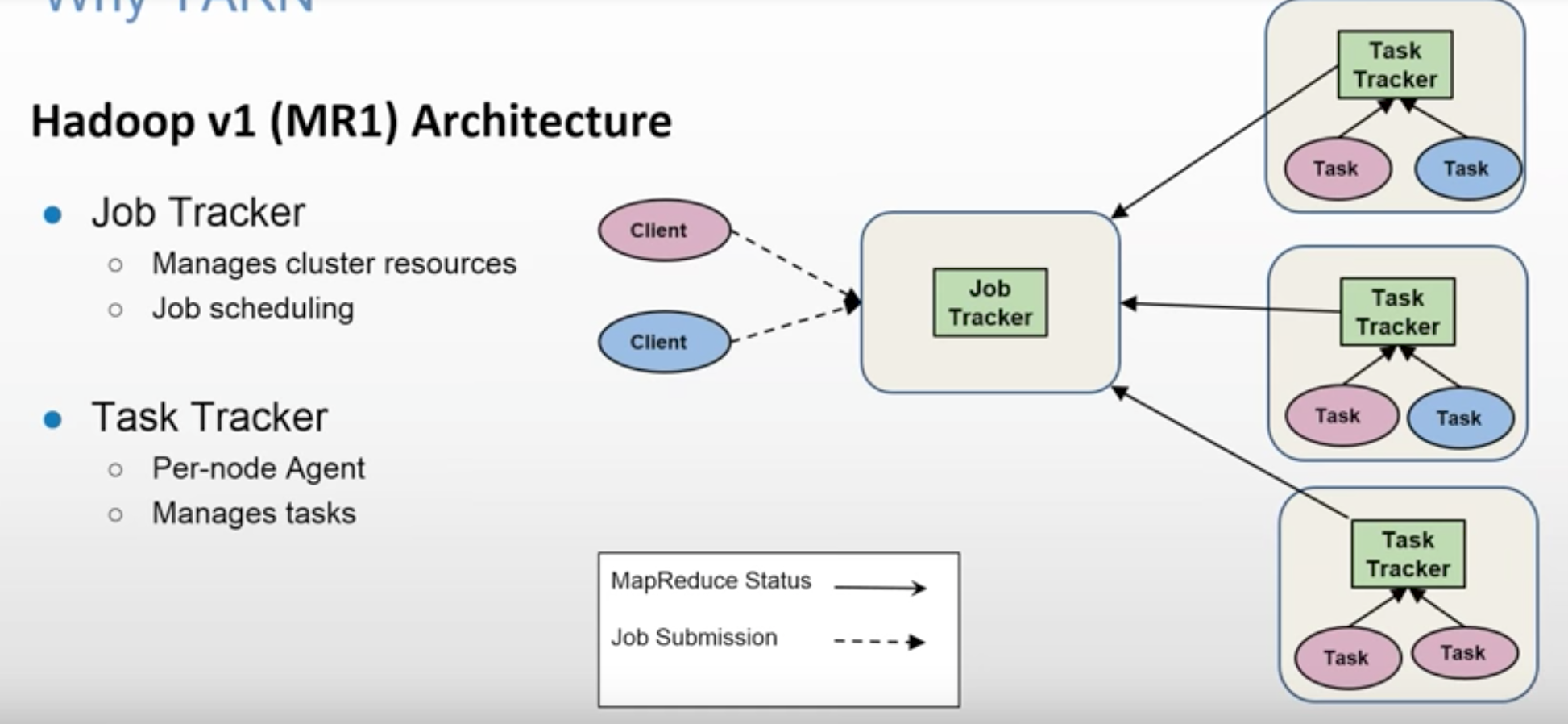
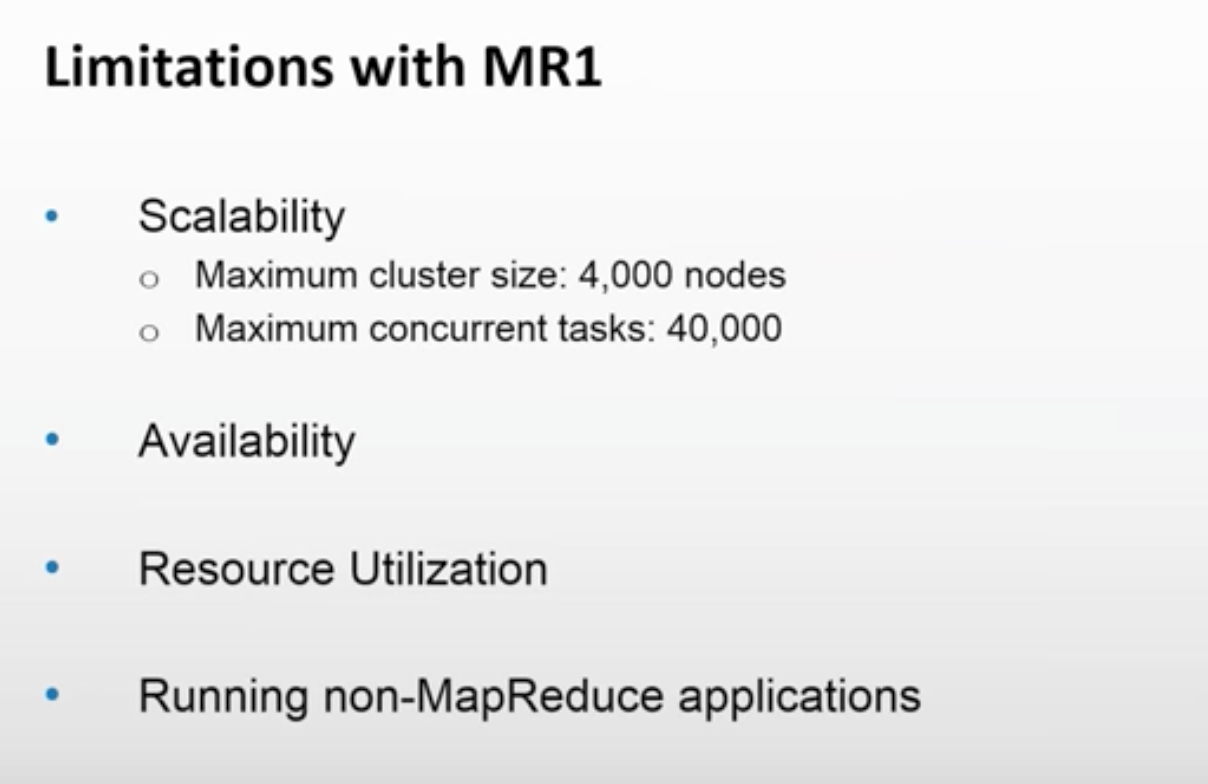
<https://www.youtube.com/watch?v=Rtd0gHl4PBo>



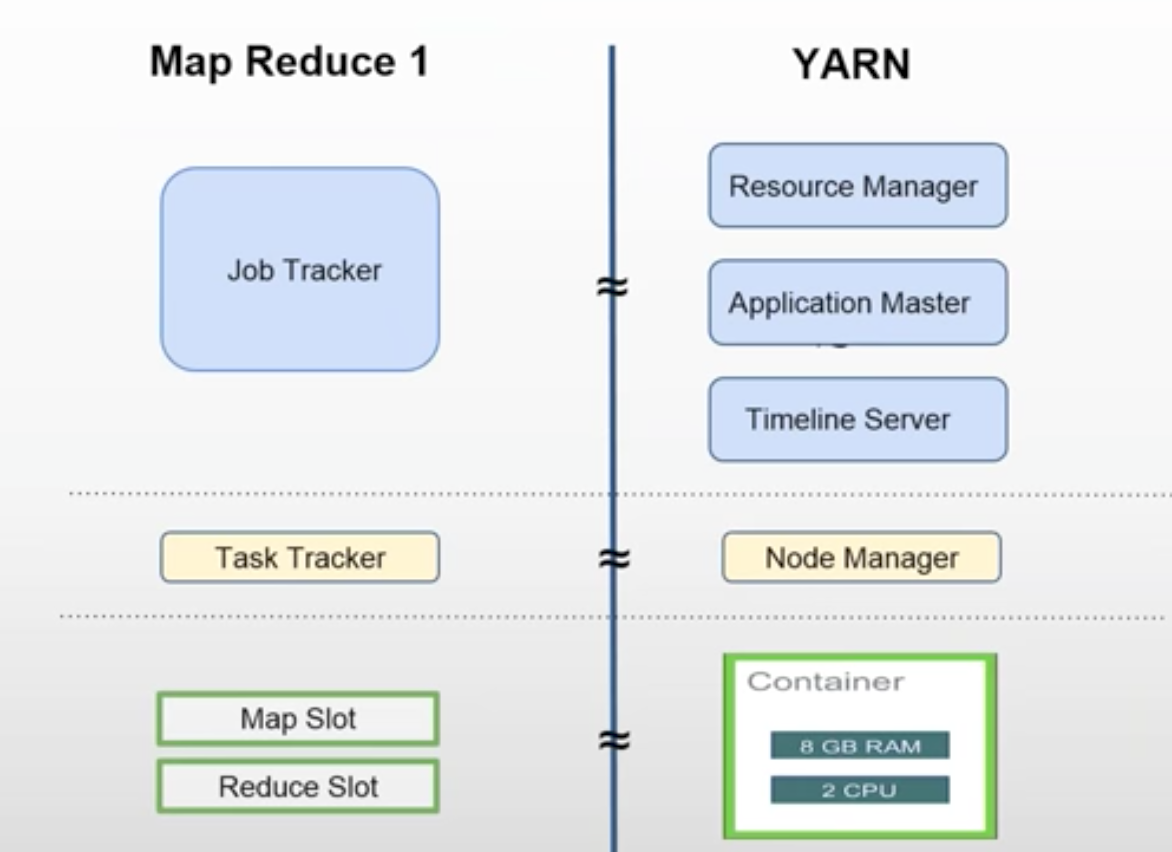
Limitations of Hadoop 1

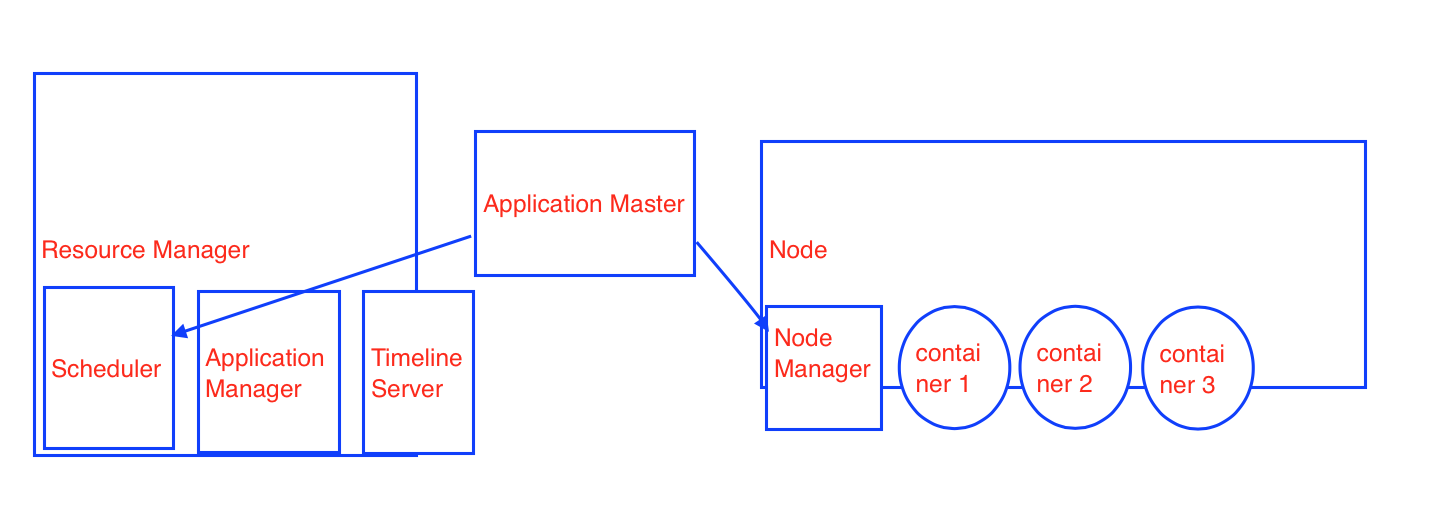


JobTracker is a SPOF. Only traditional MapReduce code can run on Hadoop V1. You cannot run other apps like Storm, Spark, Giraph etc.



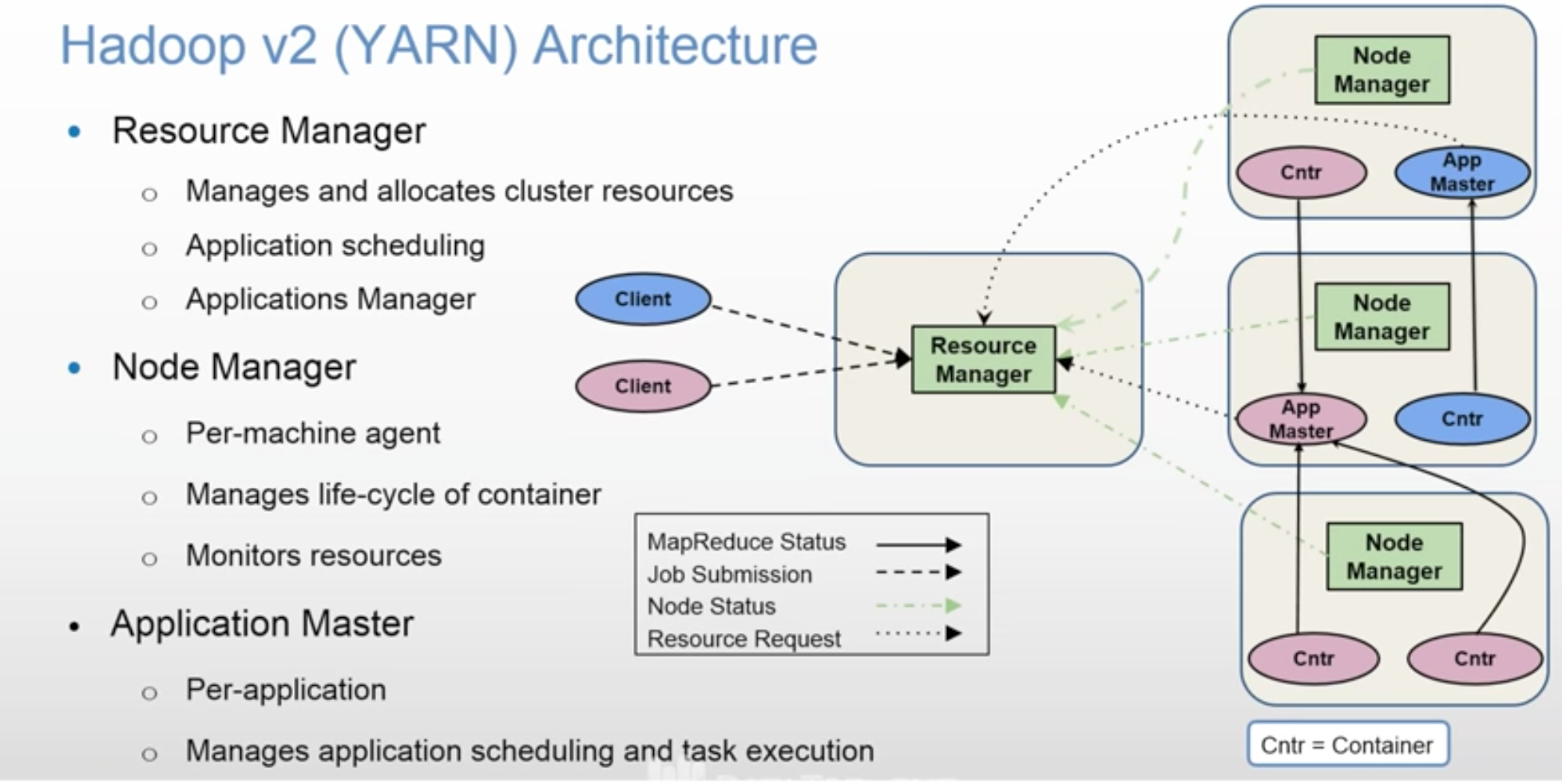
YARN has taken Hadoop beyond Batch Processing.



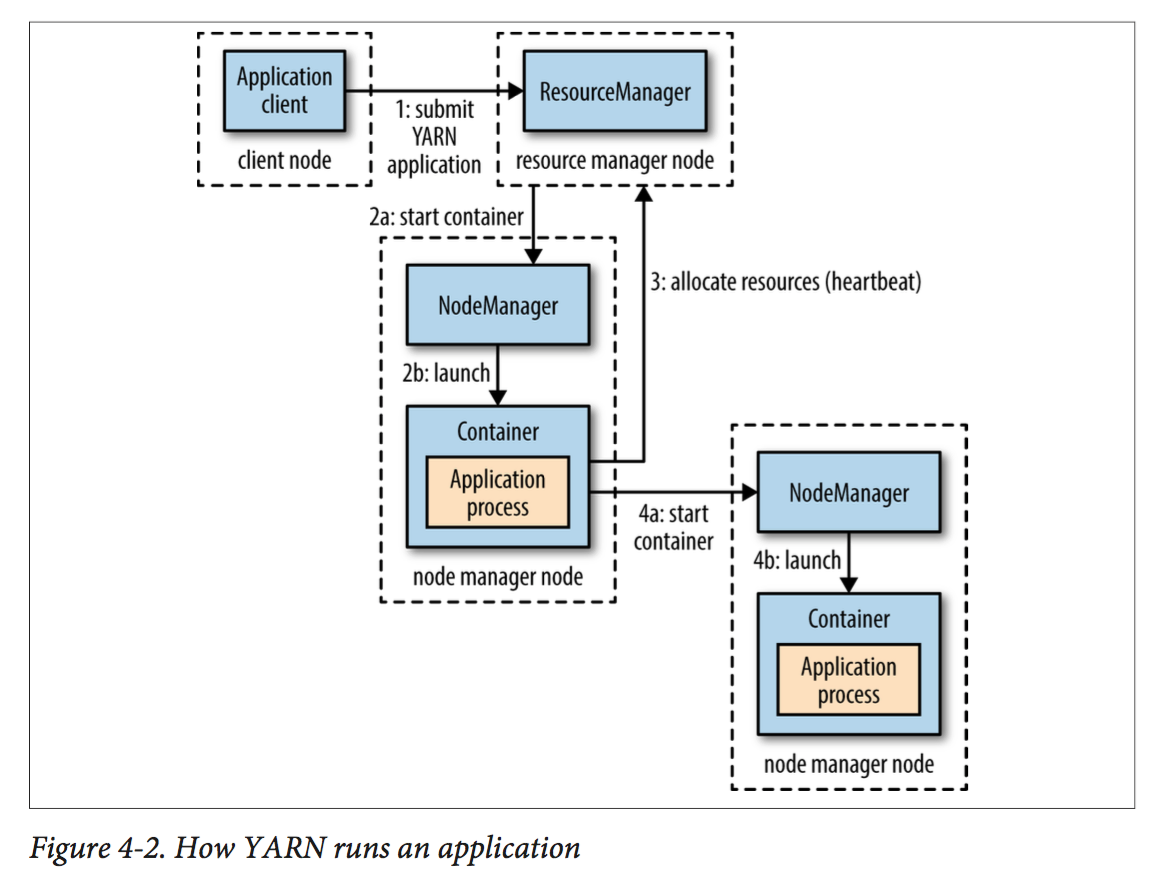


In Hadoop V1, Job Tracker used to many work. In YARN, responsibilities are divided.

Timeline Server keeps the history of all the jobs that were executed.



As there are two applications, there are two Application Masters shown in above diagram.



Application Master sits in between Resource Manager and Node Manager.

There is one Application Master per Application.

It negotiates number of containers from RM’s Scheduler and then contacts NMs on nodes and tells them to launch containers.

You submit a job to RM. RM’s scheduler schedules the job.

Node Manager keeps sending heart beats to RM, so that RM know what NMs are up. It also sends utilization of resources information to RM.

Node Manager is responsible to launch a container and killing it when job is over.

Scheduler will schedule the job in the cluster. So, your job’s state changes from

accepted to running.

3 different types of schedulers available:

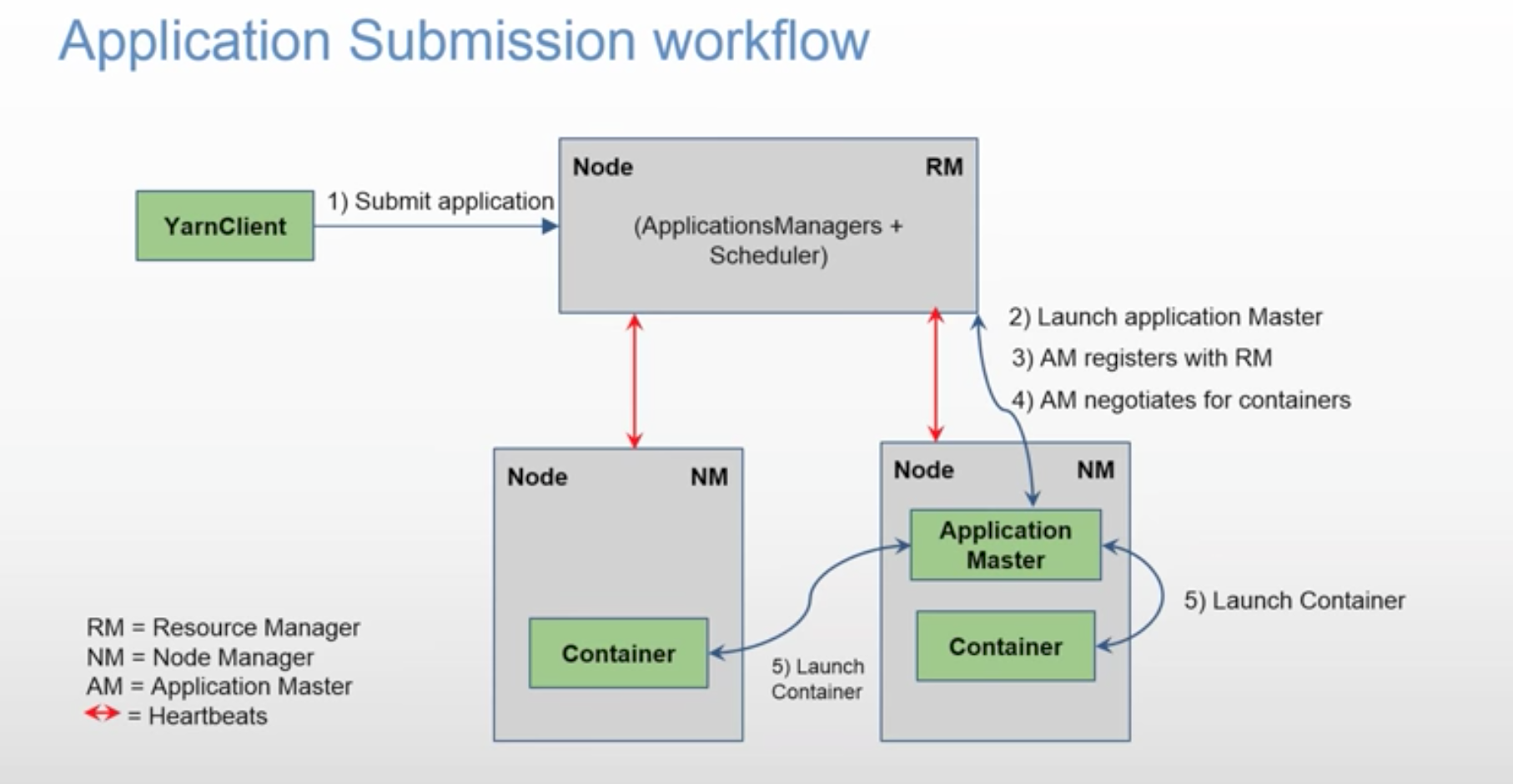
- FIFO

- Capacity

- Fair Share

You can configure it in yarn-site.xml

Timeline server keeps history of jobs that were executed.



Yarn Client submits a job to RM with some specification like how much cpu, memory etc required to launch an Application Master. Remember, only one Application Master is lauched per application.

RM’s Application Manager launches Application Master on a node that is available. Application Master also runs inside the container.

Application Master holds your MapReduce jar file.

Application Master will register itself to RM telling it is running on a particular node.

Application Master Negotiates containers with RM with so and so specifications of cpu, memory etc. Then it will ask Node Managers to launch containers.

Application Master will unregister itself from RM once the job is completed.

How the communication happens between RM and Application Master?

Application Master is a custom code. If it requires 3 containers, but RM has only 2 available, then RM can go back to Application Master saying come back after some time or you can write Application Master code in such a way that it can accept lower number of containers also.

YARN allows an application to specify locality constraints for the containers it is requesting. Locality constraints can be used to request a container on a specific node or rack, or anywhere on the cluster (off-rack).

Sometimes the locality constraint cannot be met, in which case either no allocation is made or, optionally, the constraint can be loosened.

As per the proximity and availability of the nodes, YARN schedules your job.

It tries to run the job on the closest node where block replica is available. If node is busy because of other containers running in it, then

it finds another node in the same rack. If that is also not available then it goes to a node on another rack.

In the common case of launching a container to process an HDFS block (to run a map task in MapReduce, say), the application will request a container on one of the nodes hosting the block’s three replicas, or on a node in one of the racks hosting the replicas, or, failing that, on any node in the cluster.

Failure:

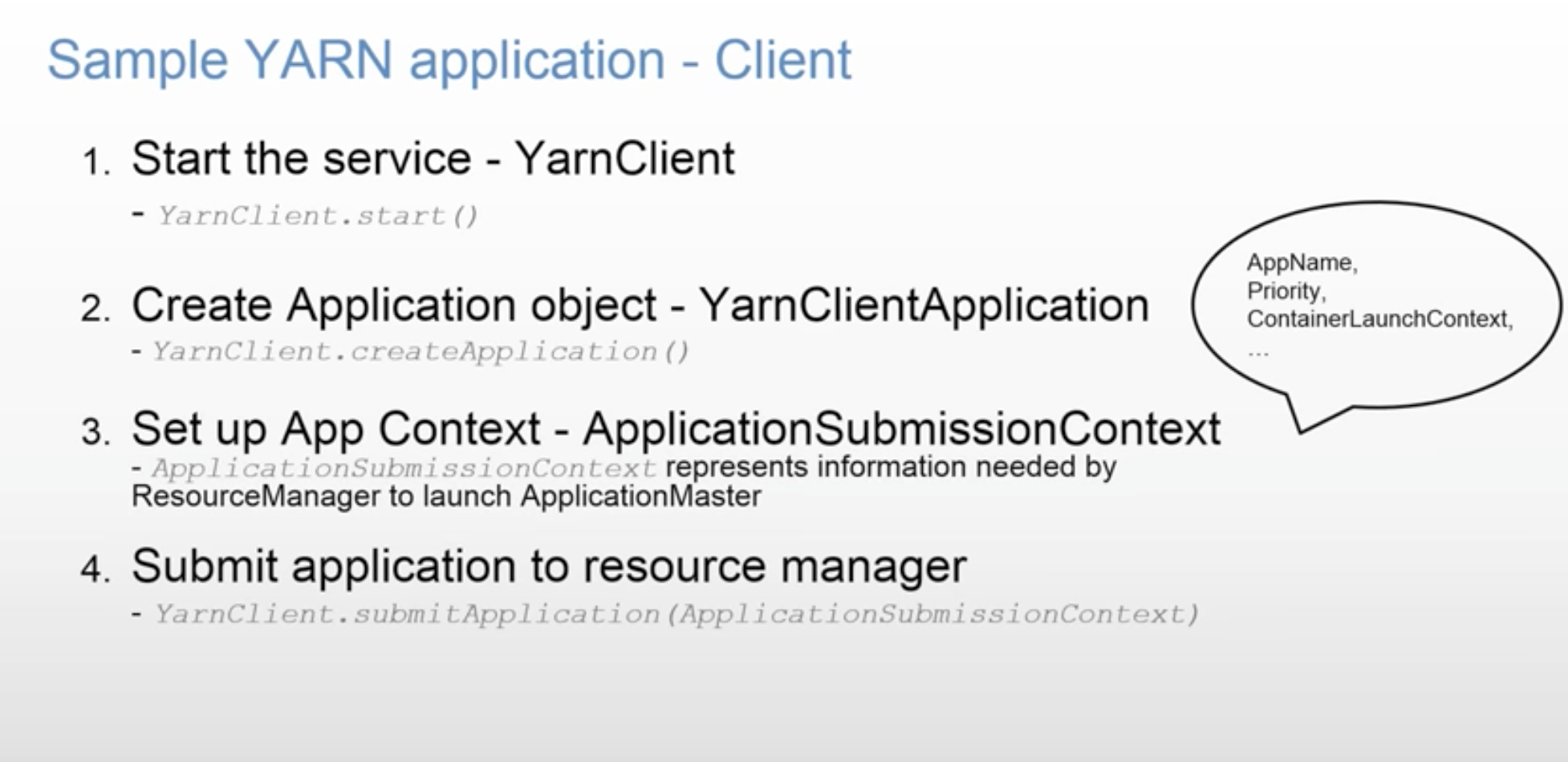
You can have Stand By RM. So, if primary RM fails, then Stand By RM is promoted as primary RM.

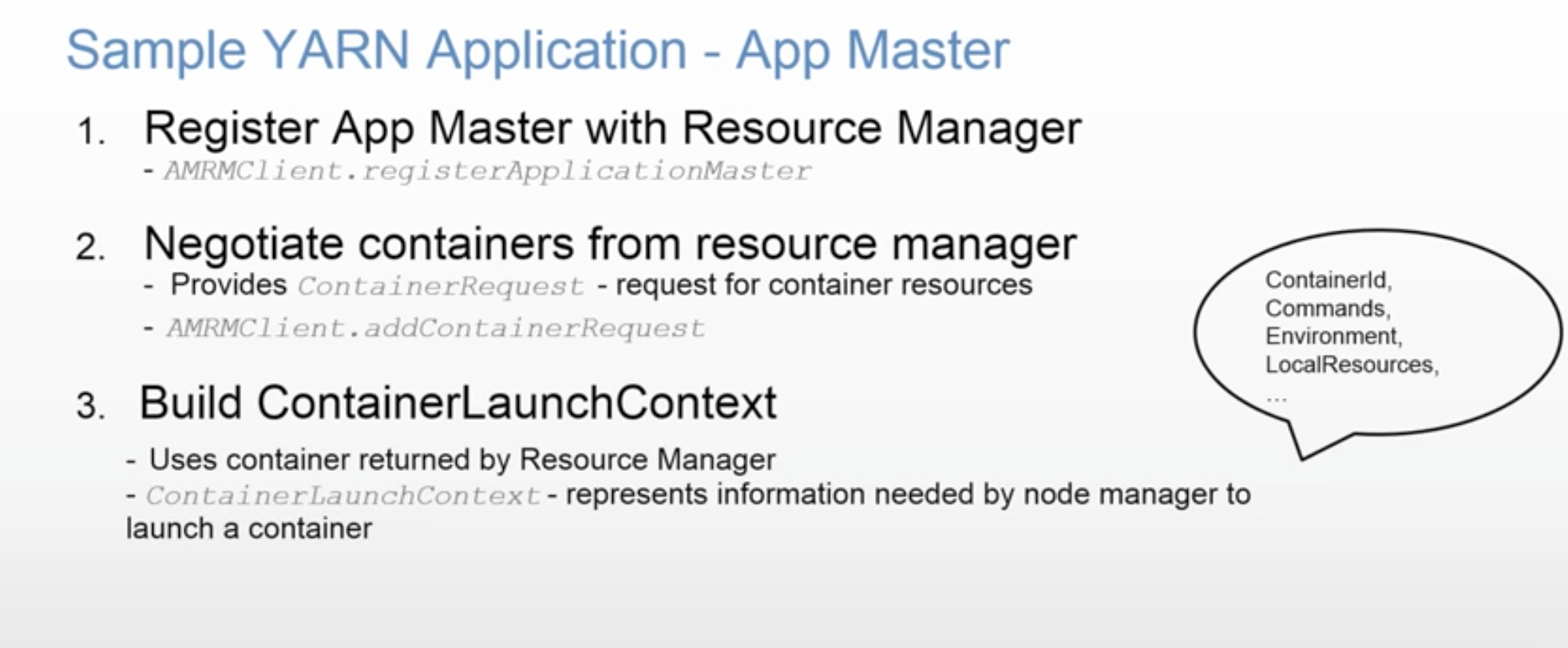
How does it handle long running jobs?

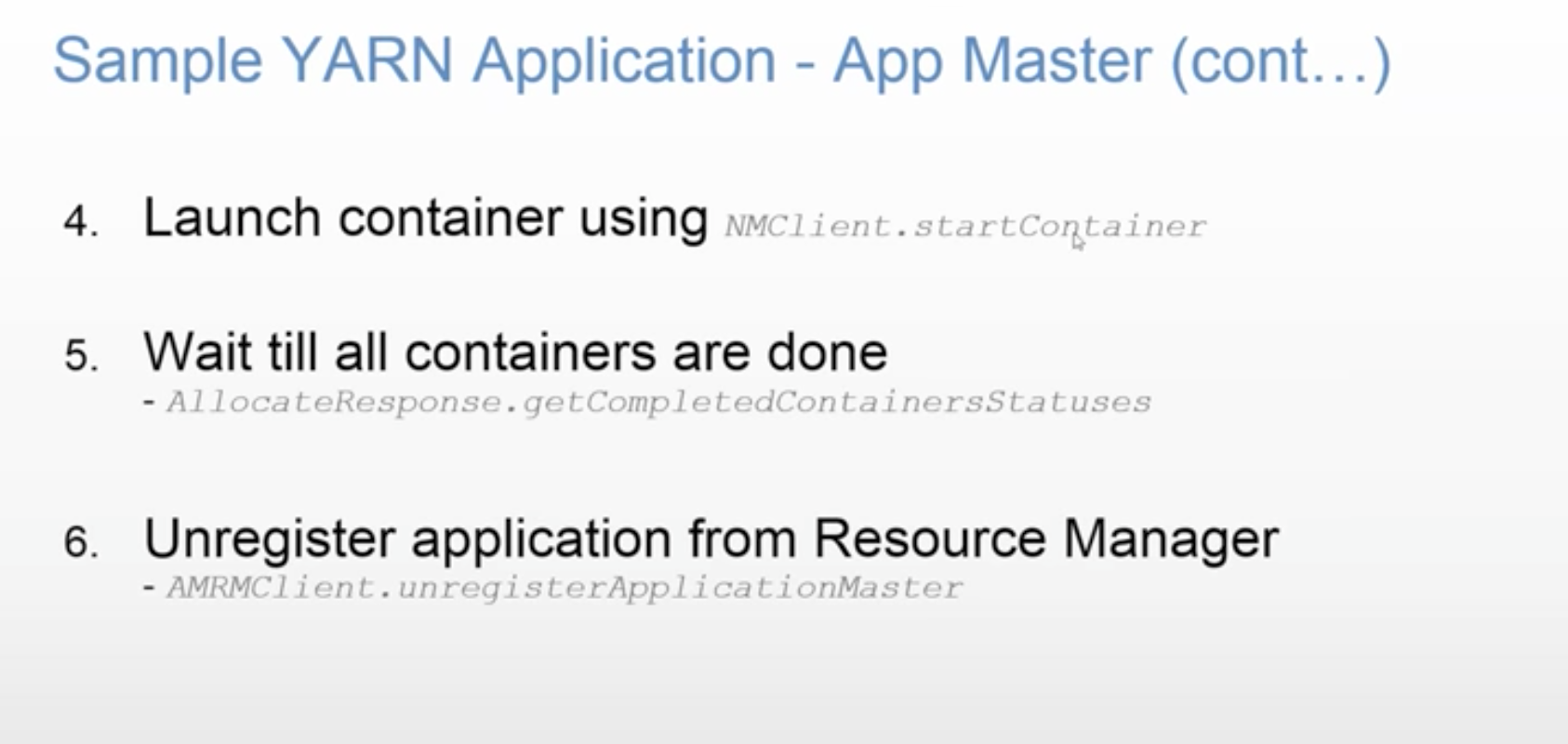
Jobs can be run for months long. Hadoop takes care of failure of nodes.

How Application Master is started?

RM can launch it on any available Node based on its availability.







You might have a need where two containers needs to share data. You can configure locality of the containers to be on the same node, so that they don’t use much network. Ideally, you should write independent applications.



