.dat

64mb – block1
64mb – block2
64mb – block3
64mb – block4
64mb – block5
64mb – block6
64mb – block7
52mb – block8

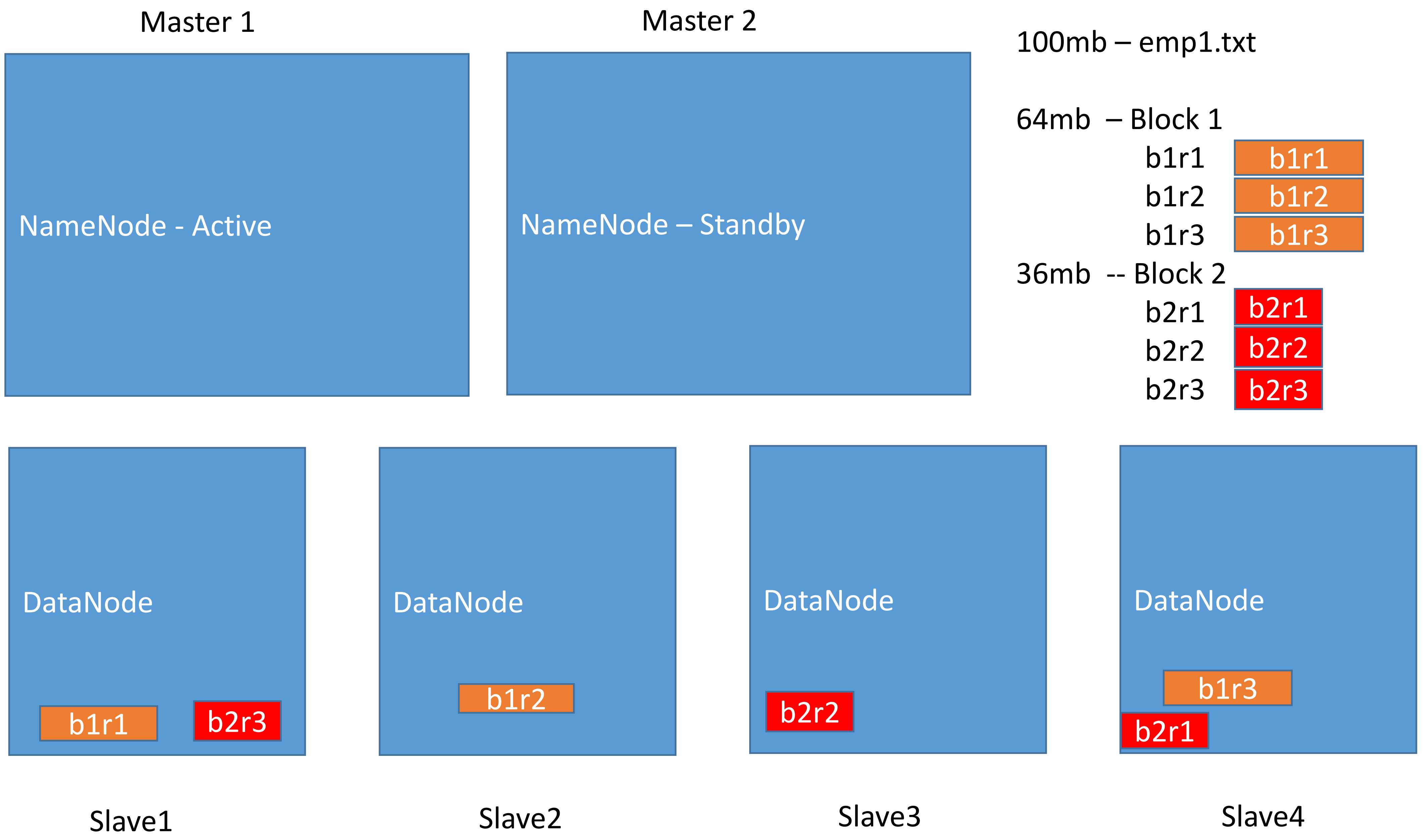
10 mb -- books.xml

Block1

100 kb – flower.jpg

Block1

Data Storage in Slaves with replication

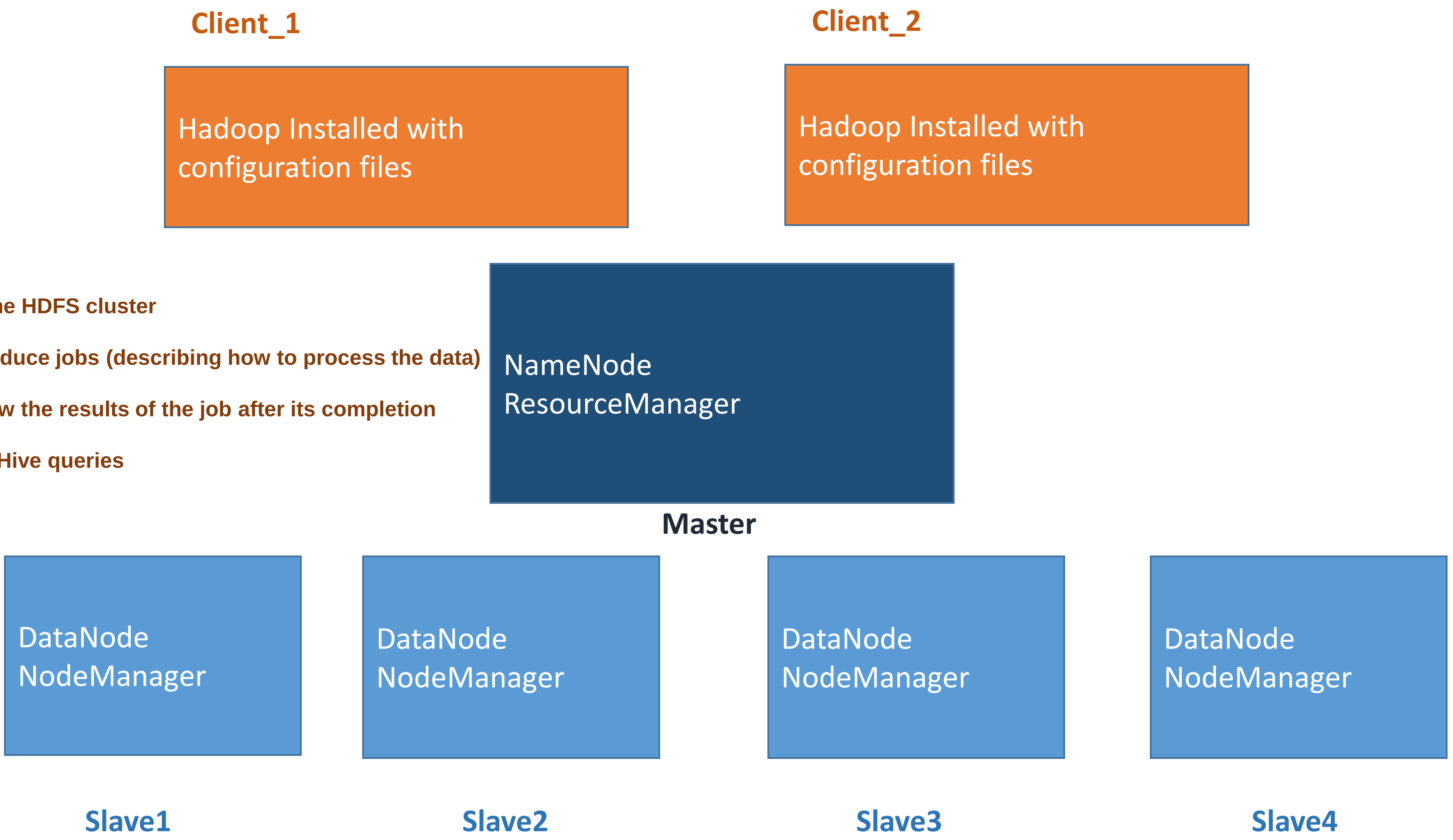


100mb – emp1.txt

64mb – Block 1
b1r1
b1r2
b1r3

36mb -- Block 2
b2r1
b2r2
b2r3

Namespace
Abc.txt
b1
r1
sl
b1
r2
s2
b1
r3
s4



Load data in the HDFS cluster

Submit MapReduce jobs (describing how to process the data)

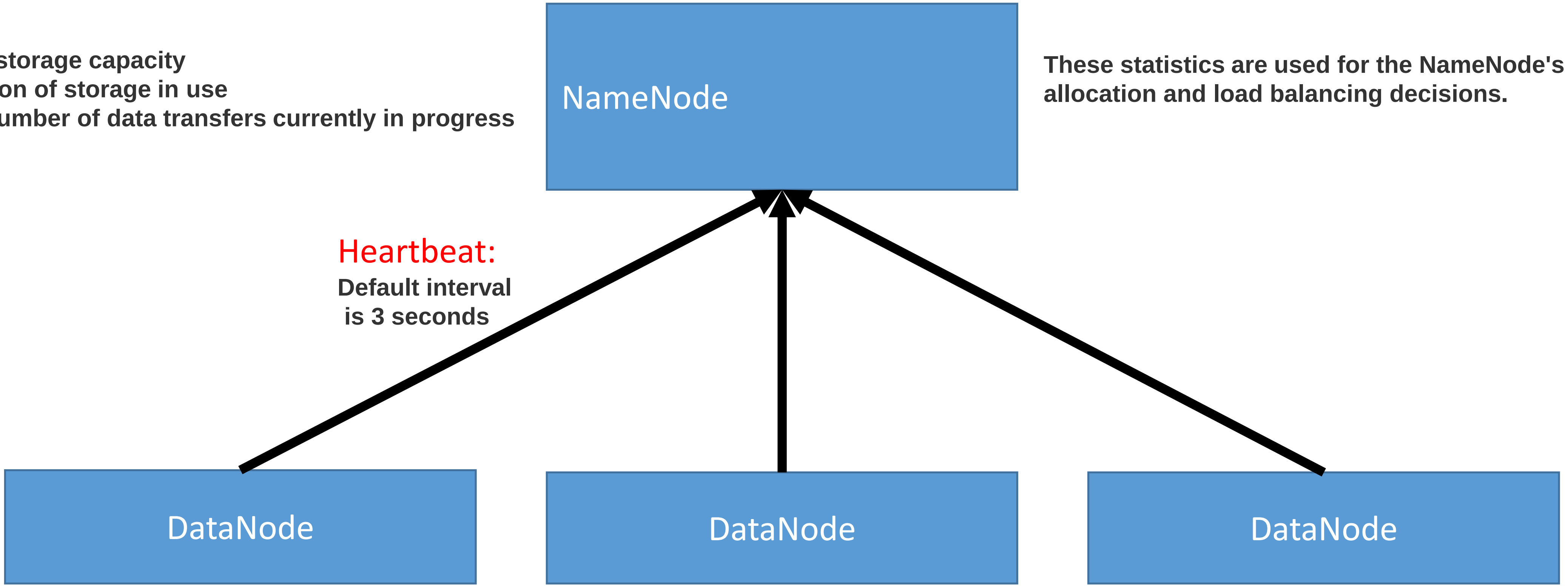
Retrieve or view the results of the job after its completion

Submit Pig or Hive queries

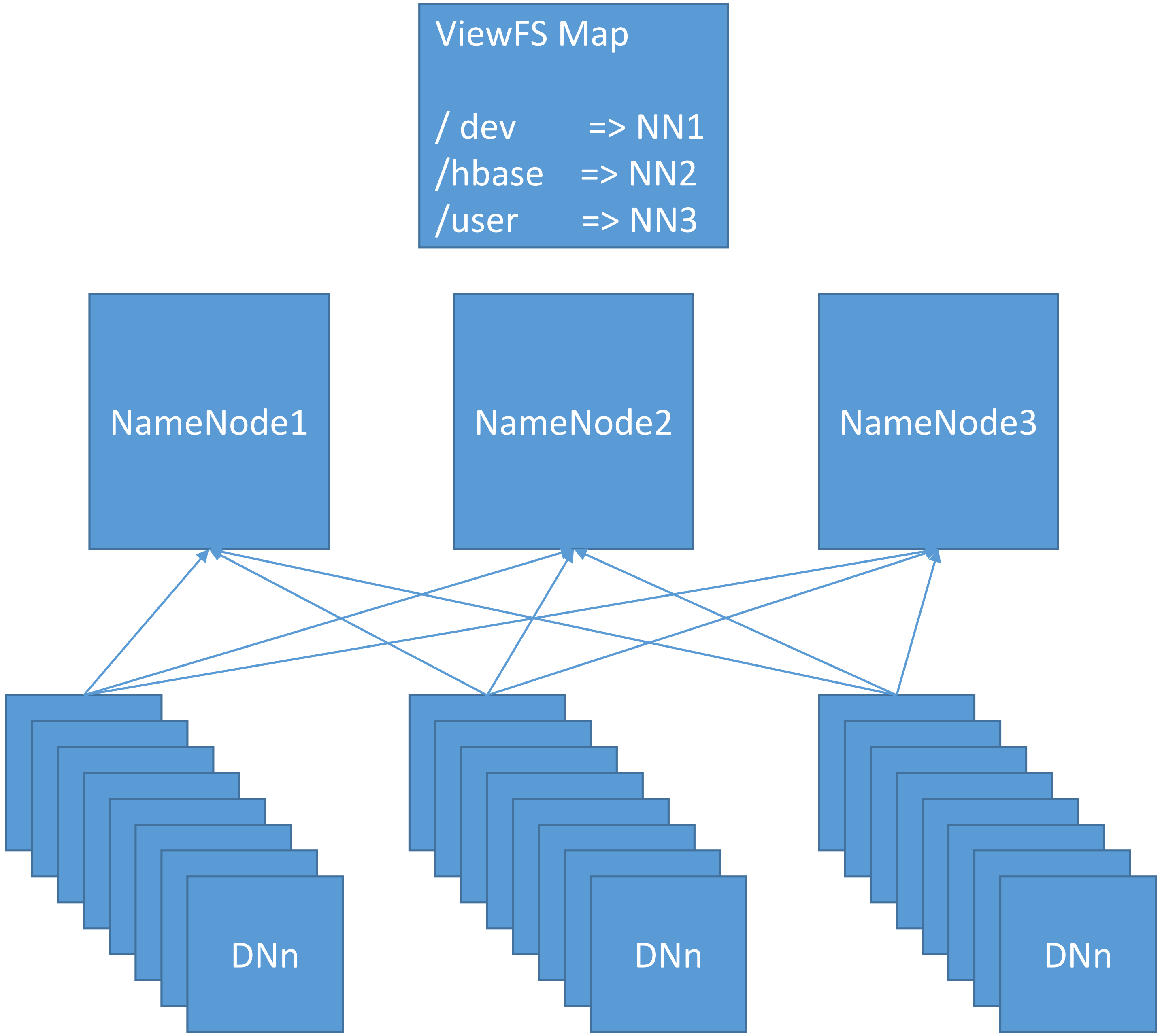
Heartbeat: Confirm that the DataNode is operating and the block replicas it hosts are available

- 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.



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



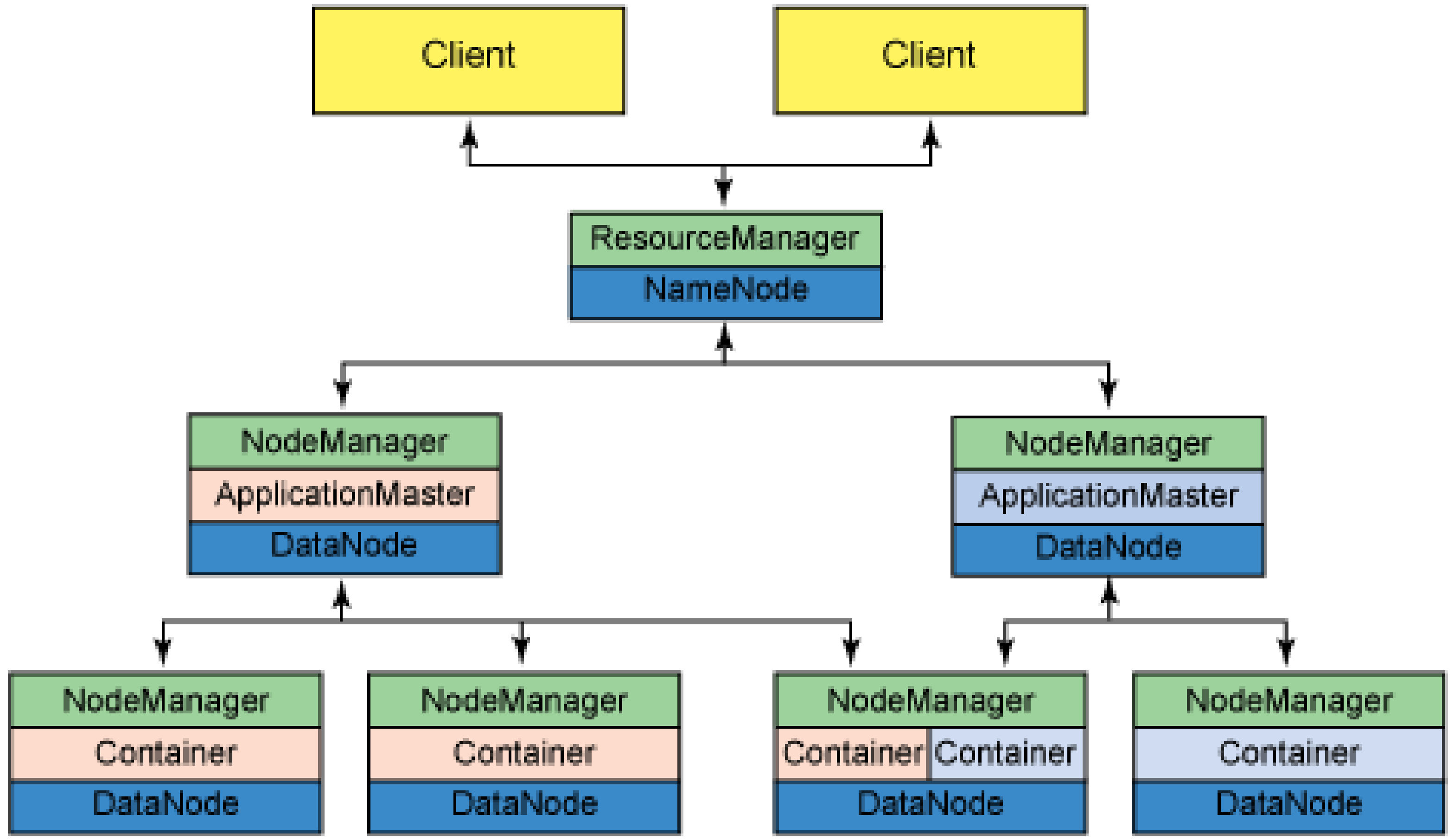
```
<configuration>
<property>
<name>dfs.nameservices</name>
<value>dev,hbase,user</value>
</property>

<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.

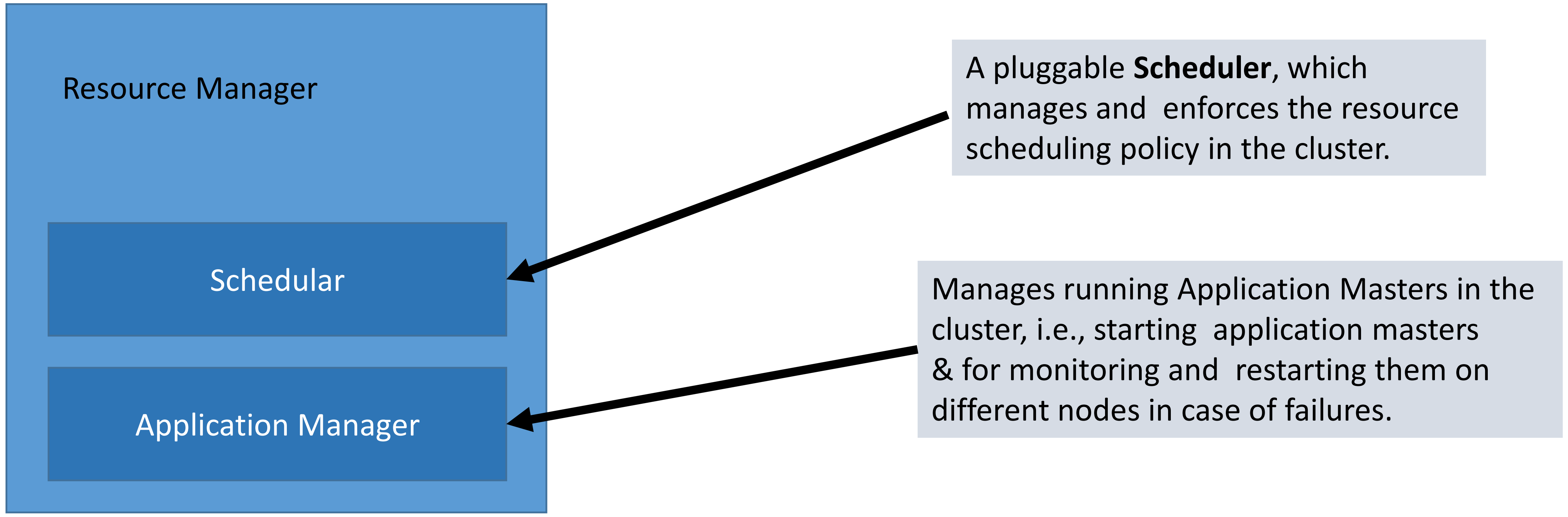
ApplicationMaster manages each instance of an application that runs within YARN.

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:



A pluggable **Scheduler**, which manages and enforces the resource scheduling policy in the cluster.

Manages running Application Masters in the cluster, i.e., starting application masters & for monitoring and restarting them on different nodes in case of failures.

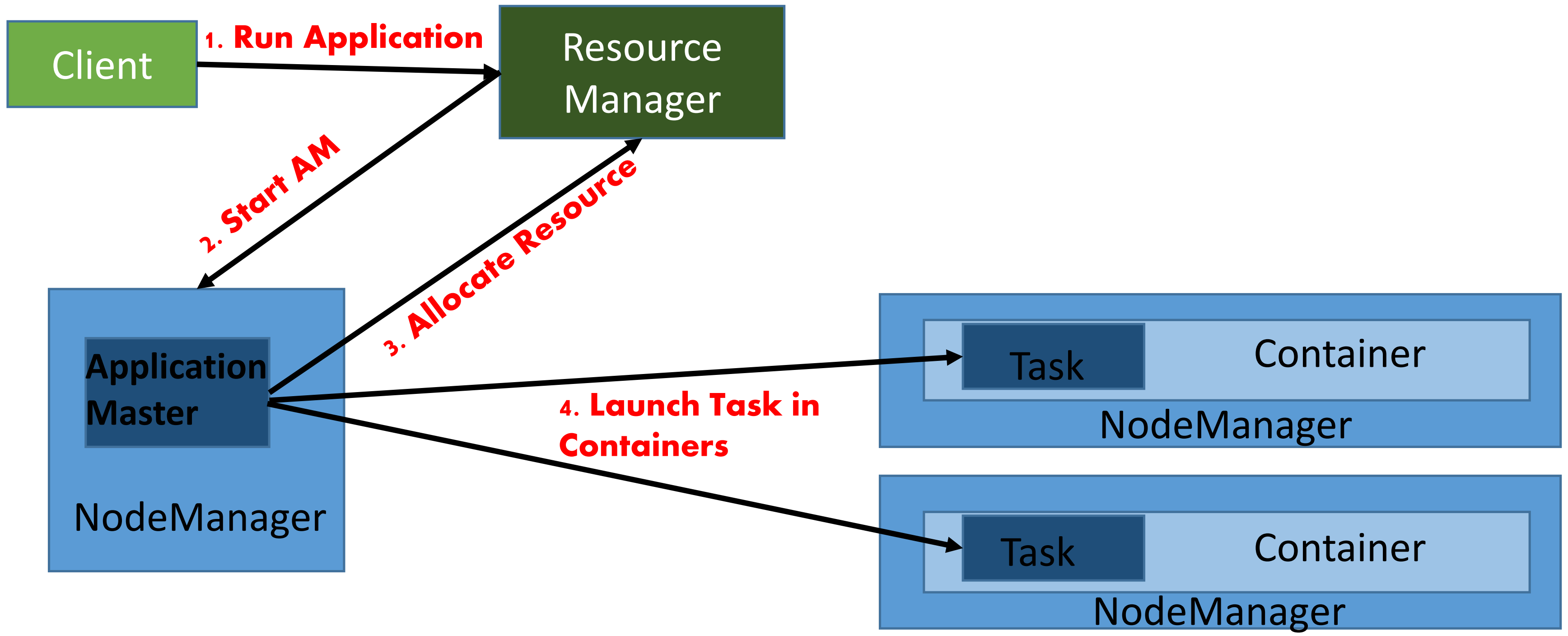
Application Submission in YARN

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.


Client
1. Run Application
Resource Manager
2. Start AM
3. Allocate Resource
Application Master
NodeManager
4. Launch Task in Containers
Task
Container
NodeManager
Task
Container
NodeManager