INTRODUCTION TO YARN

YARN

- **Hadoop 2**
- YARN
- YARN Applications
- MapReduce on YARN

Hadoop 1

- Initially, Hadoop (Hadoop1) was primarily designed only for storing large data sets (HDFS) and distributed processing of them (MapReduce)
- JobTracker
  - To setup a MapReduce cluster, in an Hadoop1 environment, we require a daemon called JobTracker, to which we submit MapReduce jobs, which
    - Allocates TaskTracker resources (schedule TastTrackers to run the submitted Map and Reduce tasks)
    - Manages the job itself

Hadoop 1

- Scheduling & Management
  - A **JobTracker** tries to make task allocations based on data locality (that is, it tries to allocate tasks that are colocated with input splits)
- Other Applications
  - When a Hadoop cluster is used for purposes other than running MapReduce, we require another resource allocation & application management mechanism particularly designed for that application

Hadoop 2

- The problem is that it is not easy to allocate common cluster resources to diverse applictions by using a per-application resource allocation mechanism, which causes resources of a Hadoop cluster to become underutilized, and hard to manage
- Fundamentally, resource allocation is a cluster-wide problem whereas managing distributed applications is application-specific
- To address these problems, Hadoop developers proposed a radical change (architecturally), which we now call Hadoop2, to separate the processes of
  - Managing and Allocating Cluster Resources
  - Managing Applications

YARN

- Hadoop 2
- **YARN**
- YARN Applications
- MapReduce on YARN

YARN

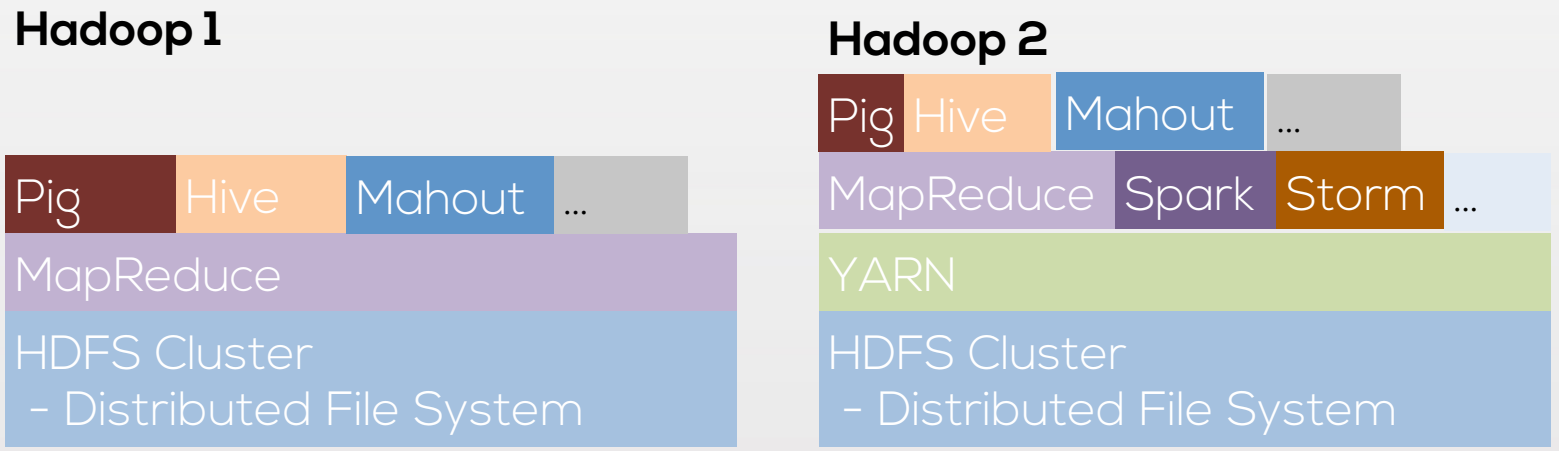
- With Hadoop 2 was introduced YARN (stands for Yet Another Resource Negotiator), a general purpose service for management of cluster resources, separating
  - Resource Management
  - Application Management

  processes
- YARN is like the operating system of a Hadoop cluster, allowing applications not bothering of managing the cluster resources

YARN

- YARN is a revolutionary change in the Hadoop world, making designing distributed processing frameworks (on top of HDFS) easier–M/R is not part of the core any more, it is just another YARN app
- Better, not much changed from the client perspective, since it is the applications who are responsible for negotiating resources, clients submit their job to the applications
- YARN is more of a foundational change in the Hadoop ecosystem

Hadoop 1 vs Hadoop 2



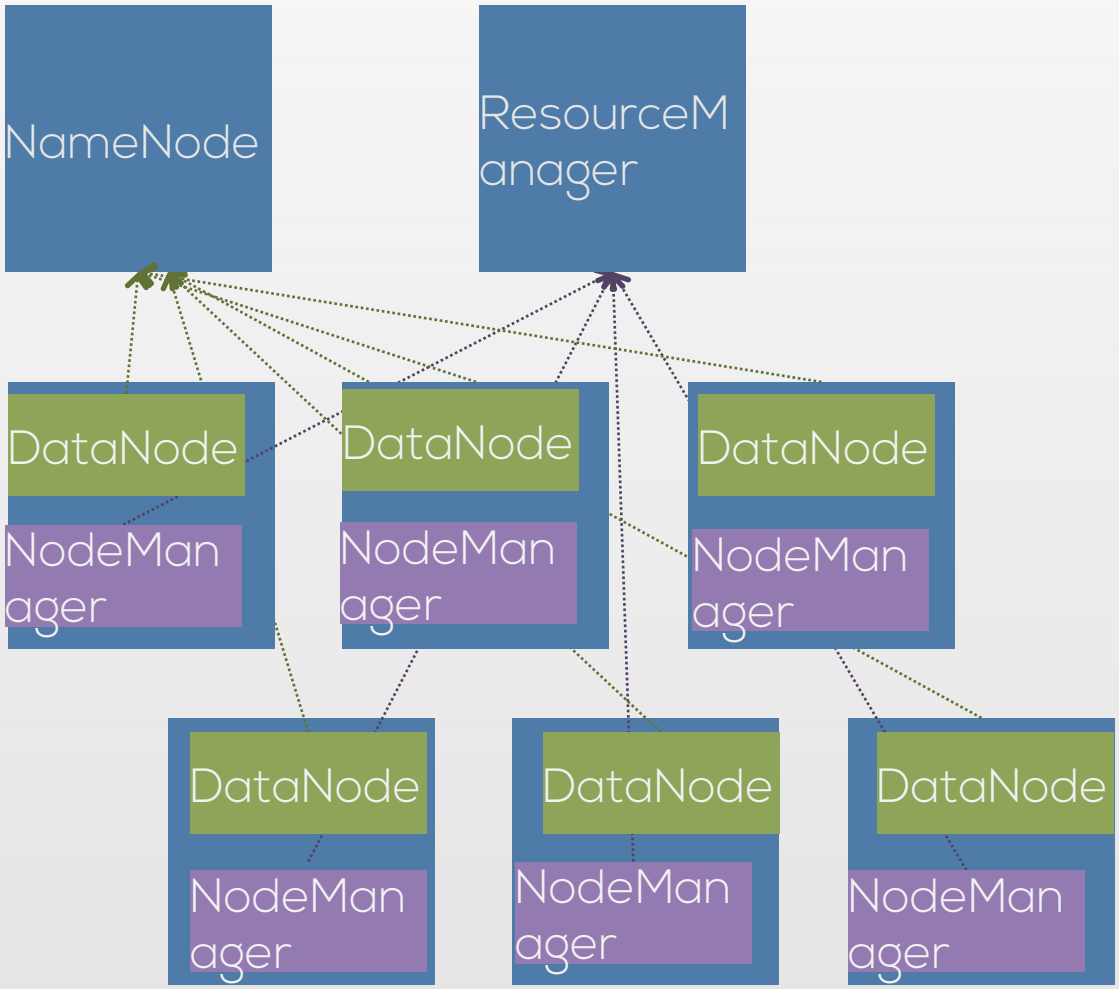
YARN

- For managing distributed applications, YARN consists of two main components:
  - ResourceManager (one per cluster)
  - NodeManager (one per node)
- Based on various constraints dictated by a pluggable Scheduler, YARN **`ResourceManager`** is responsible for allocating resources to competing applications
- The **`NodeManager`**s are per-node agents taking care of a cluster node. Its responsibilities include
  - Communicating with the **`ResourceManager`**
  - Launching Containers and monitoring them

YARN

- A container represents a set of resources (CPU, memory, …) within a task is executed
- So once a ResourceManager (per cluster) and a set of NodeManagers (per node) are designated and set up, the YARN cluster is ready to run
- In a cluster, NodeManagers are typically colocated with the DataNodes, and the ResourceManager is a master node (like the NameNode)

YARN


NameNode
ResourceManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager

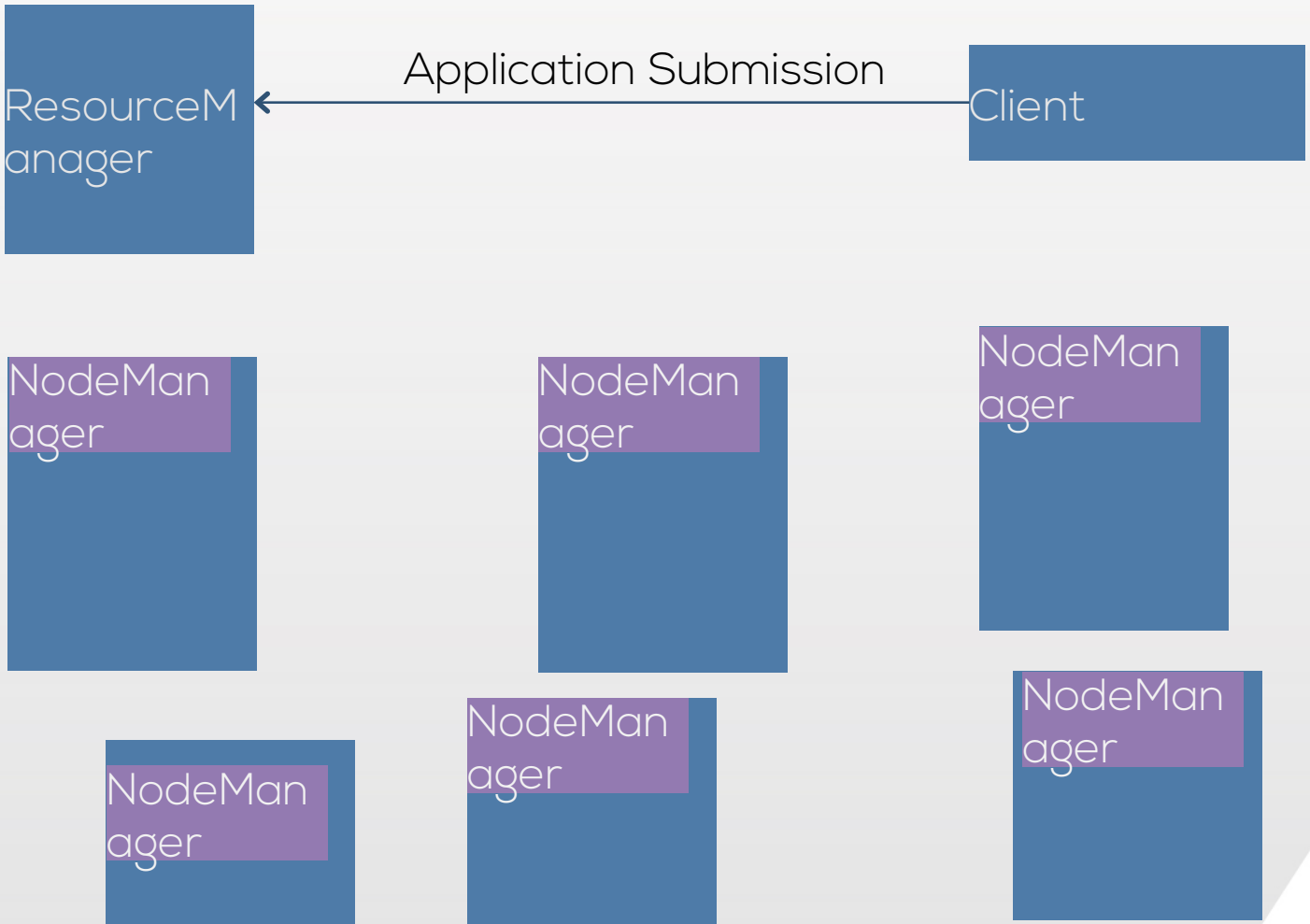
YARN

- Hadoop 2
- YARN
- **YARN Applications**
- MapReduce on YARN

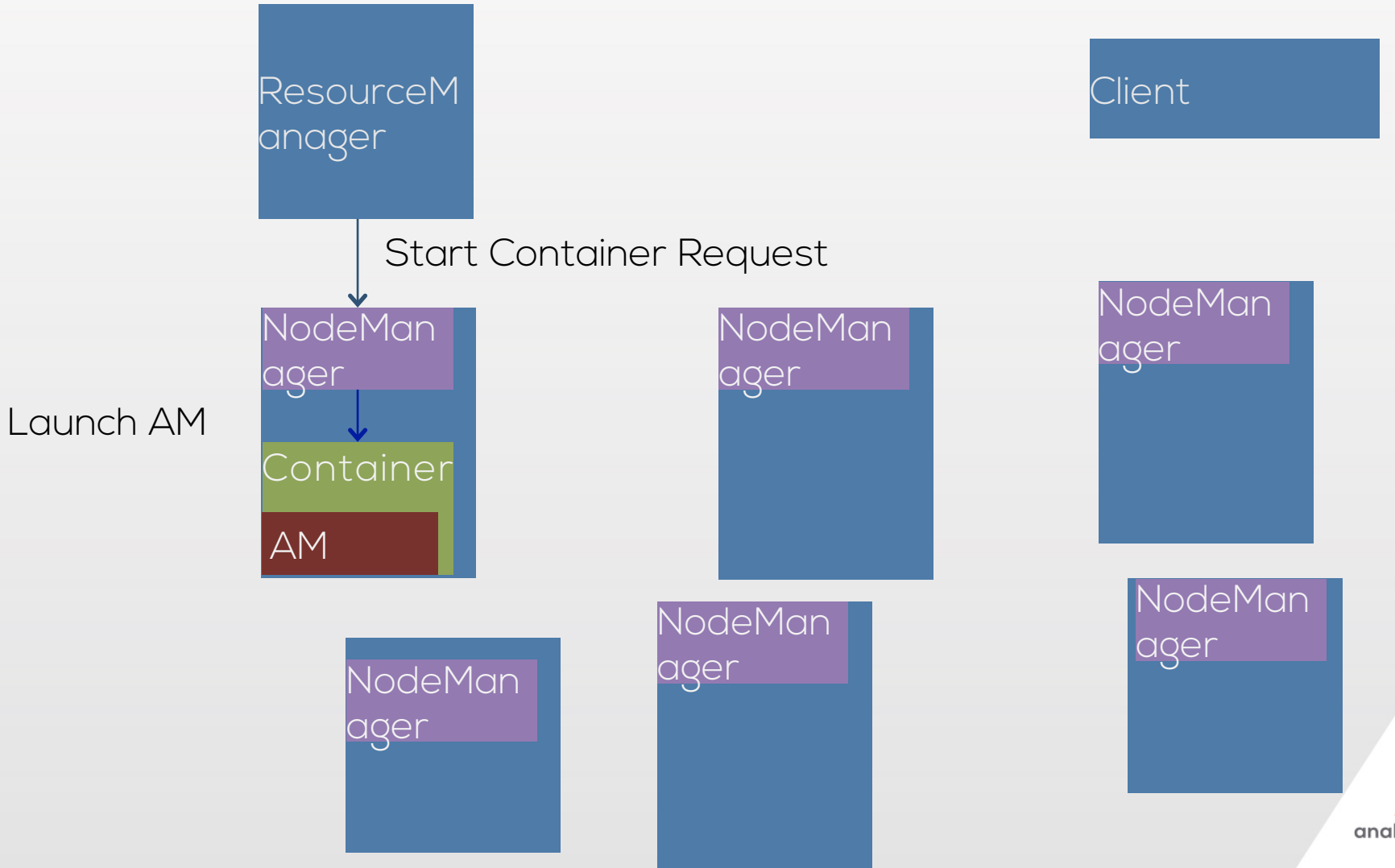
YARN Applications

- A YARN application is specified by an **`ApplicationMaster,`** and it is either a job or a DAG of jobs
- The client asks the **`ResourceManager`** to run an application-specific **`ApplicatonMaster`** process, which assigned to a **`NodeManager`** that can launch the AM (this is Container-0)
- Once the ApplicationMaster running, it can request more containers from the ResourceManager, and use them to run distributed tasks

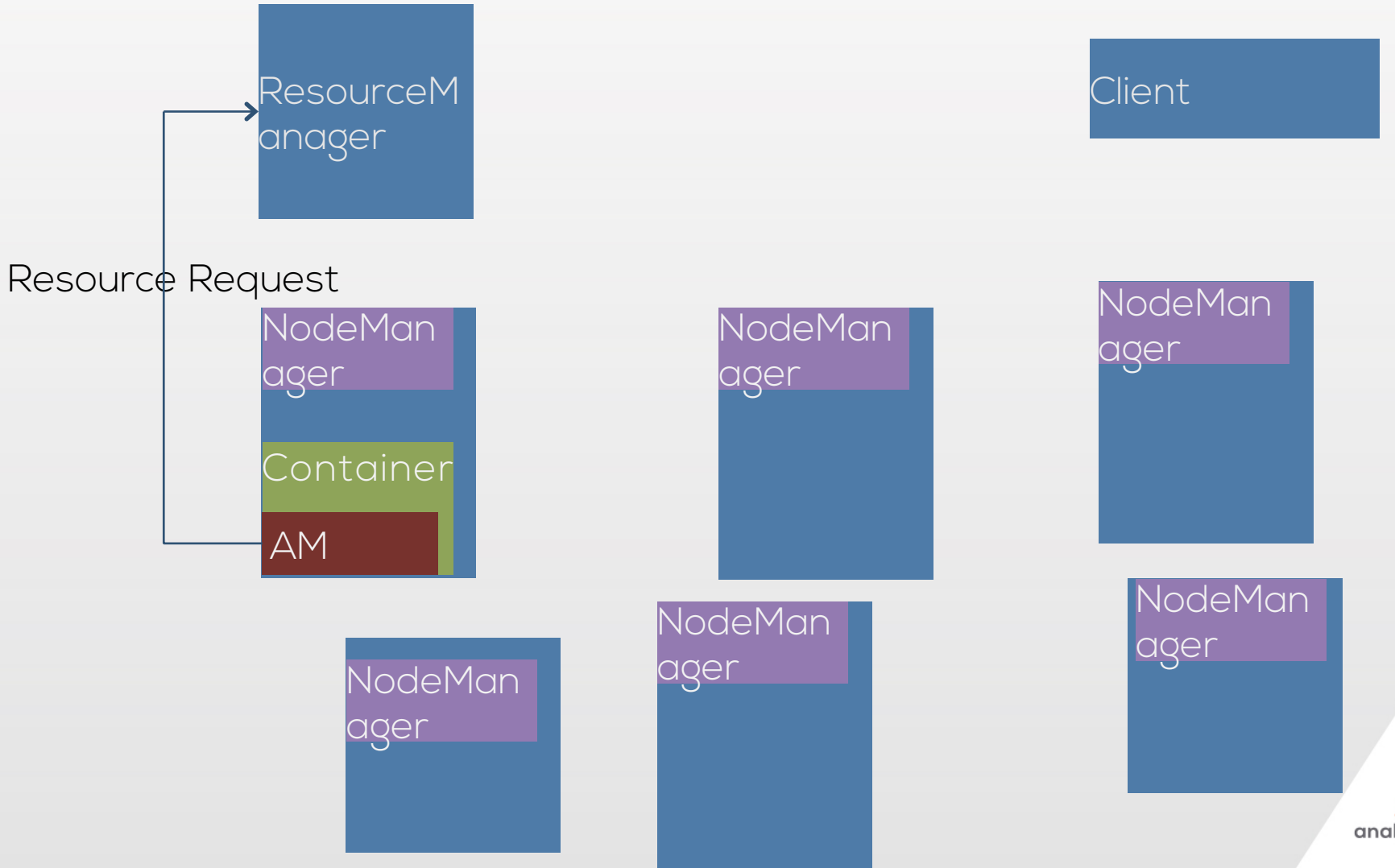
YARN Applications



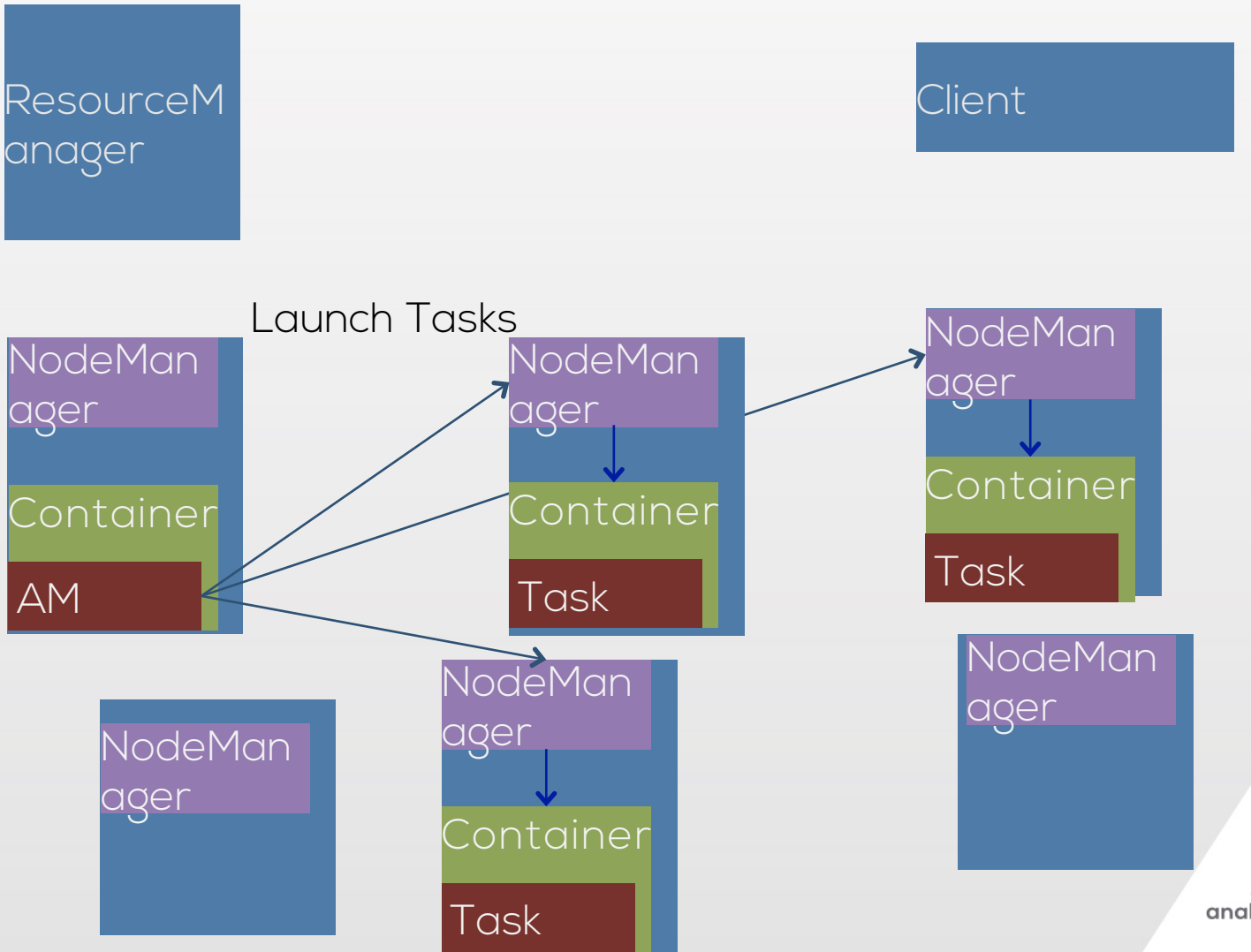
YARN Applications



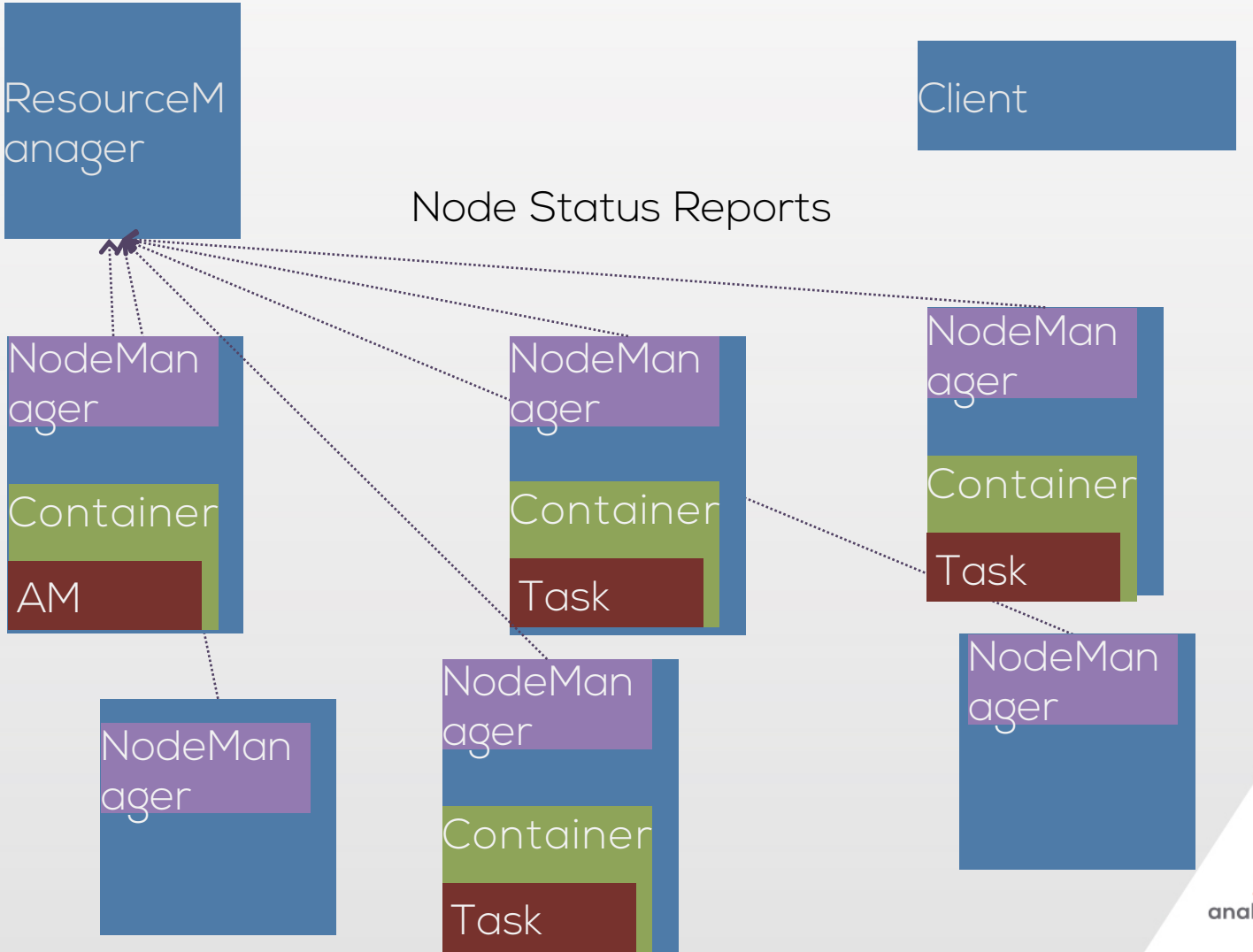
YARN Applications



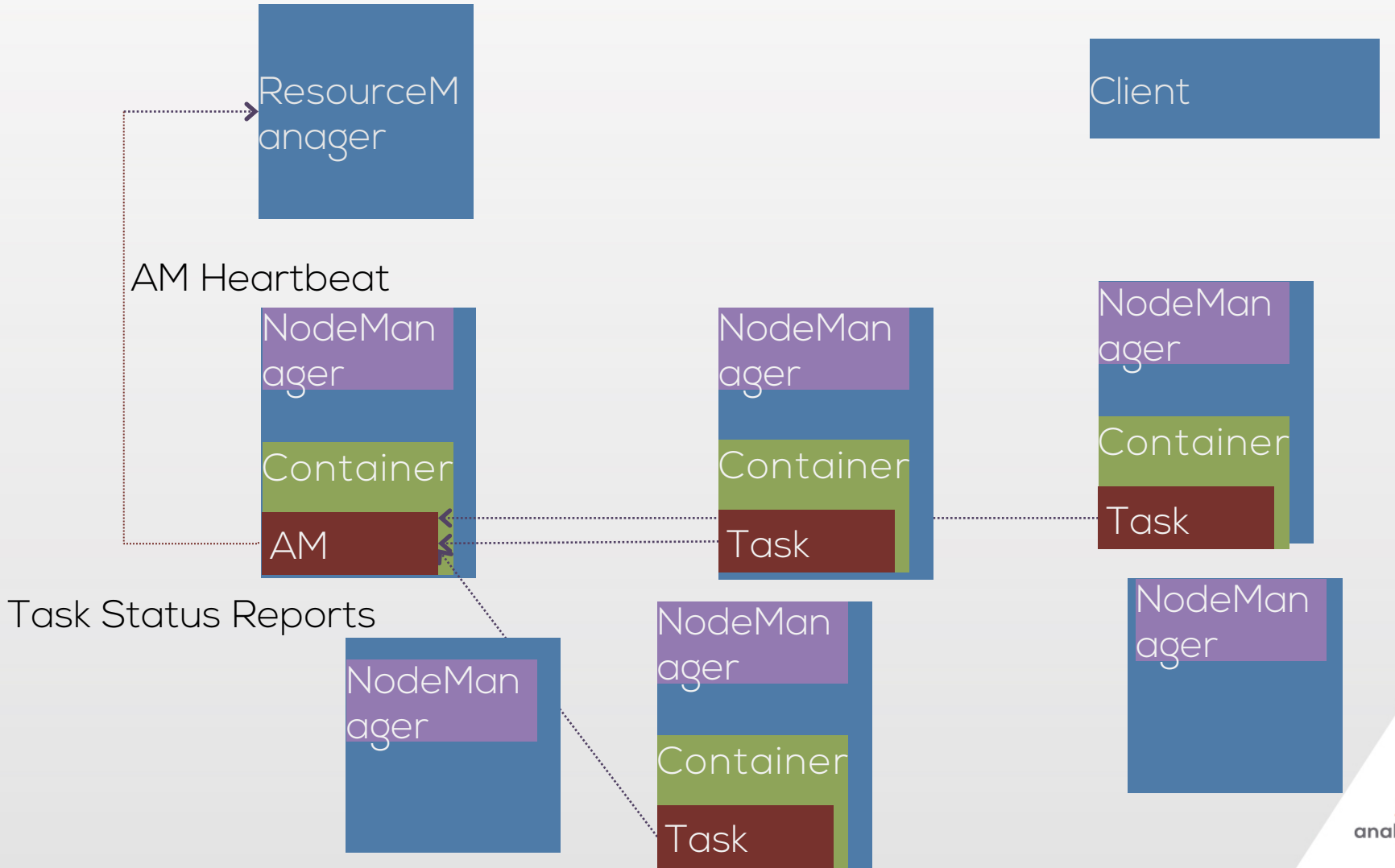
YARN Applications



YARN Applications



YARN Applications



YARN Applications

- A resource request is in terms of memory and CPU cores with locality constraints
  - Locality is critical for communication-efficient distributed processing

YARN

- Hadoop 2
- YARN
- YARN Applications
- **MapReduce on YARN**

MapReduce on YARN

- While running on YARN, the **`MRAppMaster`** first requests a bunch of Mapper containers
  - If any task fails, **`MRAppMaster`** requests new Mappers
- The **`MrAppMaster`** also requests **`numReducers`** number of Reducers, but in a low priority (and with a slow start—they don't start until 5% of Mappers are completd)
- YARN allows NodeManagers run Auxiliary Services, and in MapReduce, this is a web server that allows Reduce tasks to fetch Map outputs: the mapreduce_shuffle service (the **`ShuffleHandler`** class
  - This should be registered in the **`yarn-site.xml`** configuration (the property **`yarn.nodemanager.aux-services`**)

MapReduce on YARN

- To summarize, the MapReduce life cycle is as the following:
  - The client submits the **`Job`**, which copies the resources and input splits information to HDFS
  - **`Job`** internally asks YARN **`ResourceManager`** to run **`MRAppMaster`**
  - The **`MRAppMaster`** calculates number of tasks required (with the configured memory and CPU demands), and allocates resources
  - Once the resources are available, **`MRAppMaster`** asks corresponding **`NodeManagers`** to launch **`Containers`** running the **`YarnChild`** application, which then runs the Map or Reduce task

Introduction to Big Data Processing

End of Chapter