ACADGILD

Presents

Introduction to Big Data and Hadoop

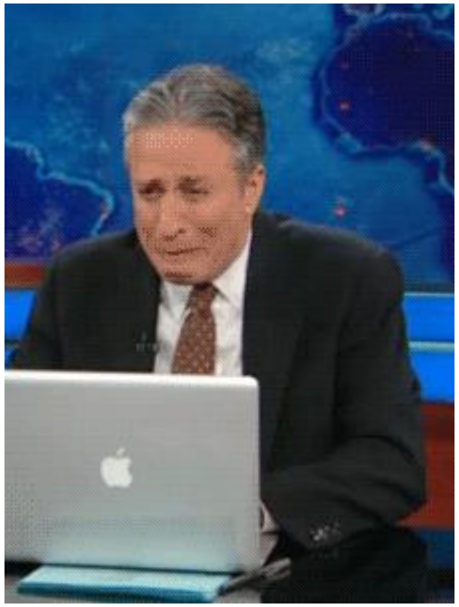
Session 2 – Hadoop Framework Description

Agenda – Hadoop

Breaking Big Data into Chunks …

Can you process a file of 1 TB size on your desktop or laptop ?

Noooooo !!



Hadoop Cluster Concepts

Hadoop cluster works as follows:

- Data is broken into chunks & distributed to different nodes.
- Each node(computer) processes data in it's storage to produce local results
- Results from each node are assembled to produce final results.

Why Hadoop Cluster?

- Data Storage has grown exponentially in recent past but data reading speed has not improved radically.

Data Read Speed Comparisons over time			
Year	**Data Size**	**Transfer Speed**	**Time Taken**
1990	1400 MB	4.5 MB/s	5 Minutes
2010	1 TB	100 MB/s	3 Hours
Hadoop Results			
2013	1TB	100 Drives	2 Minutes

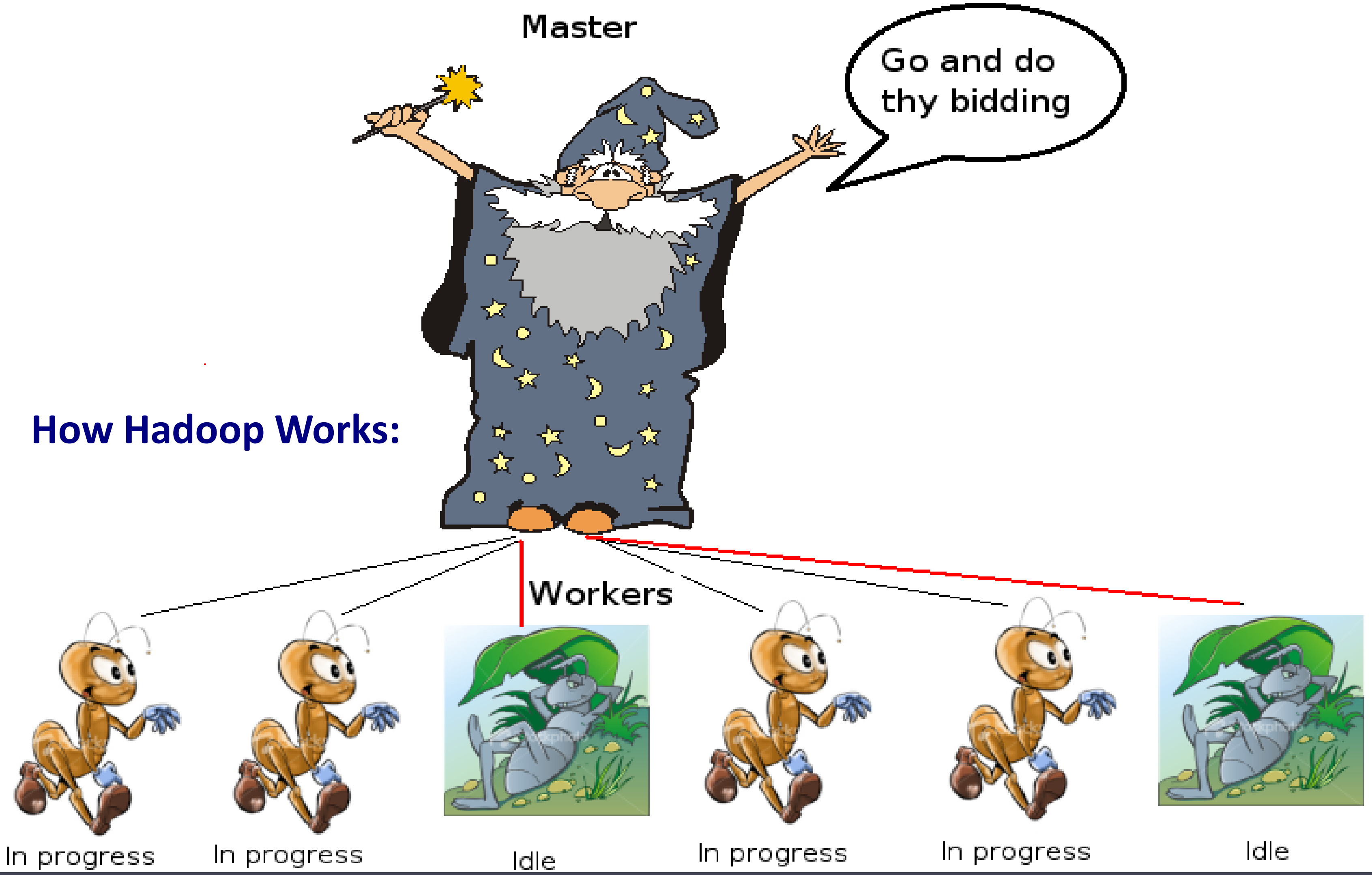
Why Hadoop 2 Came About?

MapReduce (MRv1) has following issues:

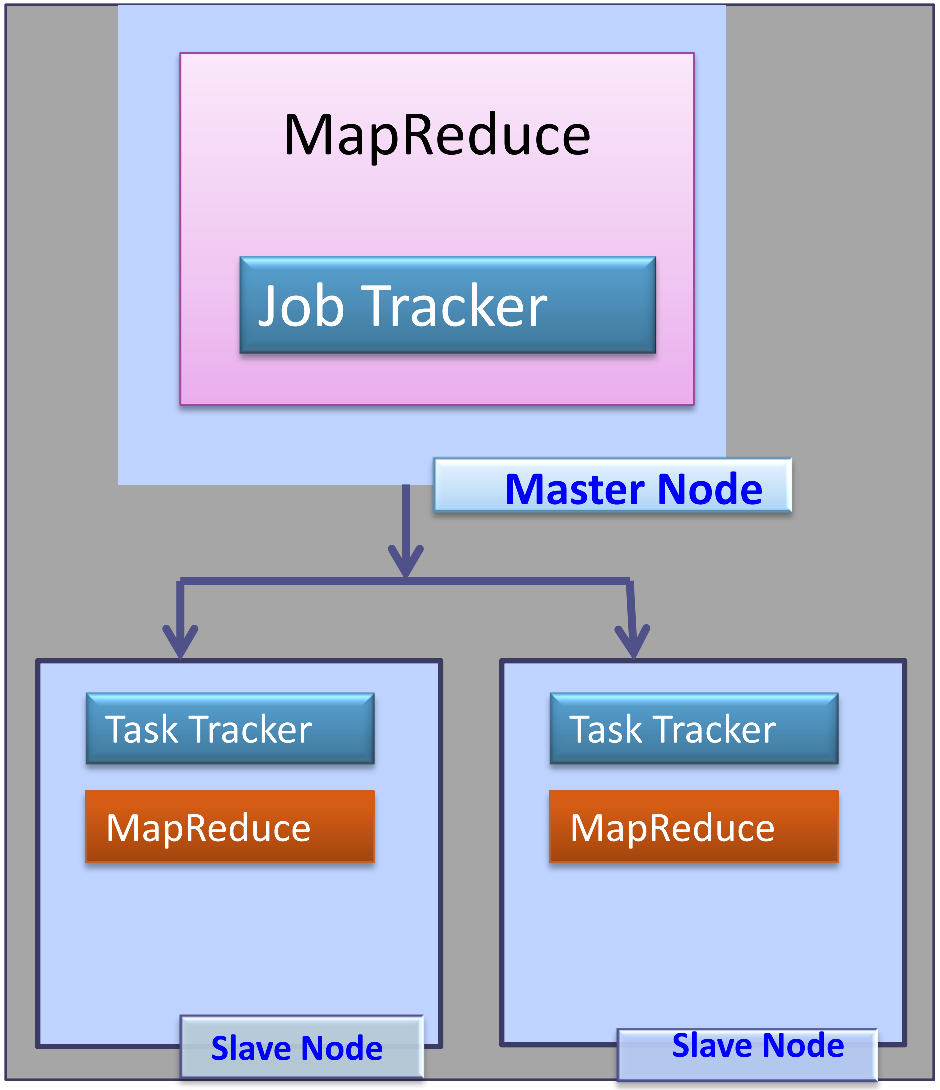
- Scalability Issues (max of 4000 nodes only) in MapReduce1
- Inflexible Resource Management Issue
  - MapReduce1 had slot based model.
  - Each TaskTracker is configured to have N slots at start-up.
  - A task is executed in a single slot.
  - Slots are configured on maximum memory at start-up (if more memory is freed up later on, it is unutilized).

How Hadoop Works

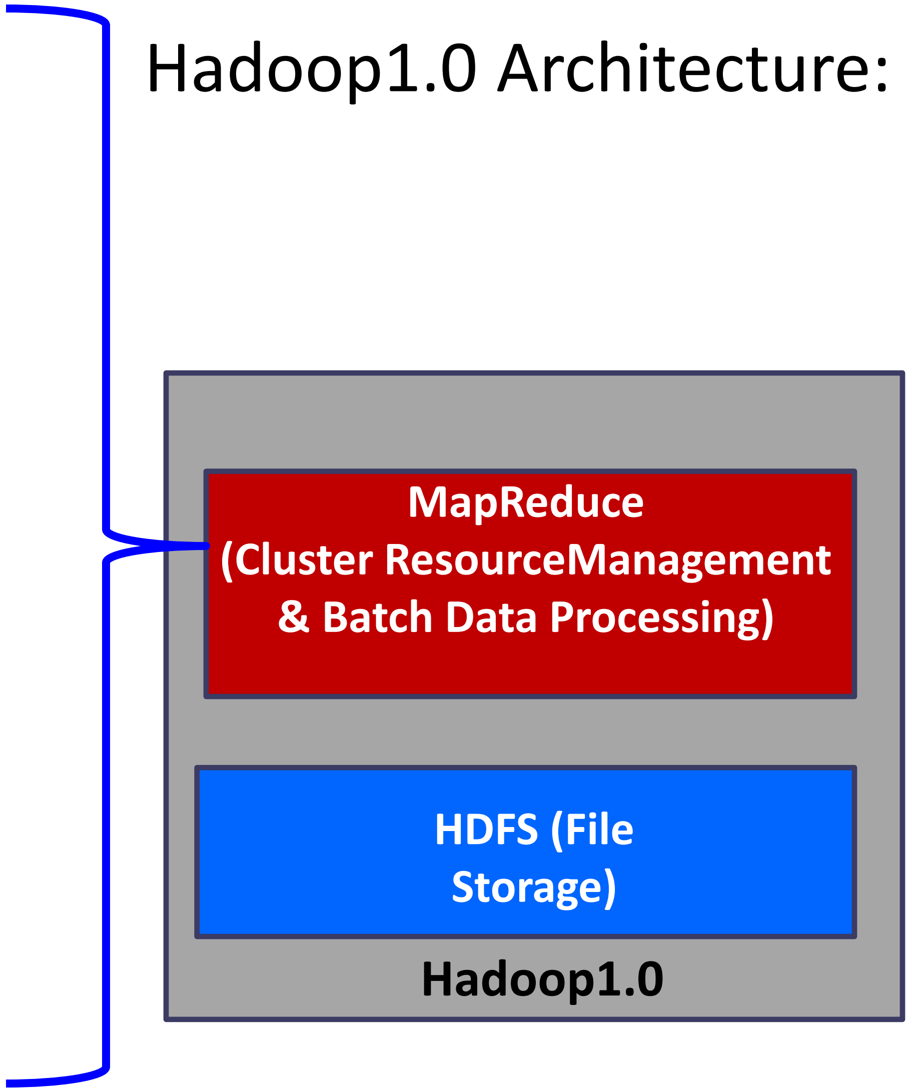
How Hadoop Works:



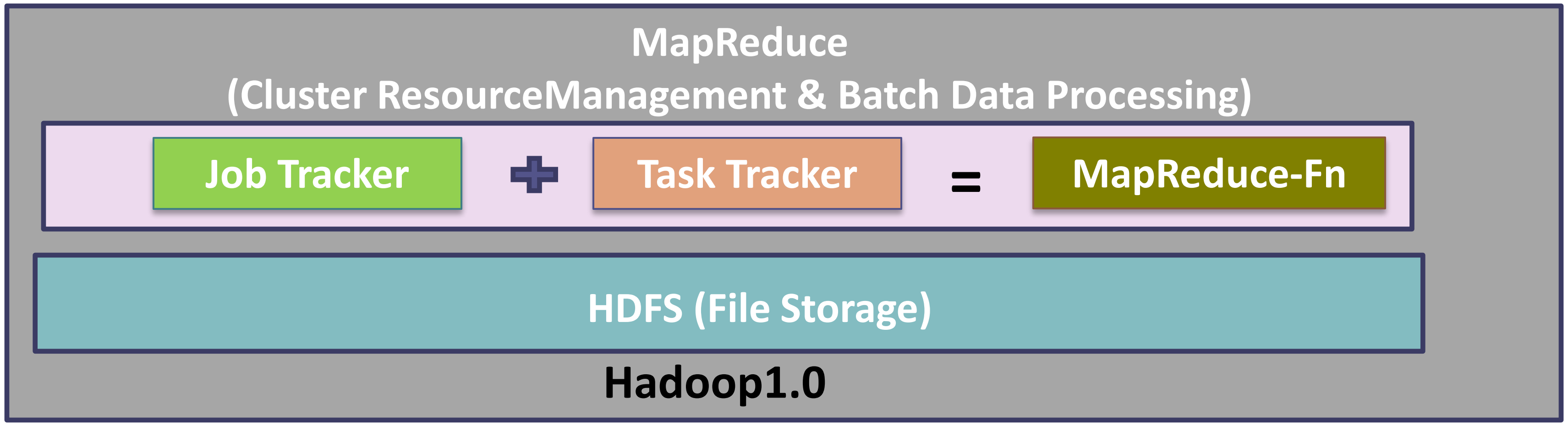
Hadoop1.x Architecture

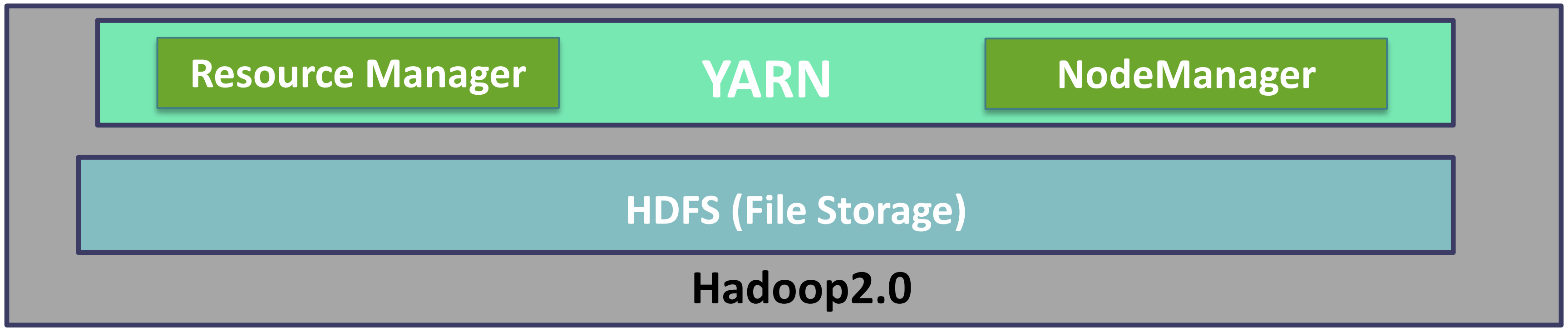


Hadoop1.0 Architecture:

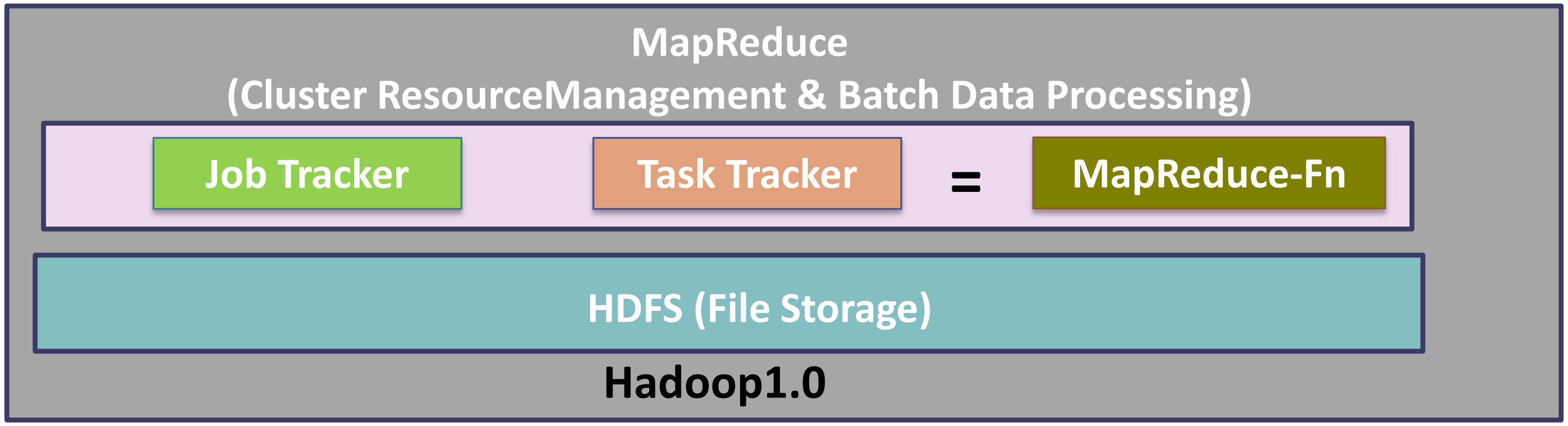

MapReduce (Cluster ResourceManagement & Batch Data Processing)
HDFS (File Storage)
Hadoop1.0

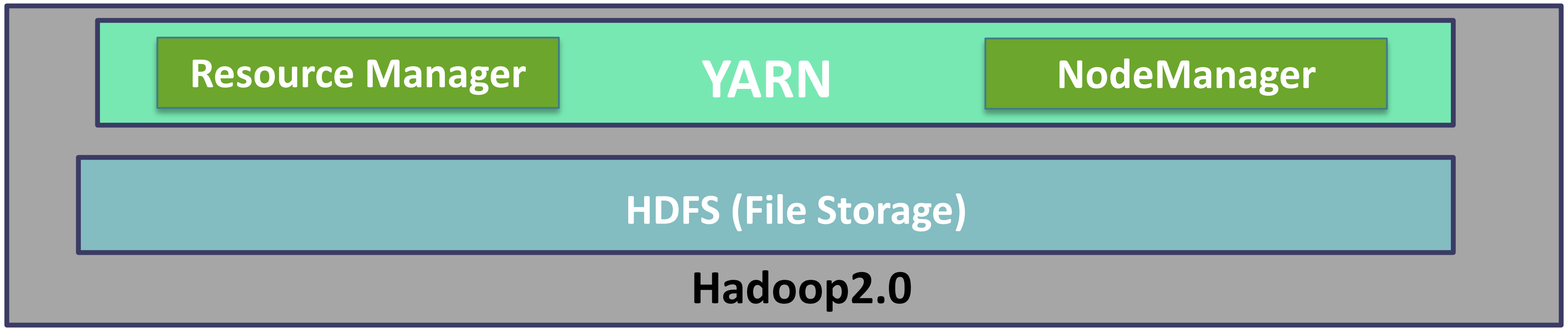
Progression from Hadoop1.x to Hadoop2



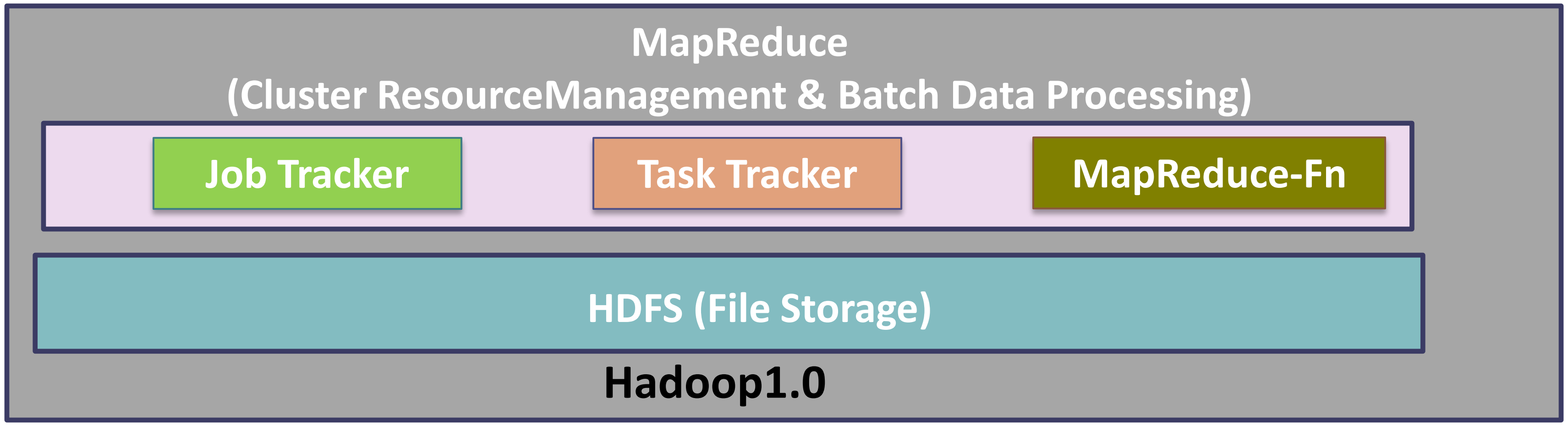


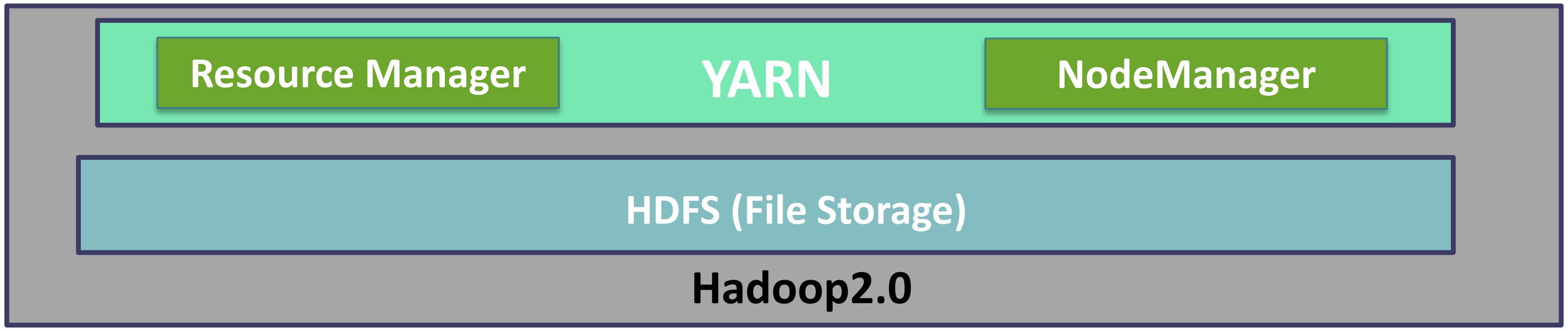
Progression from Hadoop1.x to Hadoop2



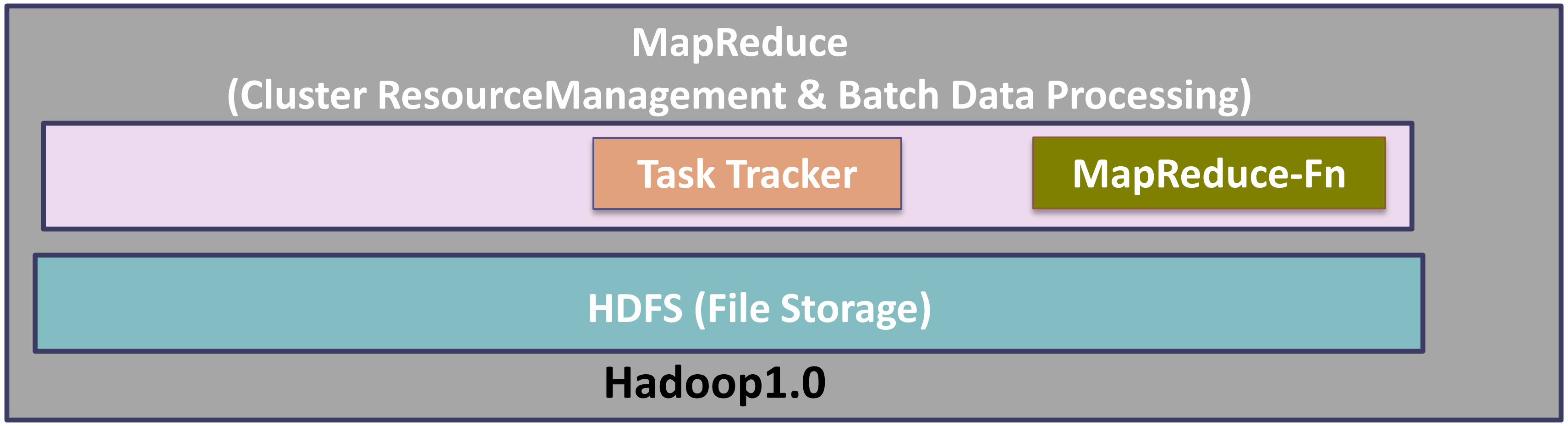


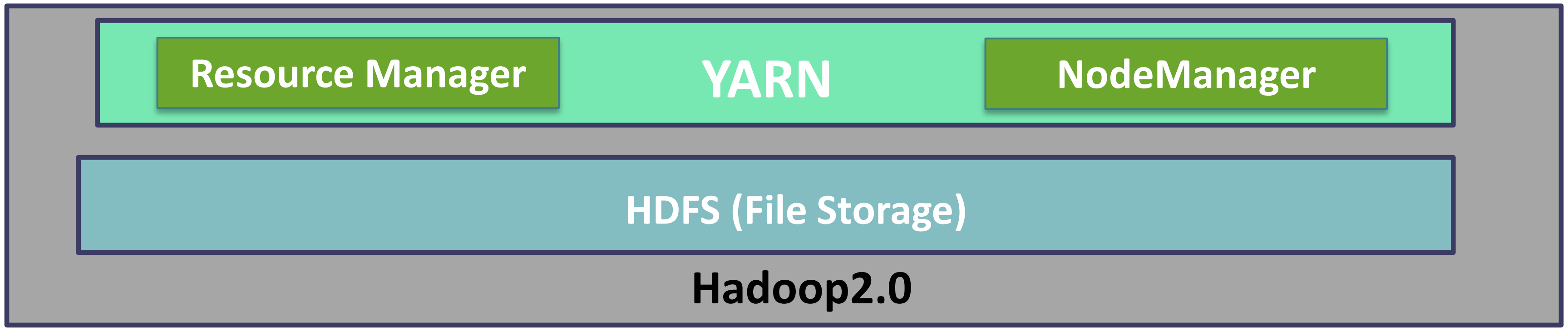
Progression from Hadoop1.x to Hadoop2



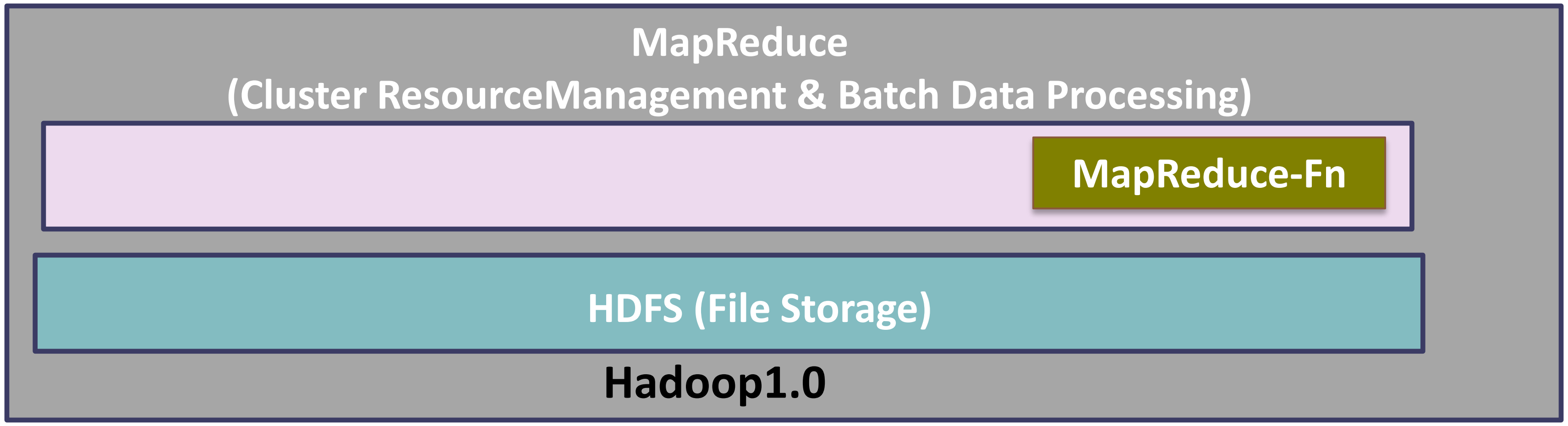


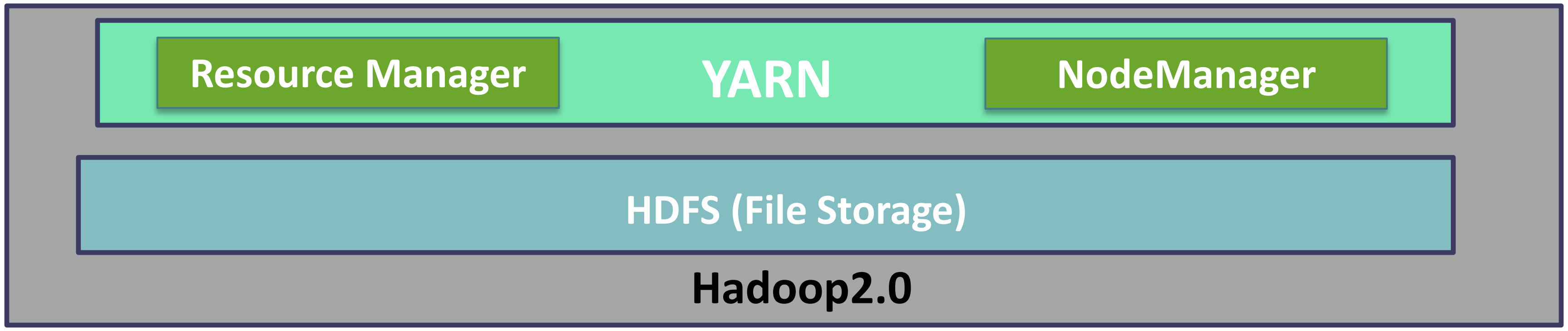
Progression from Hadoop1.x to Hadoop2



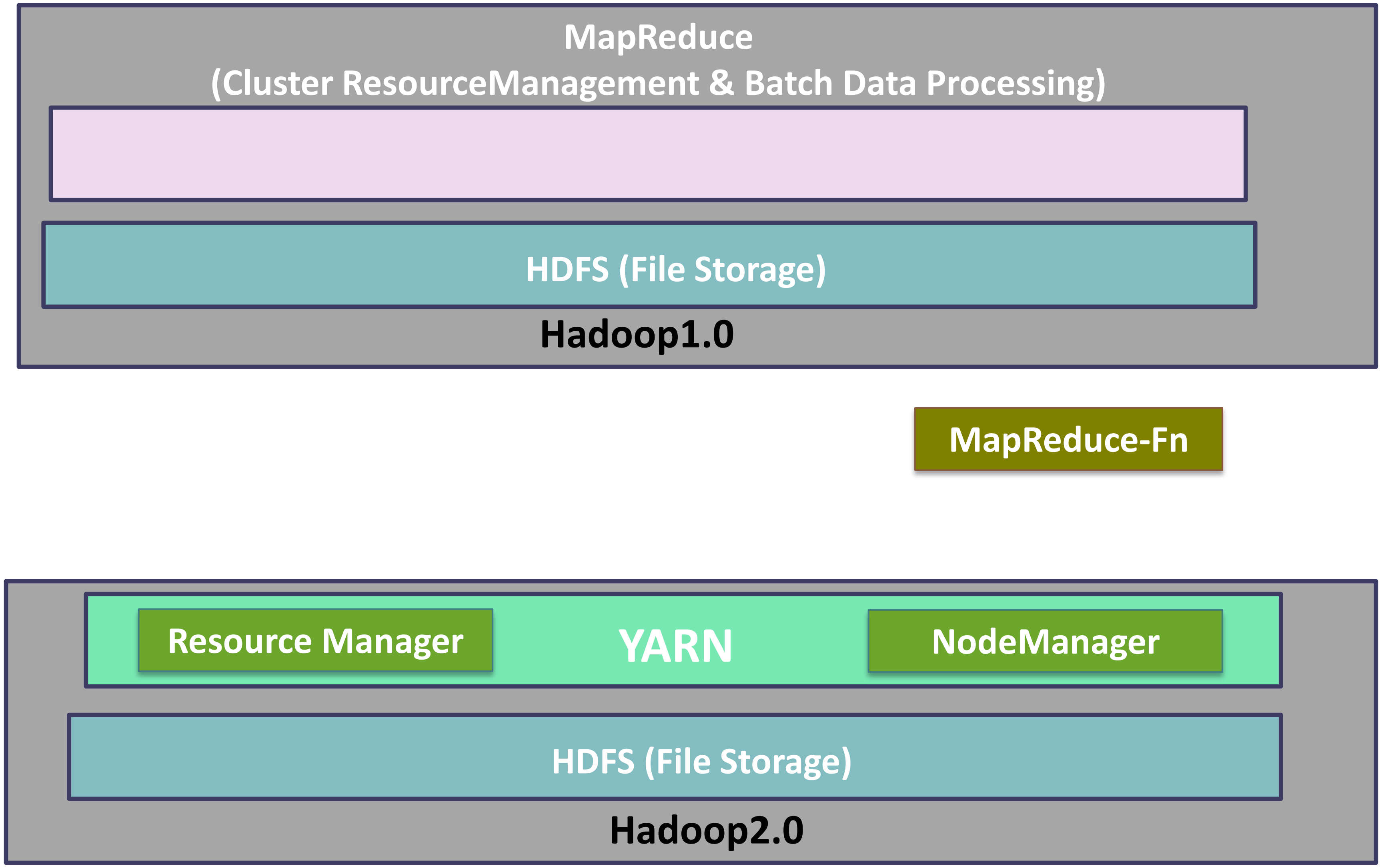


Progression from Hadoop1.x to Hadoop2

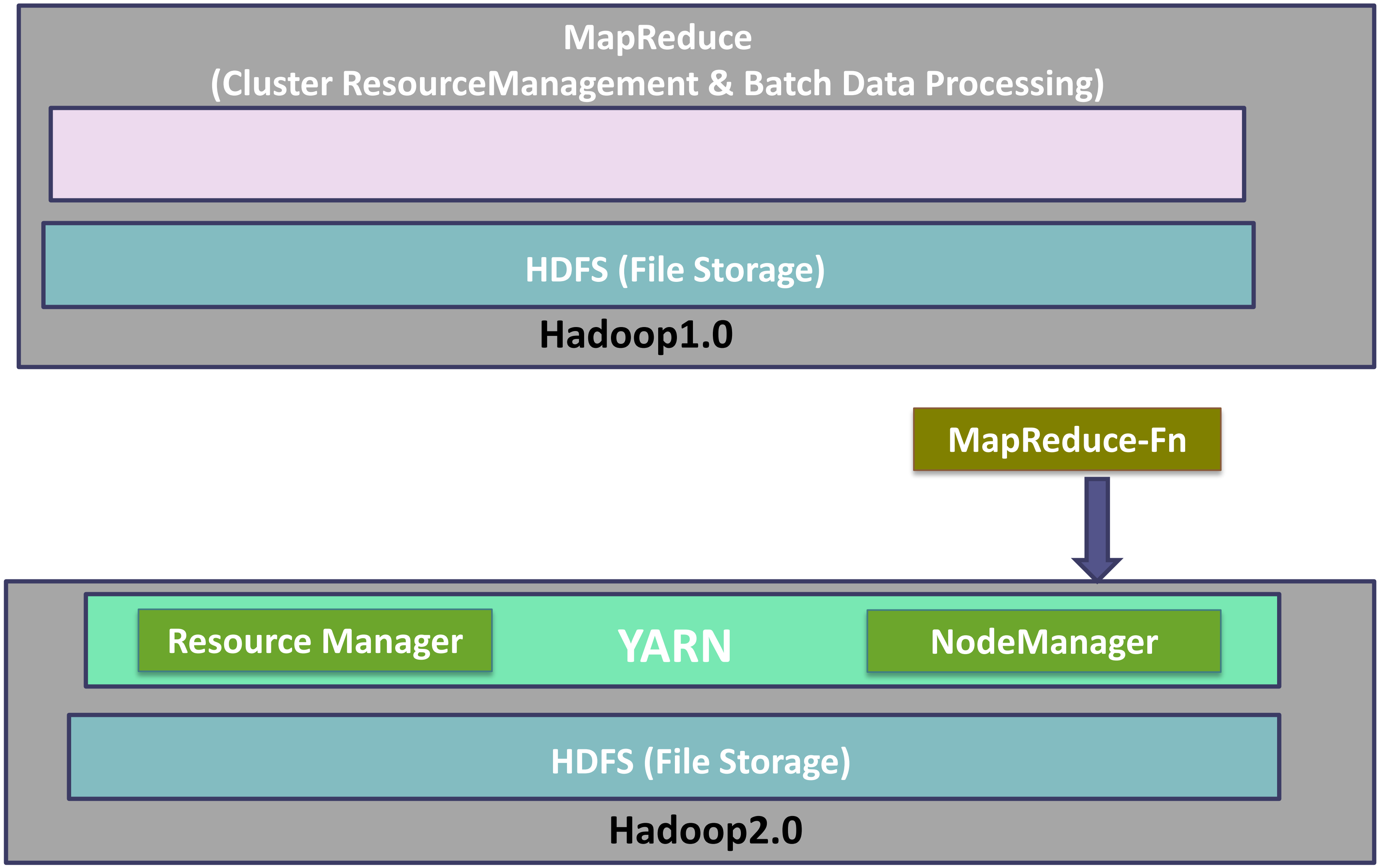




Progression from Hadoop1.x to Hadoop2



Progression from Hadoop1.x to Hadoop2

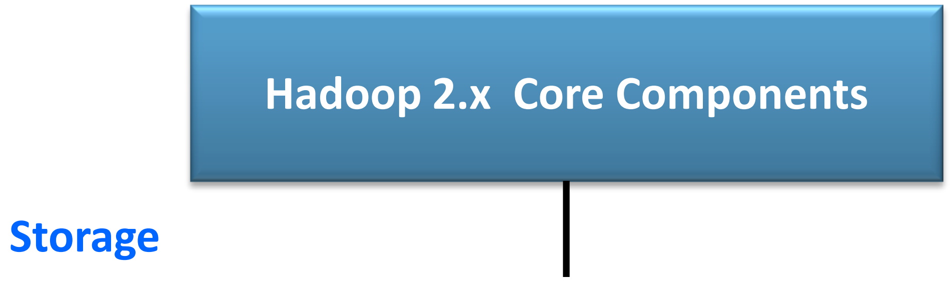


Core Components of Hadoop Cluster

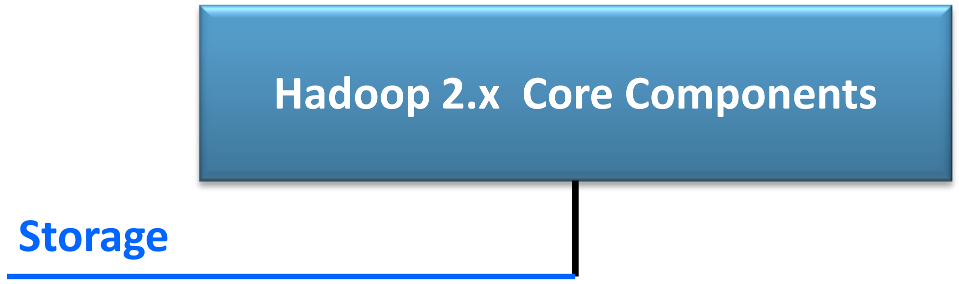
Core Components of Hadoop Cluster

Hadoop 2.x Core Components

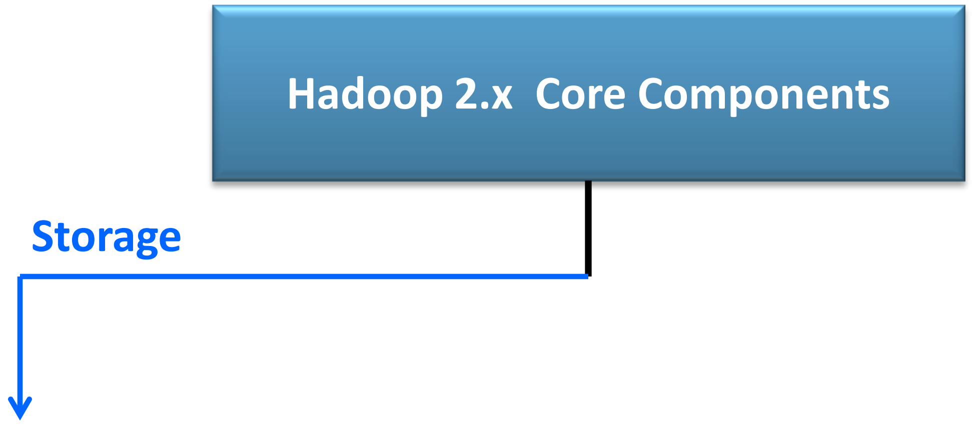
Core Components of Hadoop Cluster



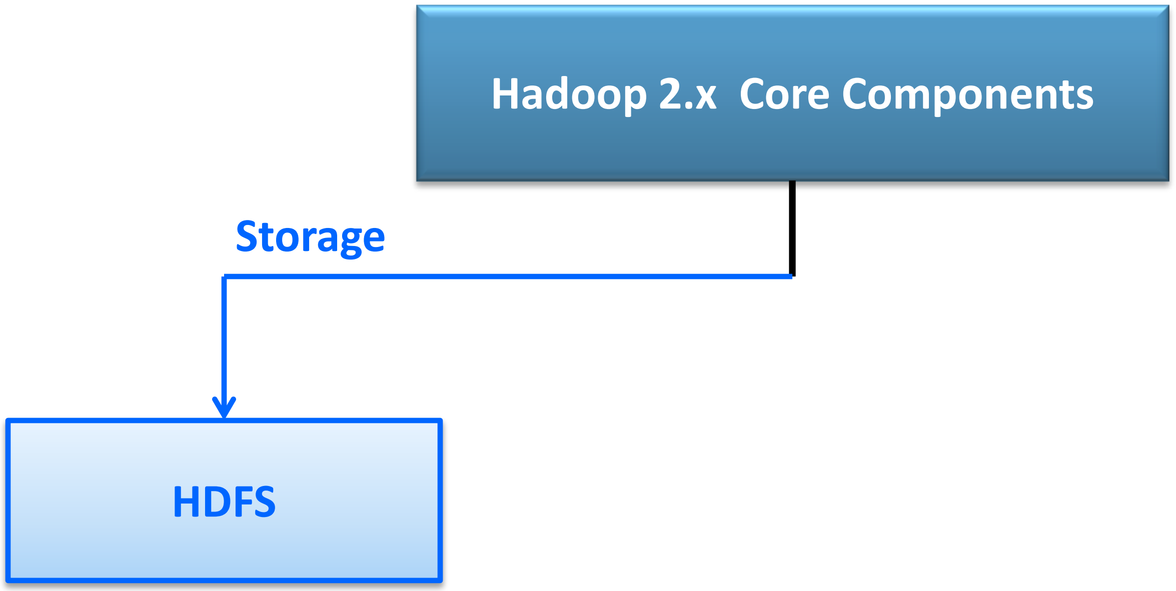
Core Components of Hadoop Cluster



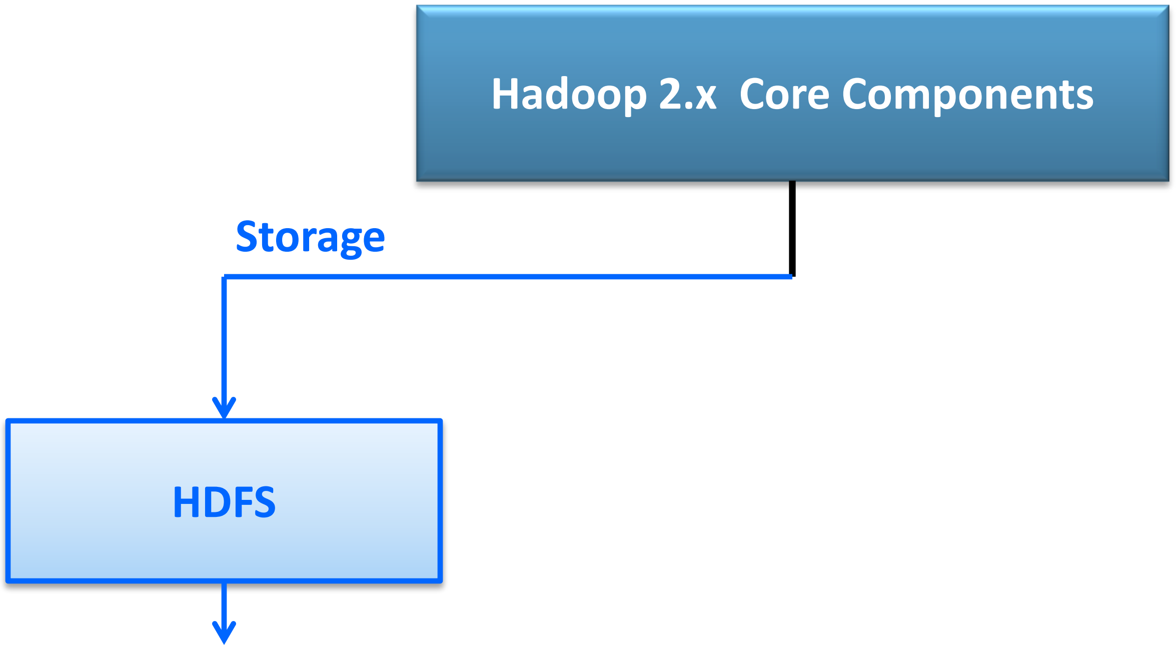
Core Components of Hadoop Cluster



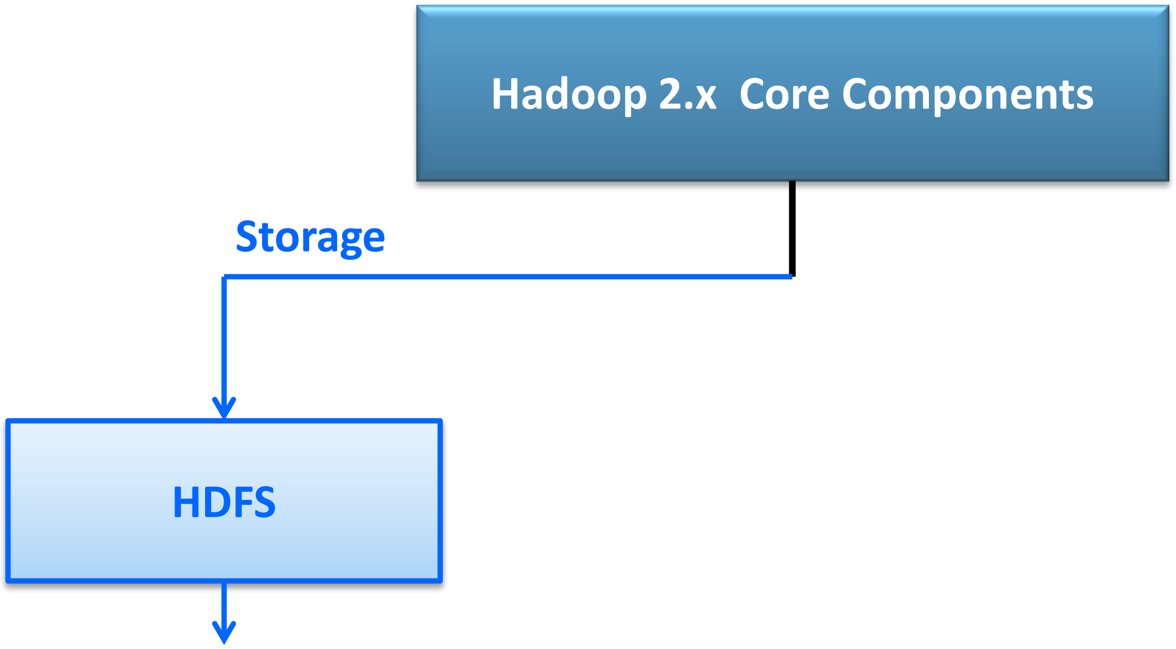
Core Components of Hadoop Cluster



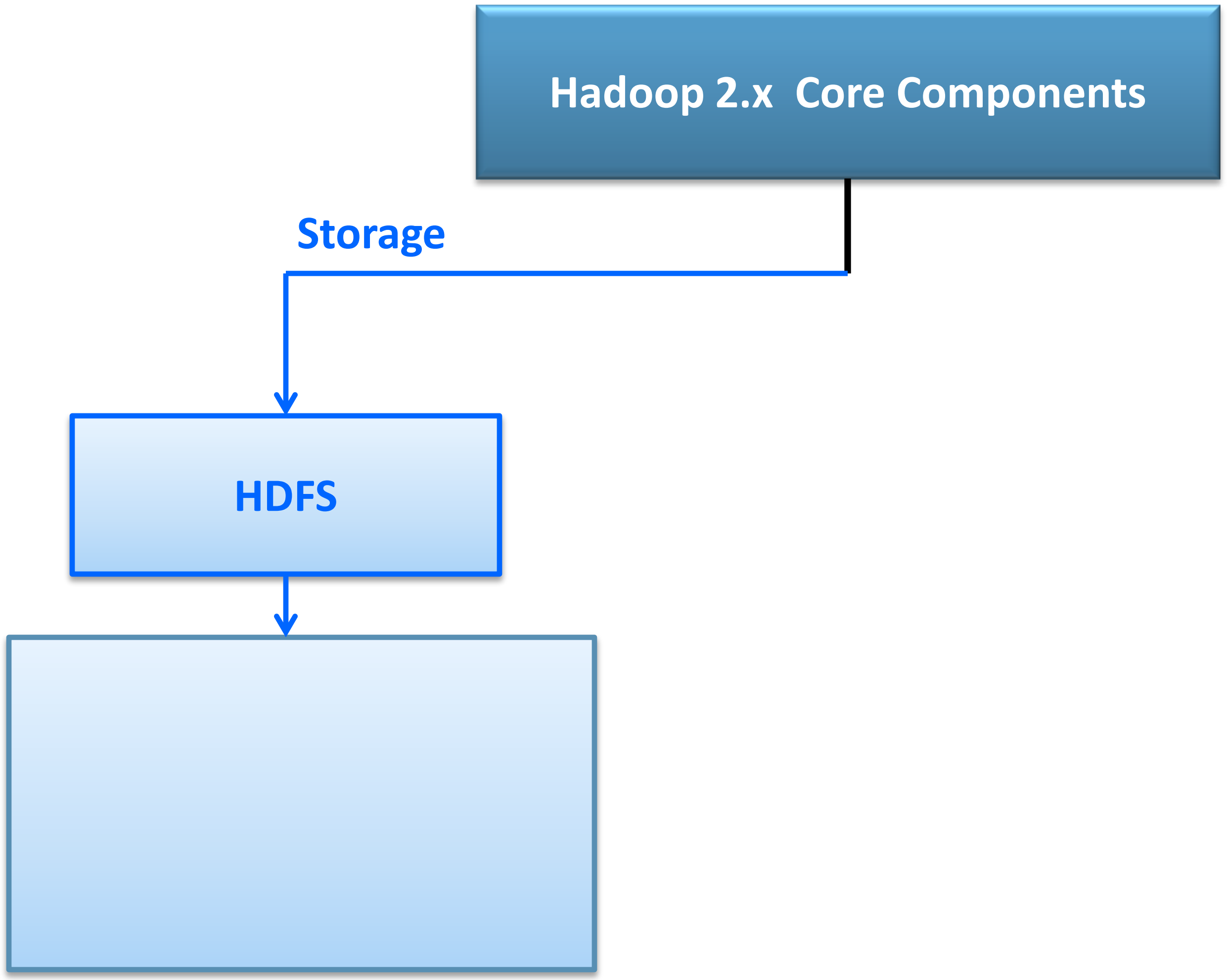
Core Components of Hadoop Cluster



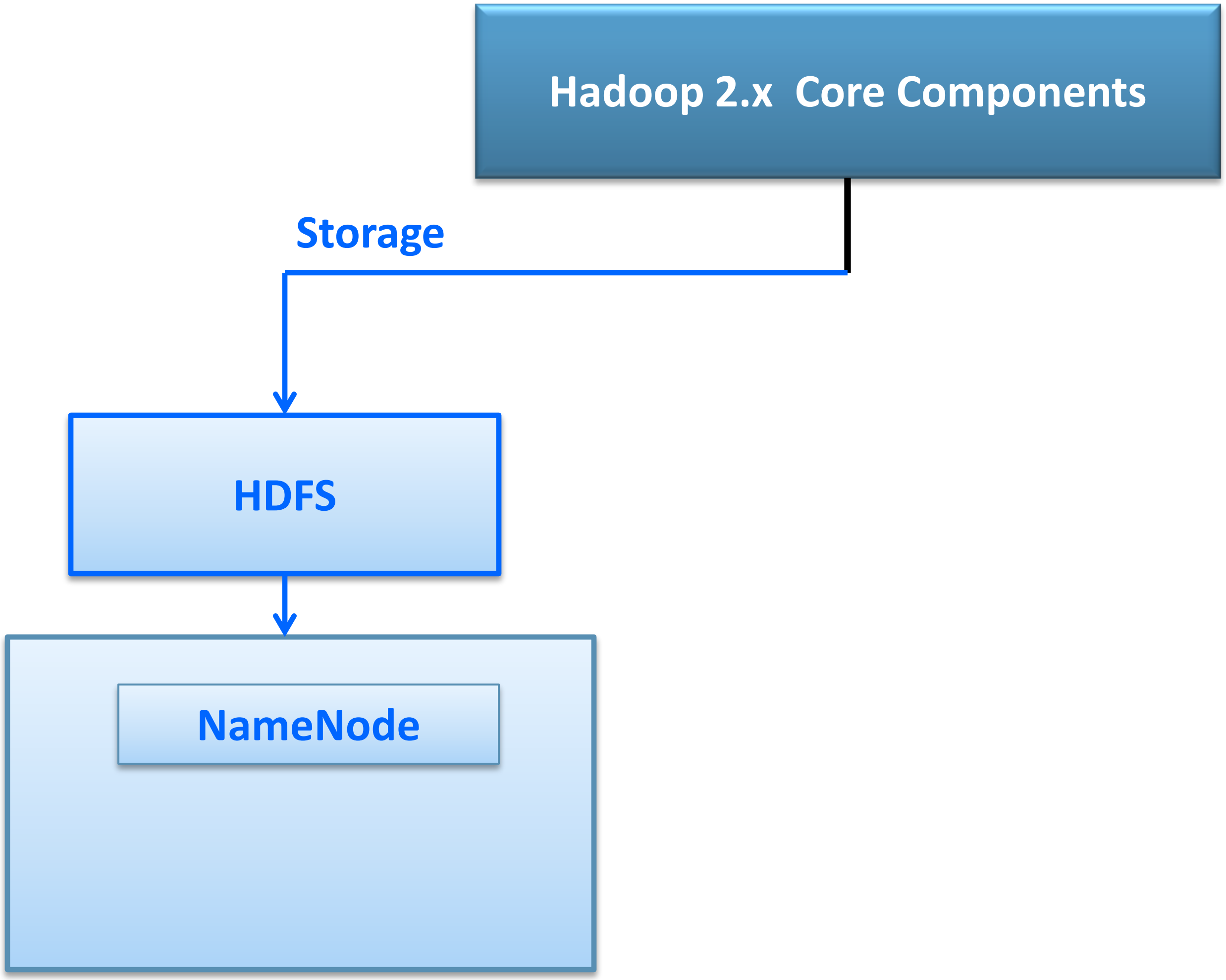
Core Components of Hadoop Cluster



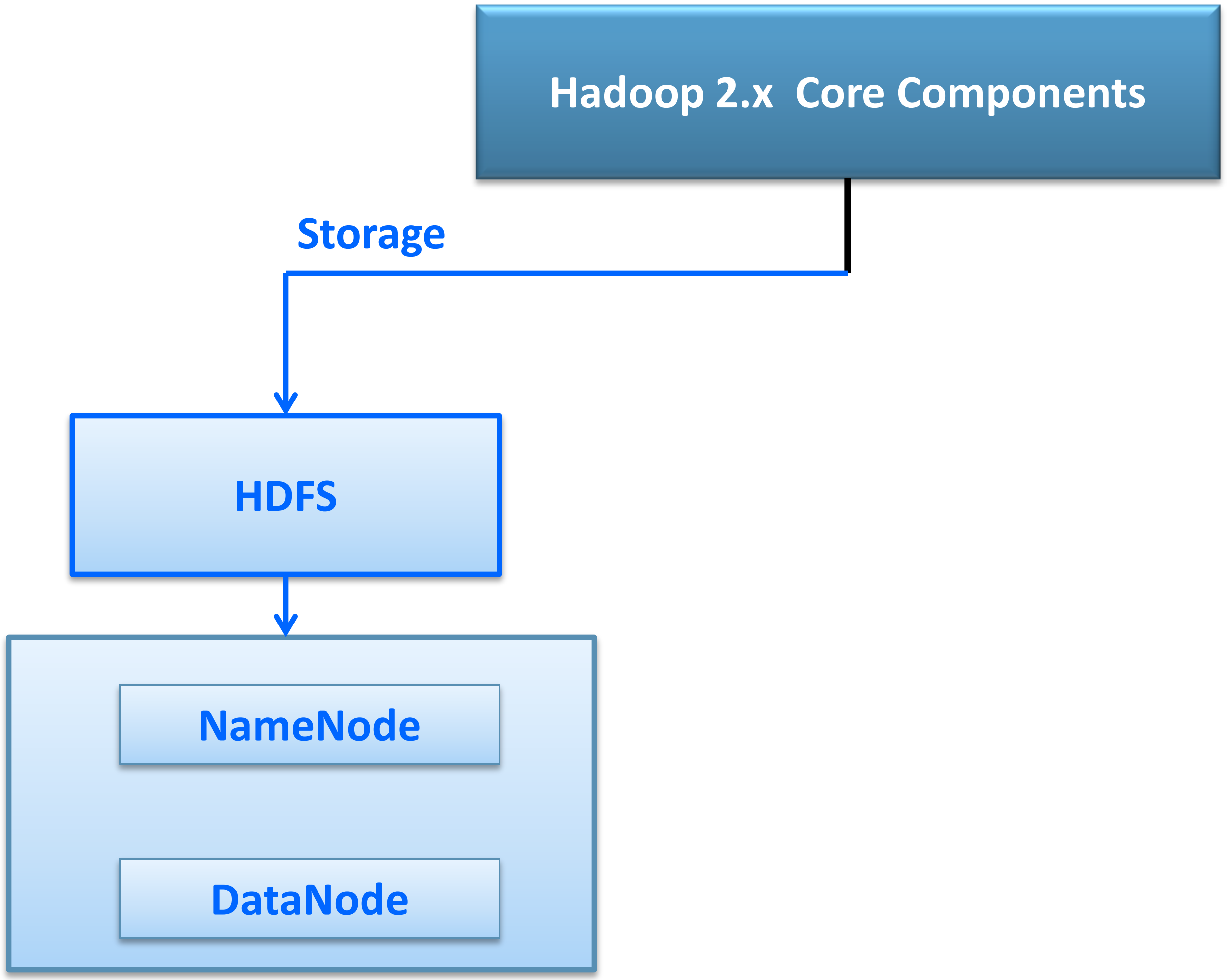
Core Components of Hadoop Cluster



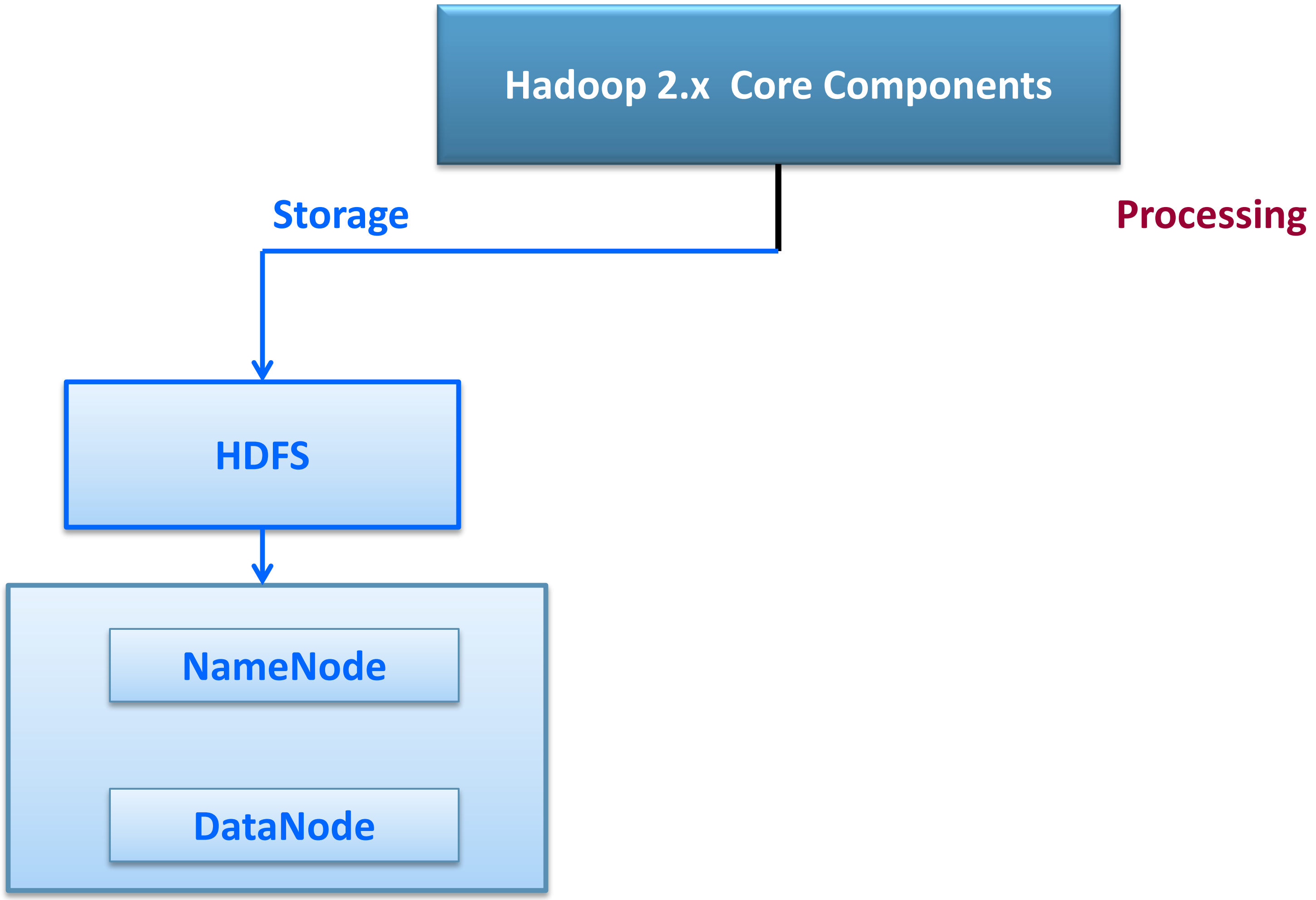
Core Components of Hadoop Cluster



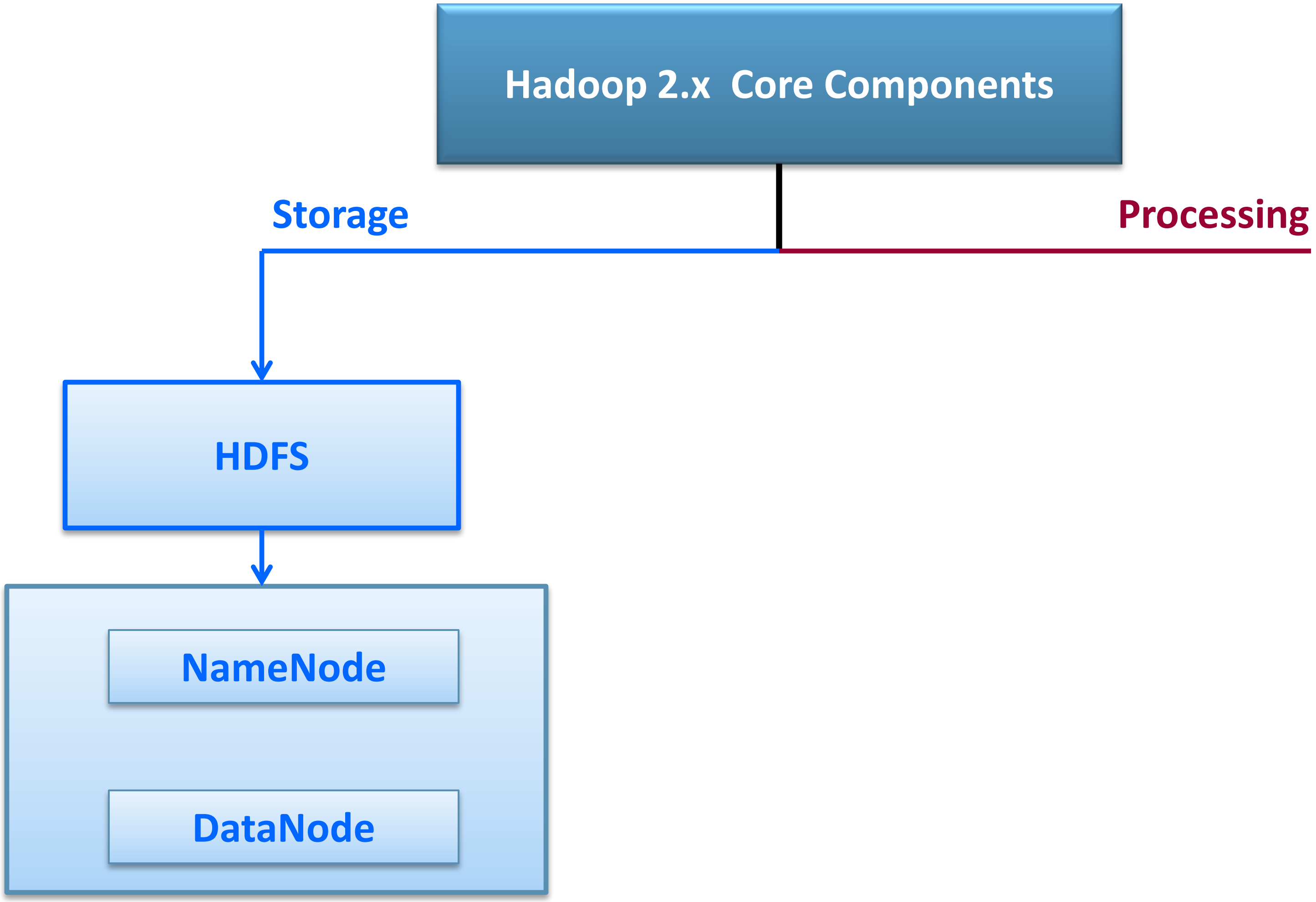
Core Components of Hadoop Cluster



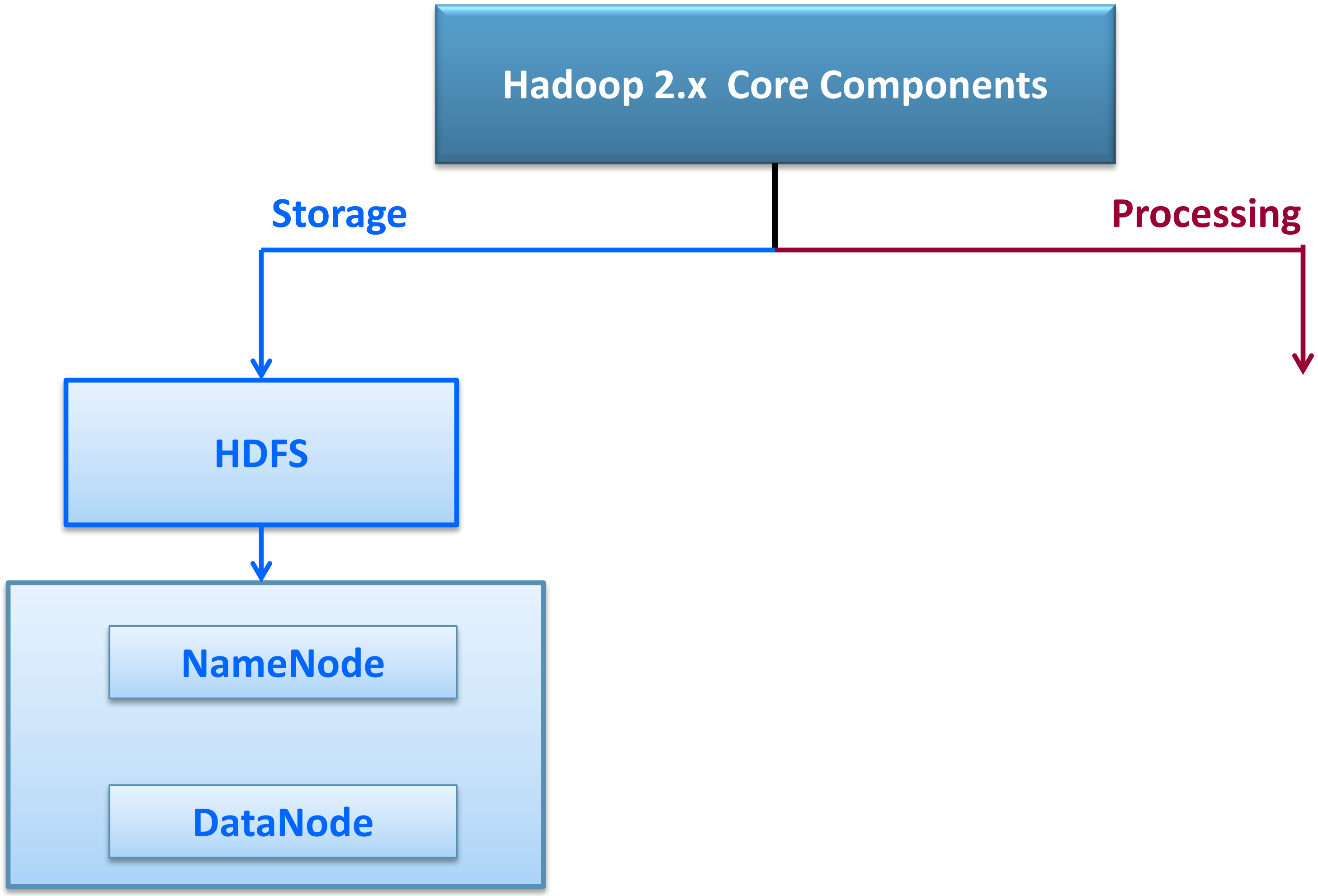
Core Components of Hadoop Cluster



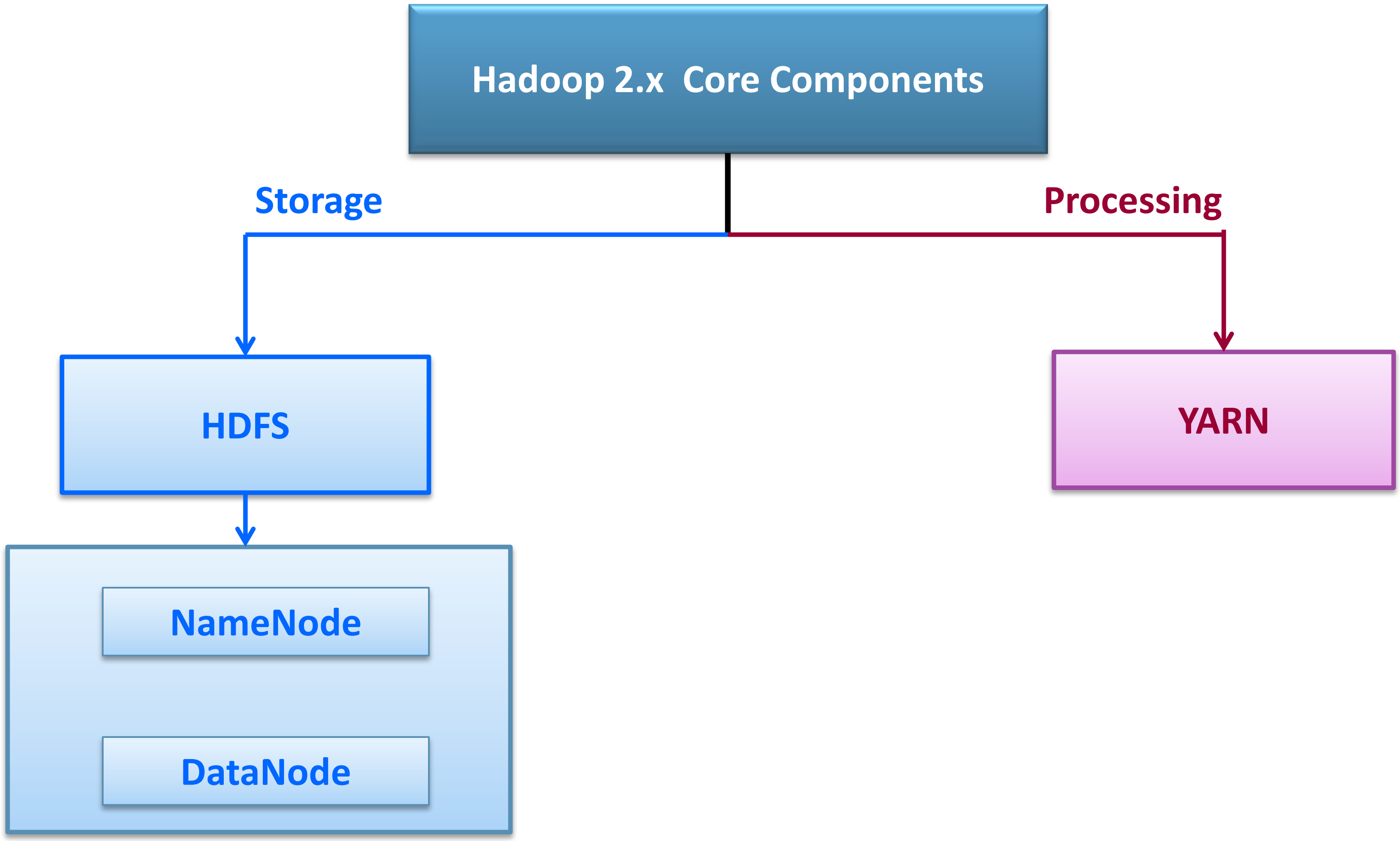
Core Components of Hadoop Cluster



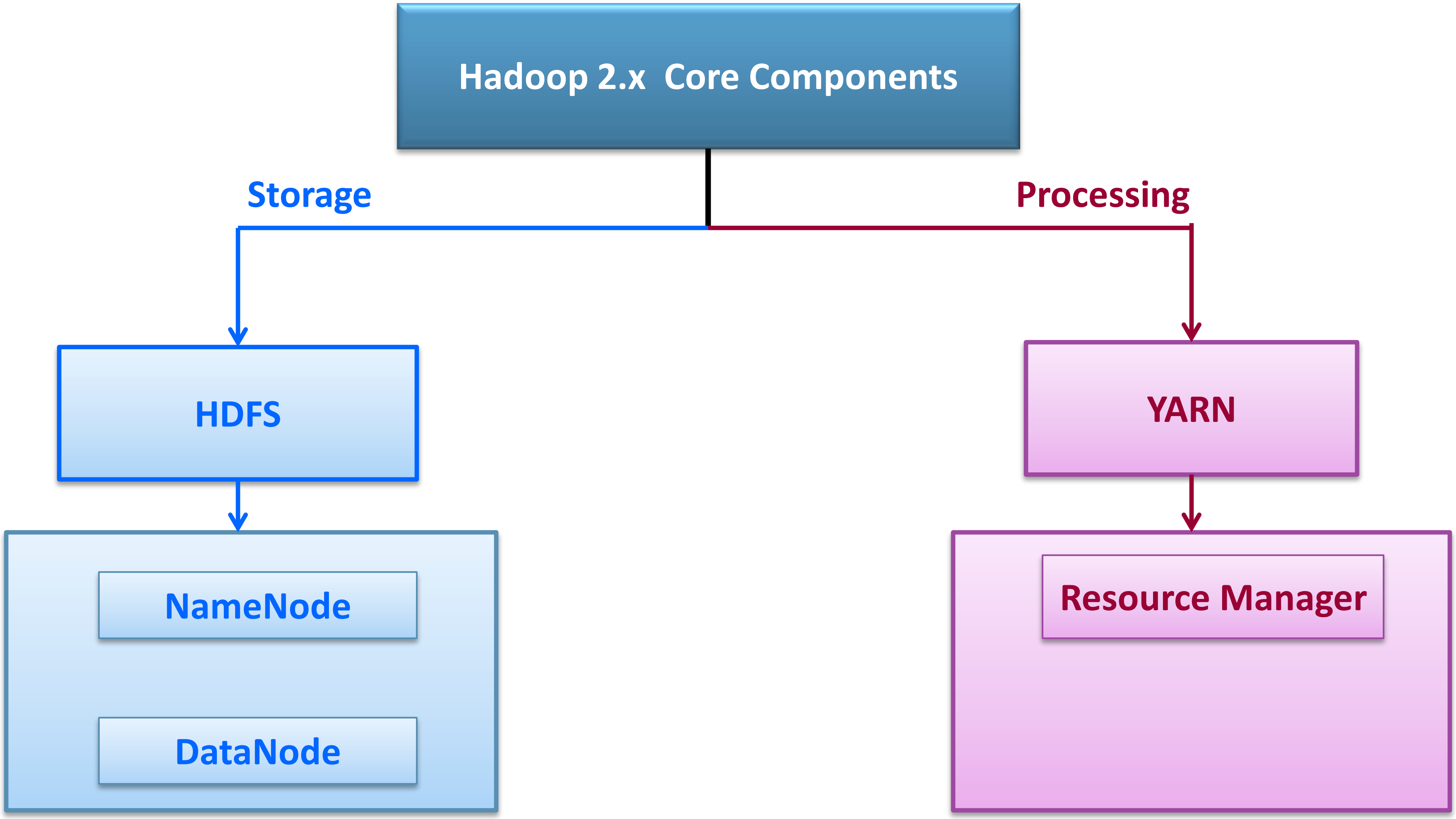
Core Components of Hadoop Cluster



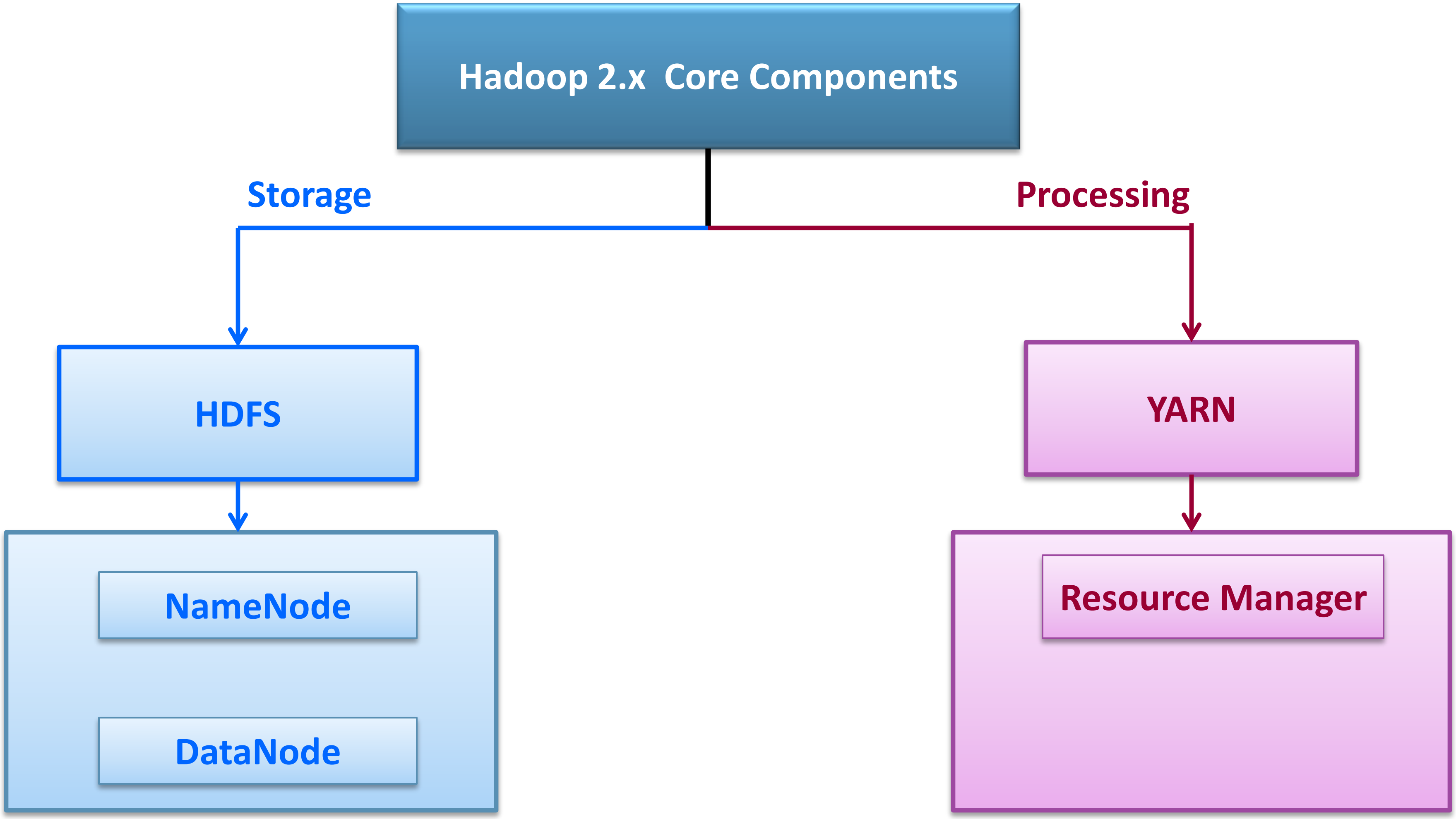
Core Components of Hadoop Cluster



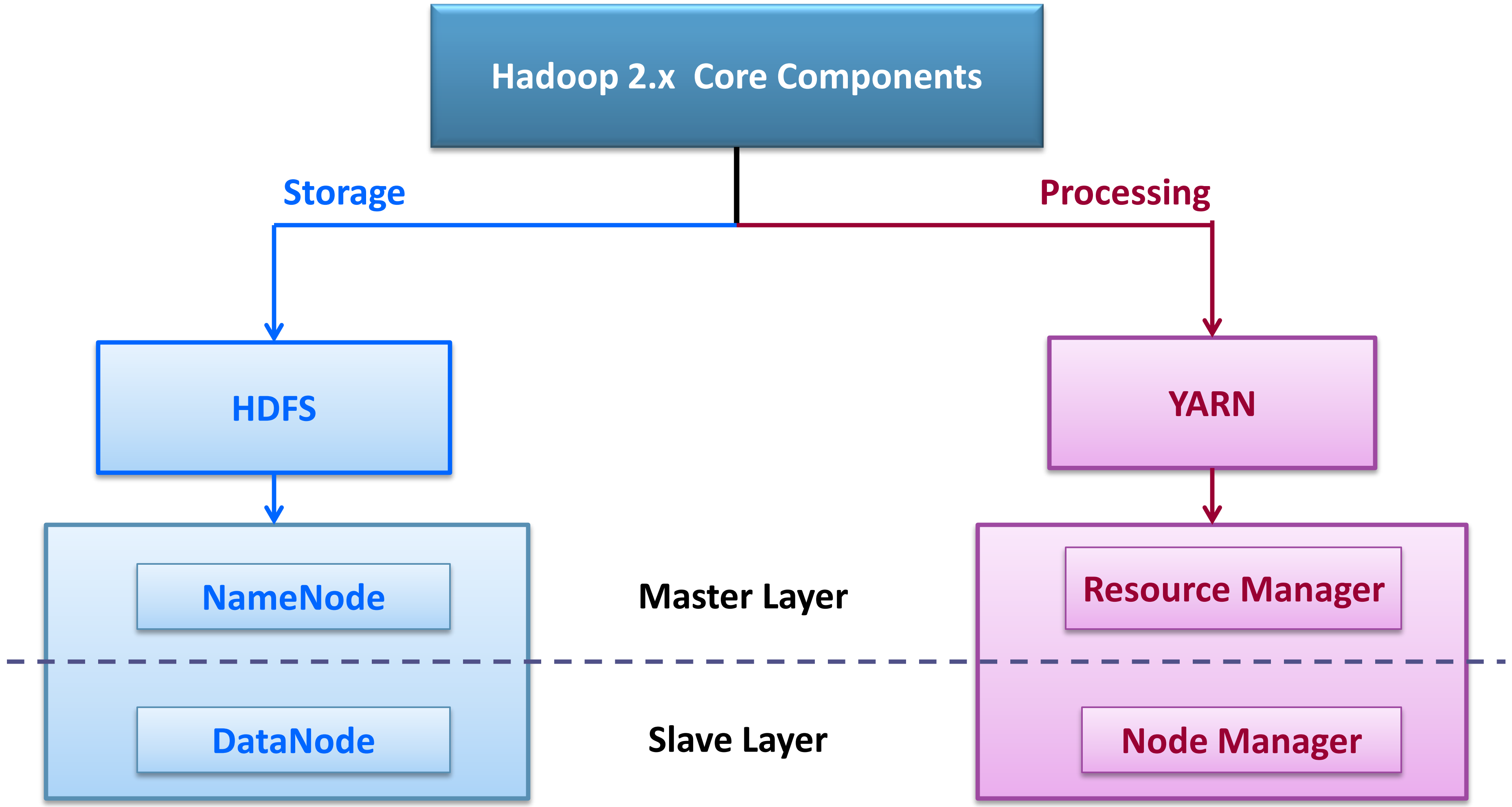
Core Components of Hadoop Cluster



Core Components of Hadoop Cluster

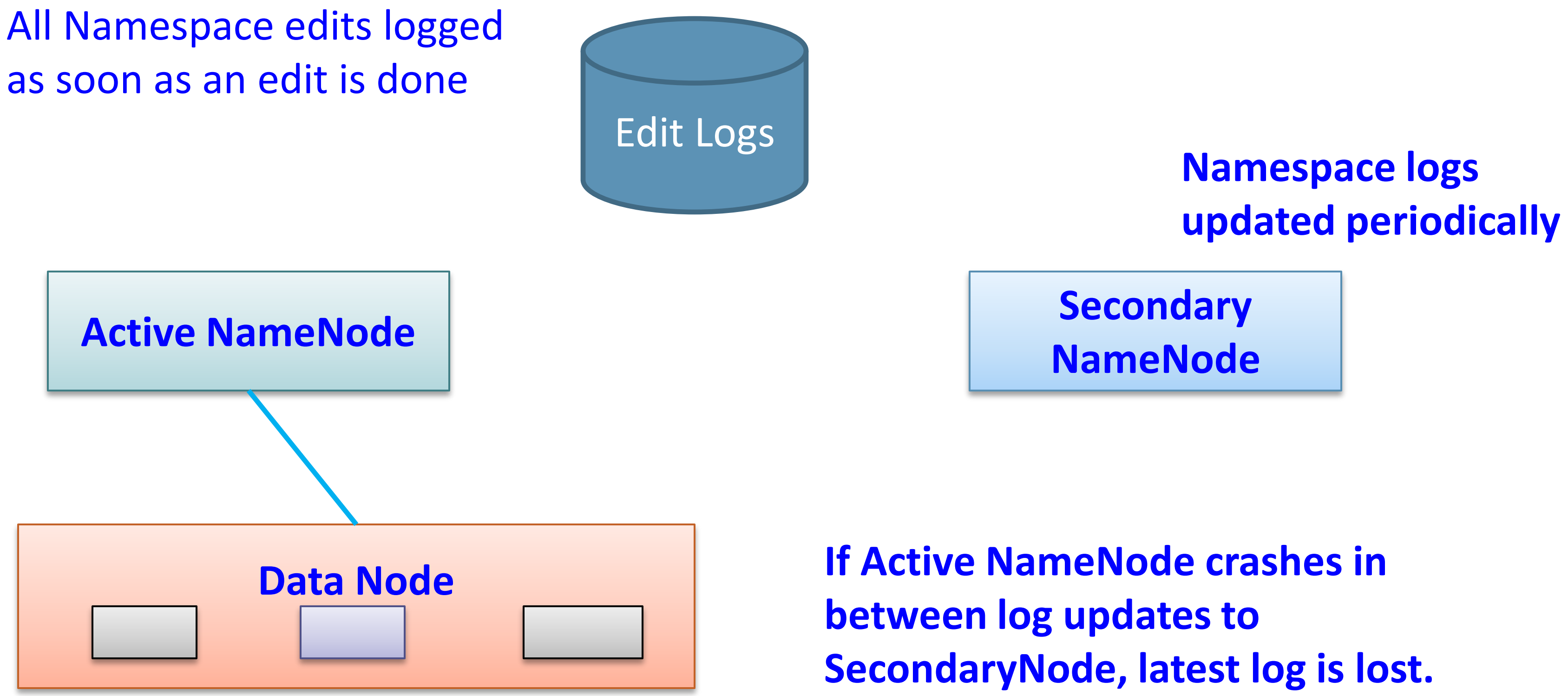


Core Components of Hadoop Cluster



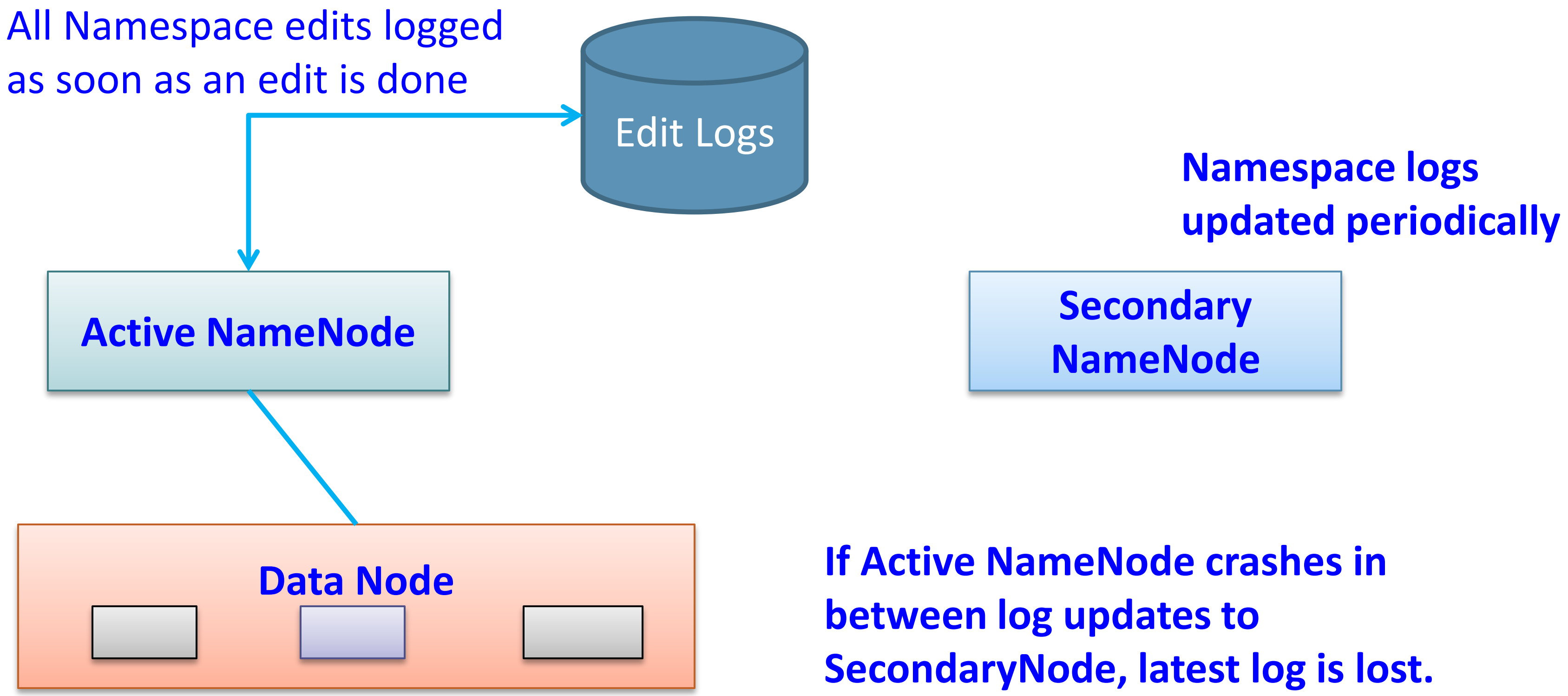
NameNode Backup in Hadoop1.x

- In Hadoop 1.x: Periodic updates of Edit Logs to Secondary NameNode are made:


All Namespace edits logged as soon as an edit is done
Edit Logs
Namespace logs updated periodically
Active NameNode
Secondary NameNode
Data Node
If Active NameNode crashes in between log updates to SecondaryNode, latest log is lost.

NameNode Backup in Hadoop1.x

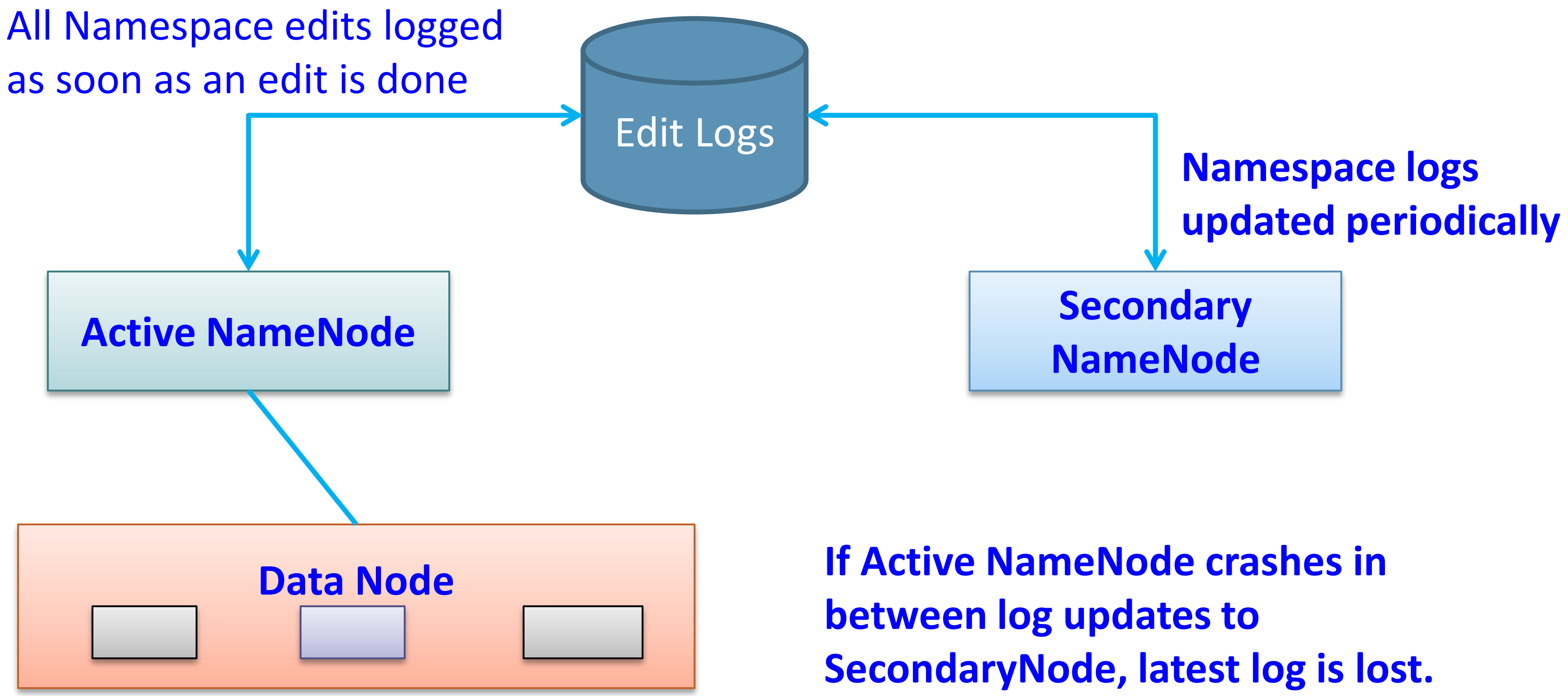
- In Hadoop 1.x: Periodic updates of Edit Logs to Secondary NameNode are made:


All Namespace edits logged as soon as an edit is done
Edit Logs
Namespace logs updated periodically
Active NameNode
Secondary NameNode
Data Node

If Active NameNode crashes in between log updates to SecondaryNode, latest log is lost.

NameNode Backup in Hadoop1.x

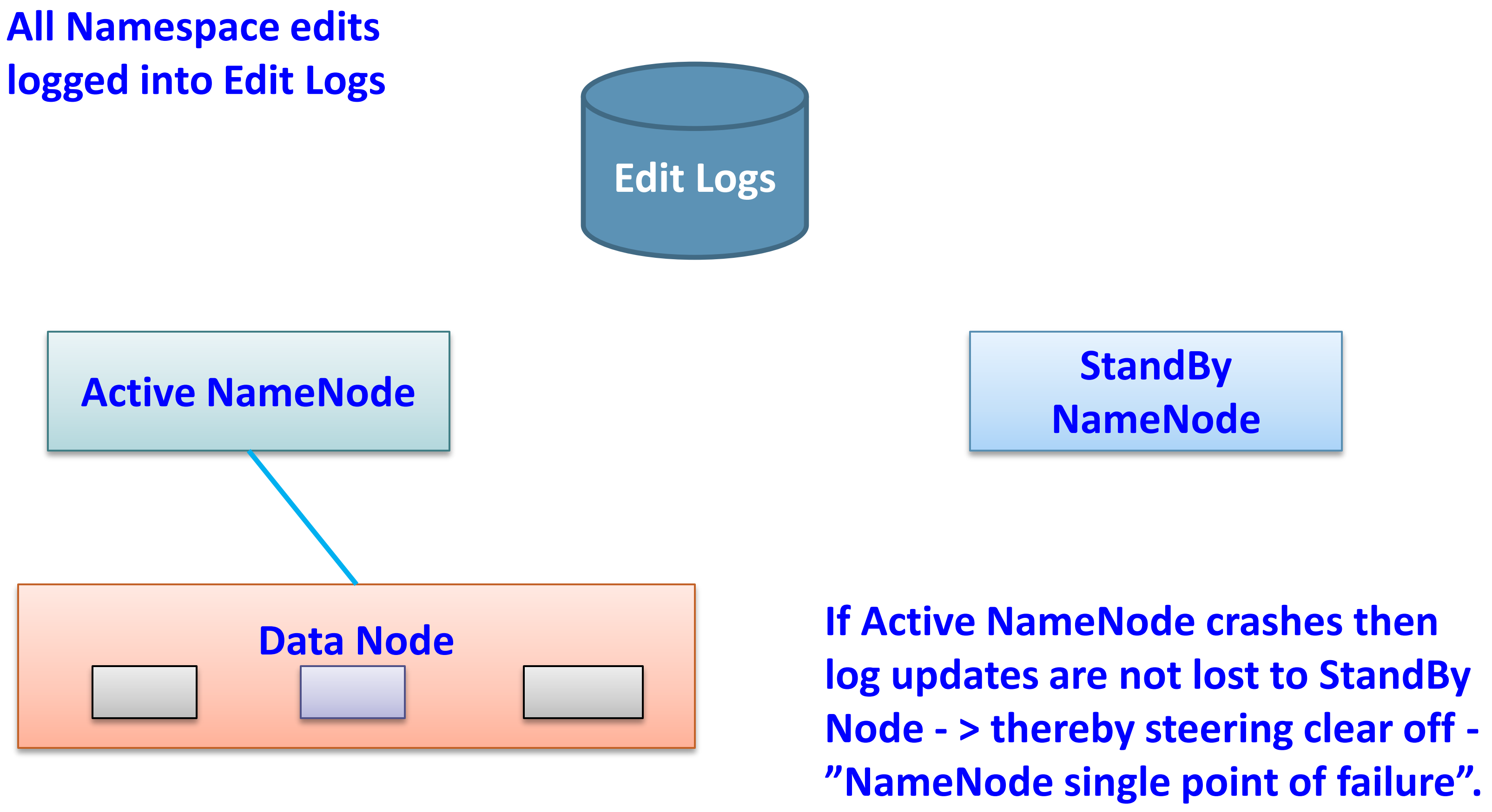
- In Hadoop 1.x: Periodic updates of Edit Logs to Secondary NameNode are made:


All Namespace edits logged as soon as an edit is done
Edit Logs
Namespace logs updated periodically
Active NameNode
Secondary NameNode
Data Node

If Active NameNode crashes in between log updates to SecondaryNode, latest log is lost.

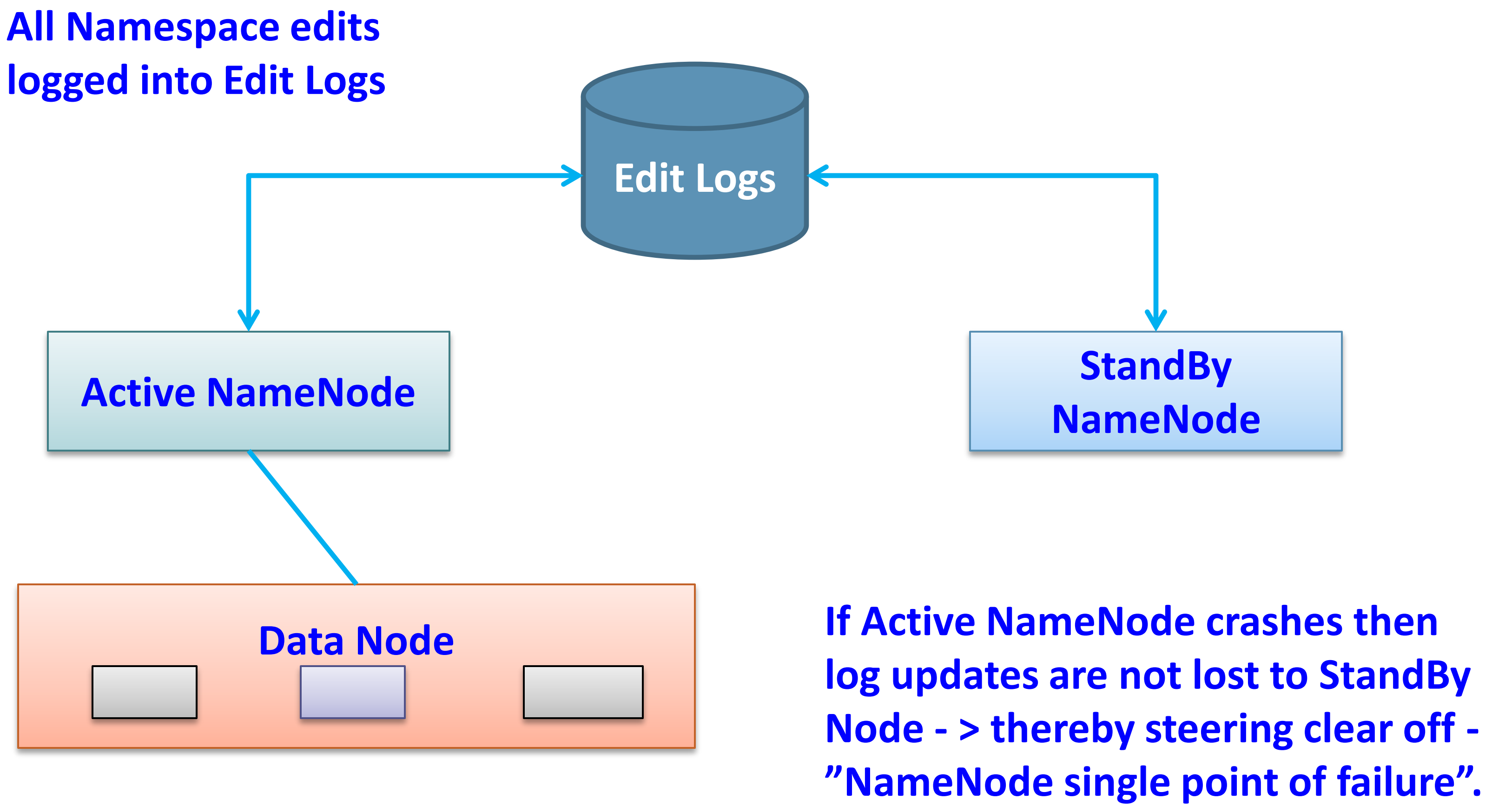
HDFS - High Availability Feature in Hadoop2.x

- StandBy NameNode in Hadoop 2.x:


All Namespace edits logged into Edit Logs
Edit Logs
Active NameNode
StandBy NameNode
Data Node
If Active NameNode crashes then log updates are not lost to StandBy Node - > thereby steering clear off - ”NameNode single point of failure”.

HDFS - High Availability Feature in Hadoop2.x

- StandBy NameNode in Hadoop 2.x:

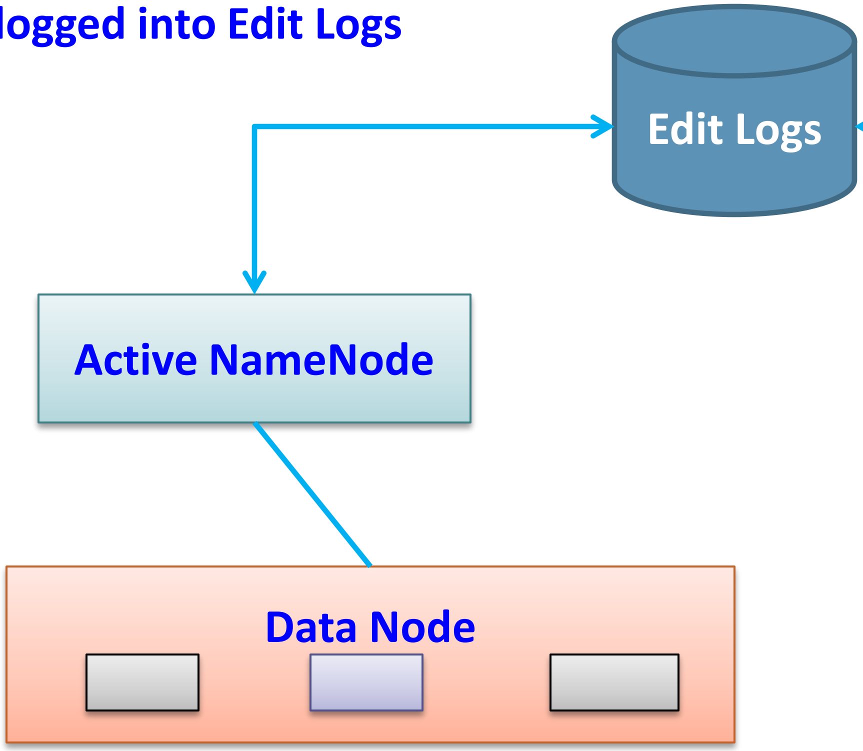

All Namespace edits logged into Edit Logs
Edit Logs
Active NameNode
StandBy NameNode
Data Node
If Active NameNode crashes then log updates are not lost to StandBy Node - > thereby steering clear off - ”NameNode single point of failure”.

HDFS - High Availability Feature in Hadoop2.x

- StandBy NameNode in Hadoop 2.x:

All Namespace edits logged into Edit Logs

Namespace logs (meta) updated simultaneously in StandBy NameNode from Edit Logs

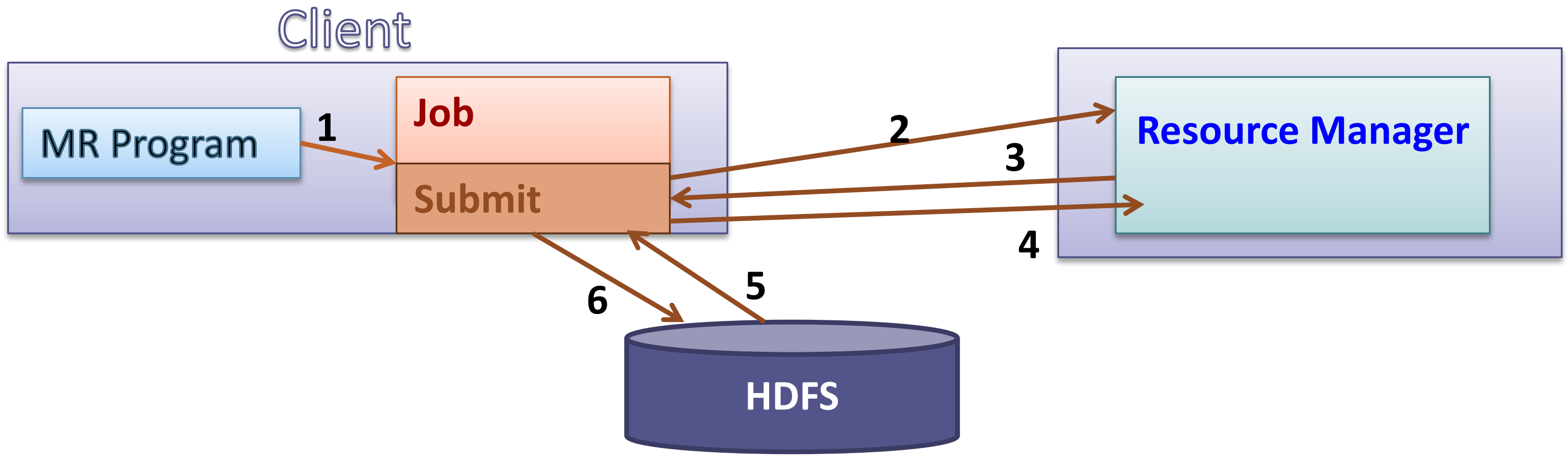


If Active NameNode crashes then log updates are not lost to StandBy Node - > thereby steering clear off - ”NameNode single point of failure”.

Introduction to a YARN Application

- In YARN a job is called an Application.
- For every job submitted to Resource Manager; it returns an acknowledgement in the form of an Application or AppId.
- YARN uses two daemons to launch & monitor containers:
  - Resource Manager
  - Node Manager
- A Container executes an application using constrained resources – CPU, memory etc.

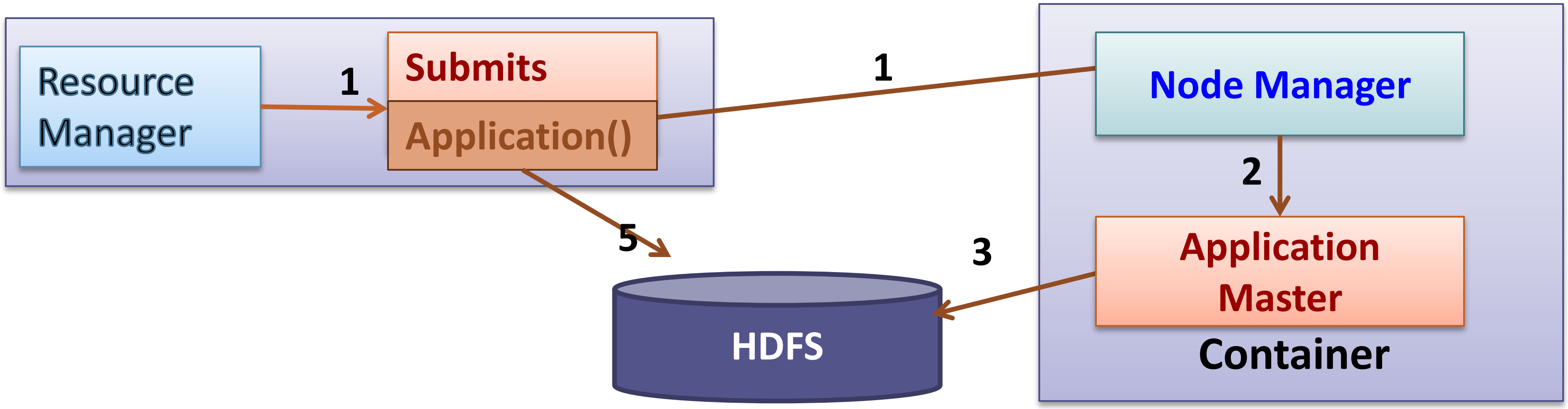
ANATOMY of a YARN Application
Phase 1: Job Submission



Process of Job Submission:

1. Client calls the Submit() method on Job.
2. Job Submit() requests meta data from Resource Manager.
3. Resource Manager returns an Application Id.
4. Submit Application
5. Job Submit() gets the Input Splits from HDFS.
6. Job Submit() creates a directory in HDFS and copies into it the jar, Config files etc.

ANATOMY of a YARN Application
Phase 2: Job Initialization



Process of Job Initialization

1. Resource Manager finds a Node Manager and submits to it an Application() method to start the Application Master.
2. Node Manager launches Application Master in a Container.
3. Application Master initializes the job by creating a number of book keeping objects to track the job's progress.
4. Application Master retrieves Input Splits from HDFS.
5. Application Master now creates one map task per split & checks the mapreduce.job.reducers property and creates that many number of reducers.

Questions - Hadoop

- What is the approach to solve Big data problem?
- What are the core components of Hadoop?
- Do all nodes work all the time in Hadoop cluster?
- What is the role of Master Node in Hadoop Cluster?
- Can Slave node directly communicate with each other?
- How is the concept of High Availability more efficient than Secondary Name Node concept?

Next Session (3)

- HDFS Internals:
- Hadoop Distributed File System (HDFS):
  - Introduction
  - Design of HDFS

- Data Model:
  - HDFS Data Flow
  - Blocks in HDFS
  - HDFS high level architecture
  - Processing constraint with Hadoop physical block
  - Processing on Input Split.
  - Relation between Hadoop Block and Split
  - HDFS File-Write

Get In touch with AcadGild

Contact Info:

- **Website :** [http://www.acadgild.com]
- **LinkedIn :** https://www.linkedin.com/company/acadgild
- **Facebook :** https://www.facebook.com/acadgild
- **Support:** support@acadgild.com