Spark and YARN: Better Together

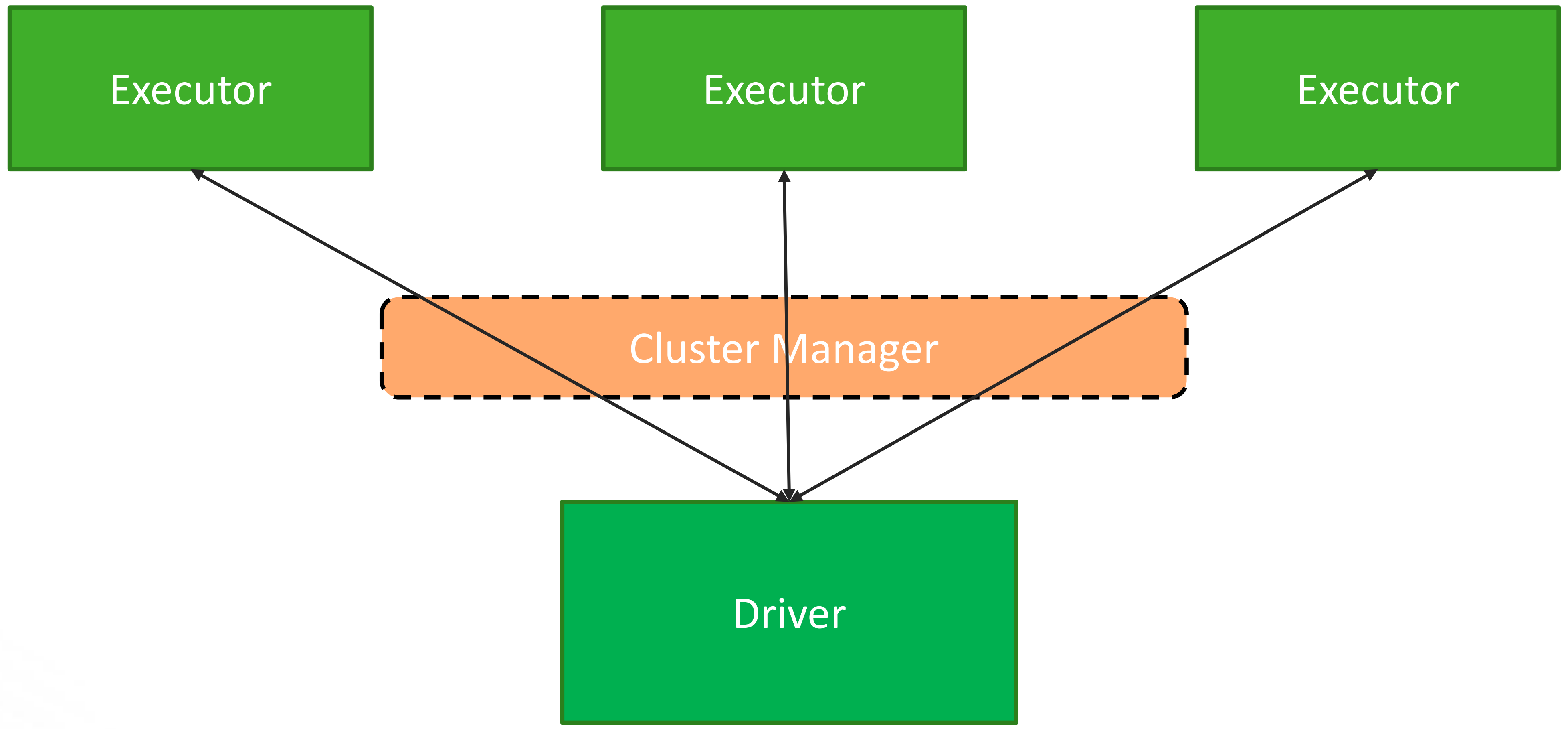
Saisai Shao
sshao@hortonworks.com

May 15, 2016

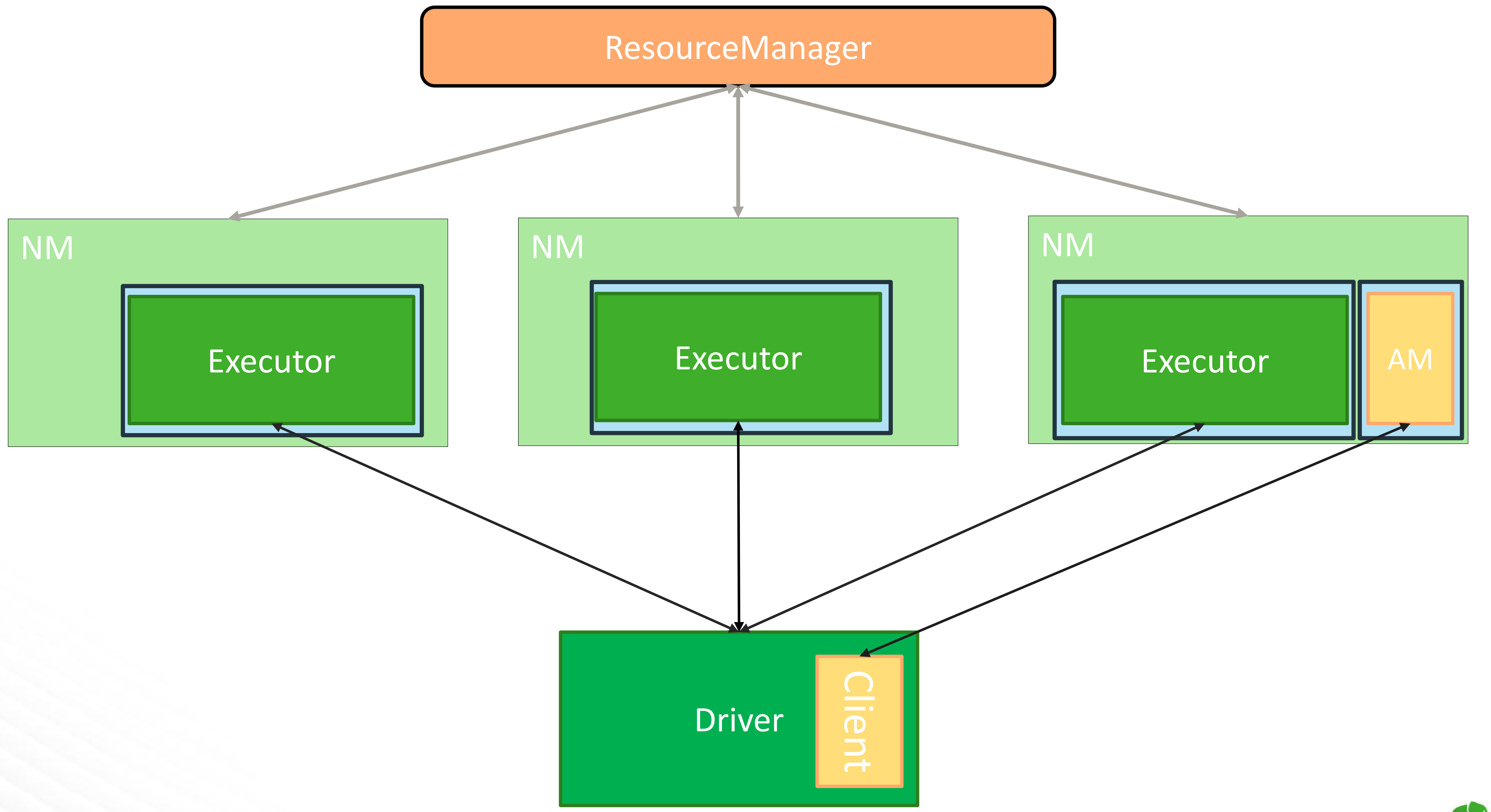
HORTONWORKS
POWERING THE FUTURE OF DATA™

Spark on YARN Recap

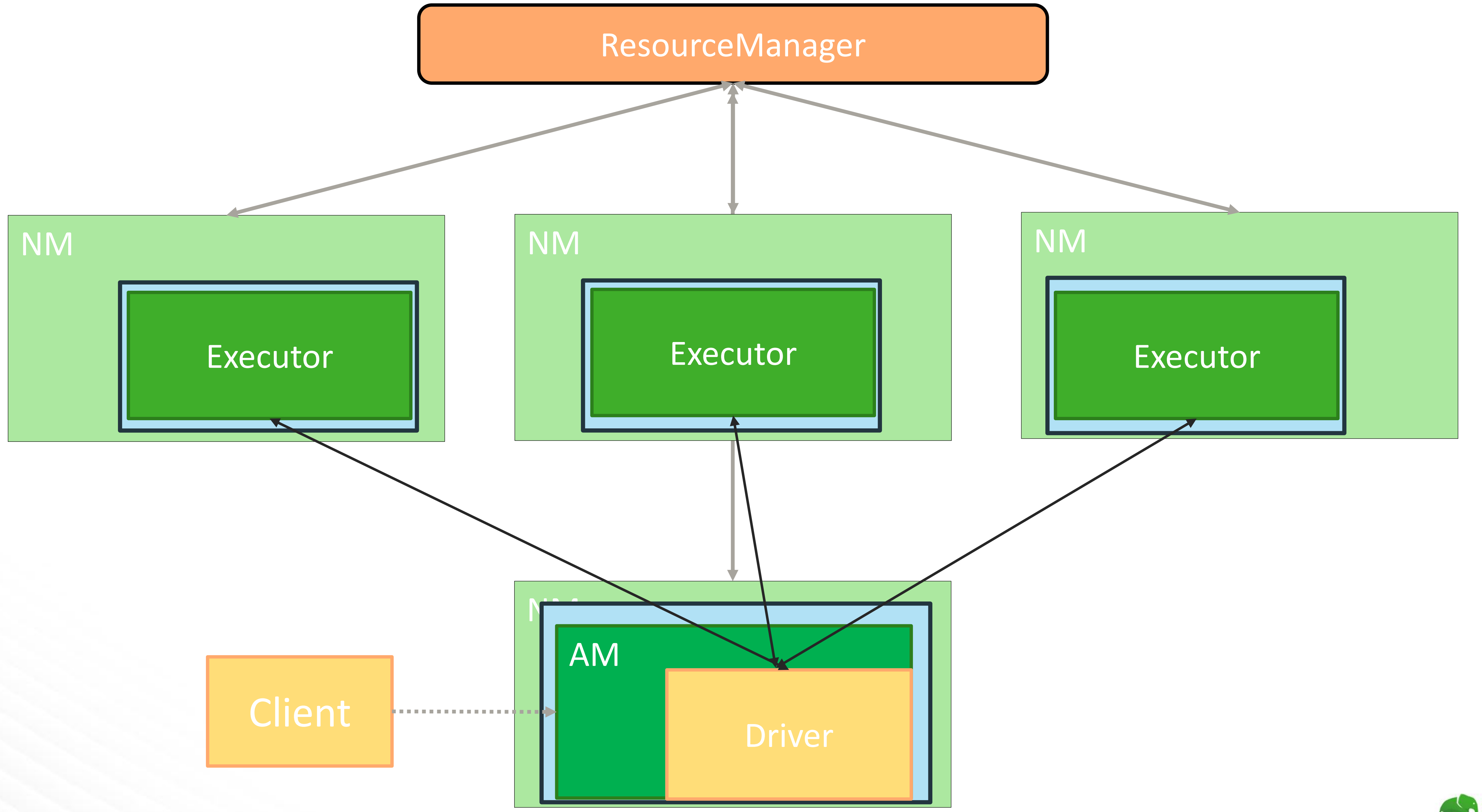
Overview of Spark Cluster



Spark Running On YARN (Client Mode)



Spark Running On YARN (Cluster Mode)



Difference Compared to Other Cluster Manager

- Application has to be submitted into a queue
- Jars/files/archives are distributed through distributed cache
- Additional ApplicationMaster
- …

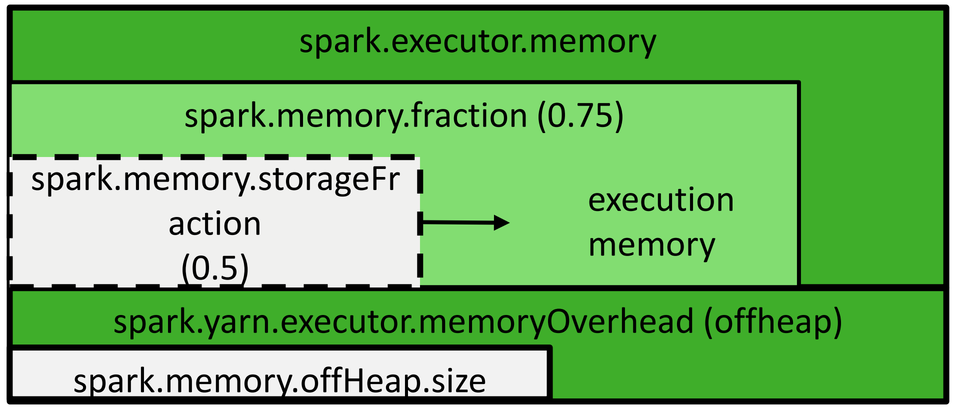
Better Run Spark On YARN

What Do We Concern About ?

- Better use the resources
- Better run on cluster
- Easy to debug

Calculate Container Size

- What is the size of a Container ?
  - Memory
  - CPU#


spark.executor.memory
spark.memory.fraction (0.75)
spark.memory.storageFraction (0.5)
execution memory
spark.yarn.executor.memoryOverhead (offheap)
spark.memory.offHeap.size

container memory = spark executor memory + overhead memory

yarn.scheduler.minimum-allocation-mb <= container memory
<= *yarn.nodemanager.resource.memory-mb*

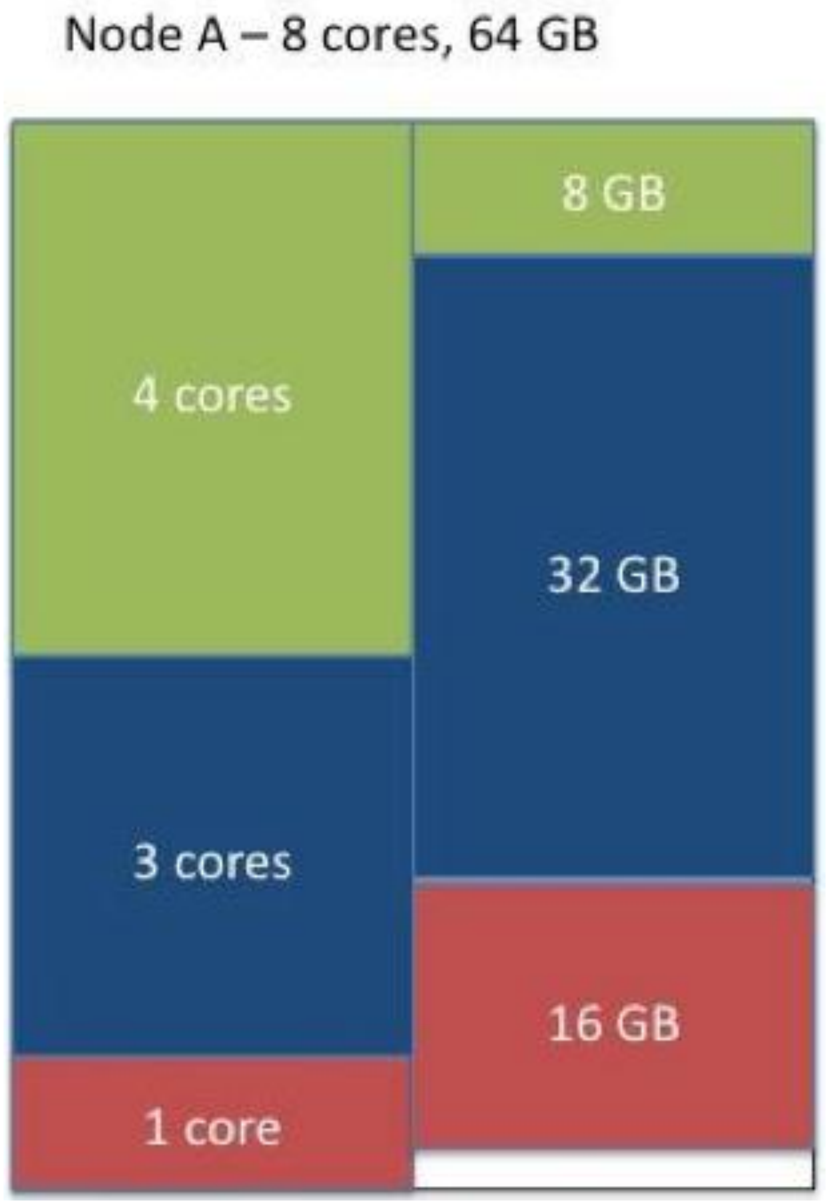
container memory will be round to *yarn.scheduler.increment-allocation-mb*

# Depend on CPU scheduling is enabled or not

Calculate Container Size (Cont'd)

- Enable CPU Scheduling
  - Capacity Scheduler with *DefaultResourceCalculator* (default)
    - Only takes memory into account
    - CPU requirements are ignored when carrying out allocations
    - The setting of "--executor-cores" is controlled by Spark itself
  - Capacity Scheduler with *DominantResourceCalculator*
    - CPU will also take into account when calculating
    - Container vcores = executor cores

container cores <= *nodemanger.resource.cpu-vcores*



Isolate Container Resource

Containers should only be allowed to use resource they get allocated, they should not be affected by other containers on the node

- How do we ensure containers don’t exceed their *vcore* allocation?
- What’s stopping an errant container from spawning bunch of threads and consume all the CPU on the node?

CGroups

With the setting of *LinuxContainerExecutor* and others, YARN could enable CGroups to constrain the CPU usage (https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html).

Label Based Scheduling

How to specify applications to run on specific nodes?

- Label based scheduling is what you want.

- To use it:
  - Enable node label and label scheduling in YARN side (Hadoop 2.6+)
  - Configure node label expression in Spark conf:
    - spark.yarn.am.nodeLabelExpression
    - spark.yarn.executor.nodeLabelExpression

Dynamic Resource Allocation

How to use the resource more effectively and more resiliently?

- Spark supports dynamically requesting or releasing executors according to the current load of jobs.
- This is especially useful for long-running applications like Spark shell, Thrift Server, Zeppelin.

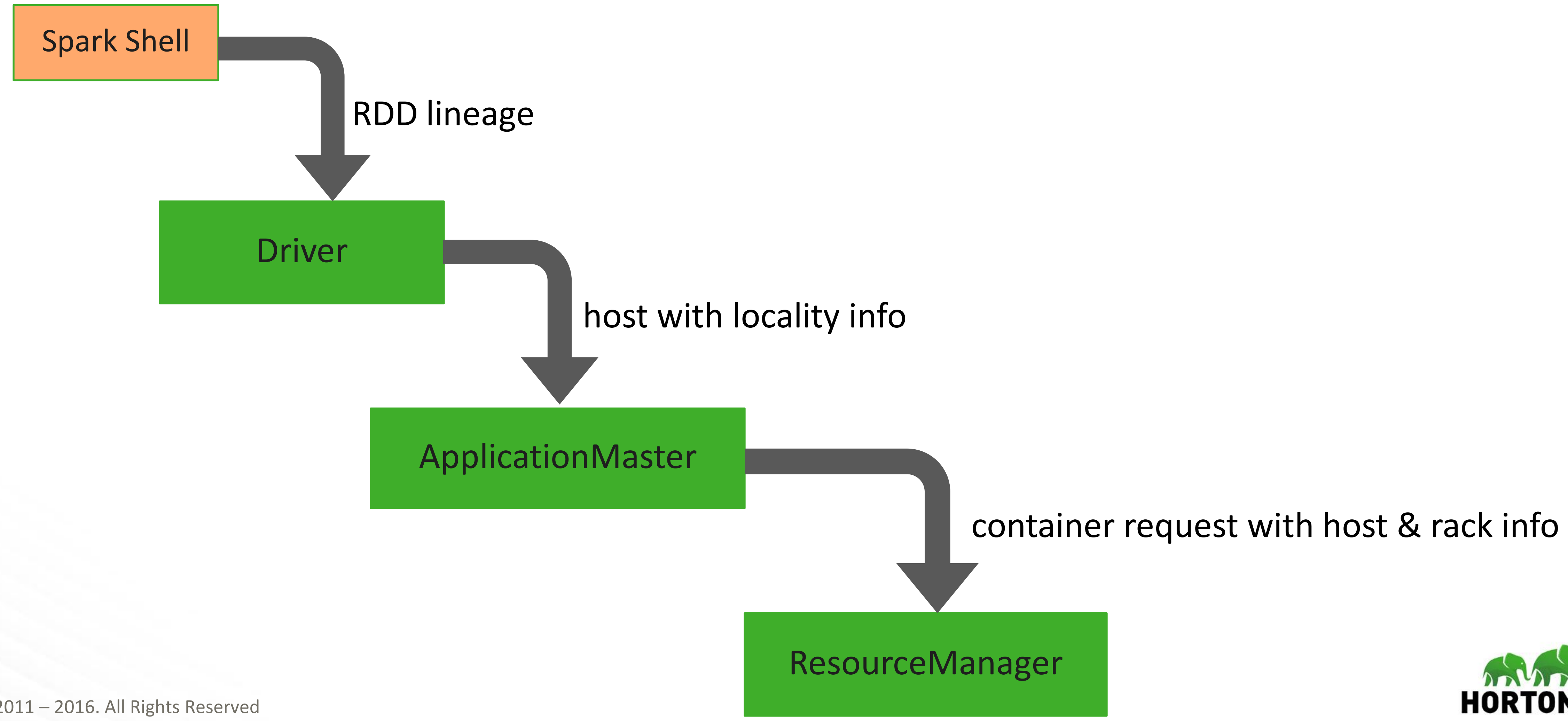
- To Enable Dynamic Resource Allocation

spark.streaming.dynamicAllocation.enabled	true
spark.shuffle.service.enabled	true

```
<property>
    <name>yarn.nodemanager.aux-services.spark_shuffle.class</name>
    <value>org.apache.spark.network.yarn.YarnShuffleService</value>
</property>
```

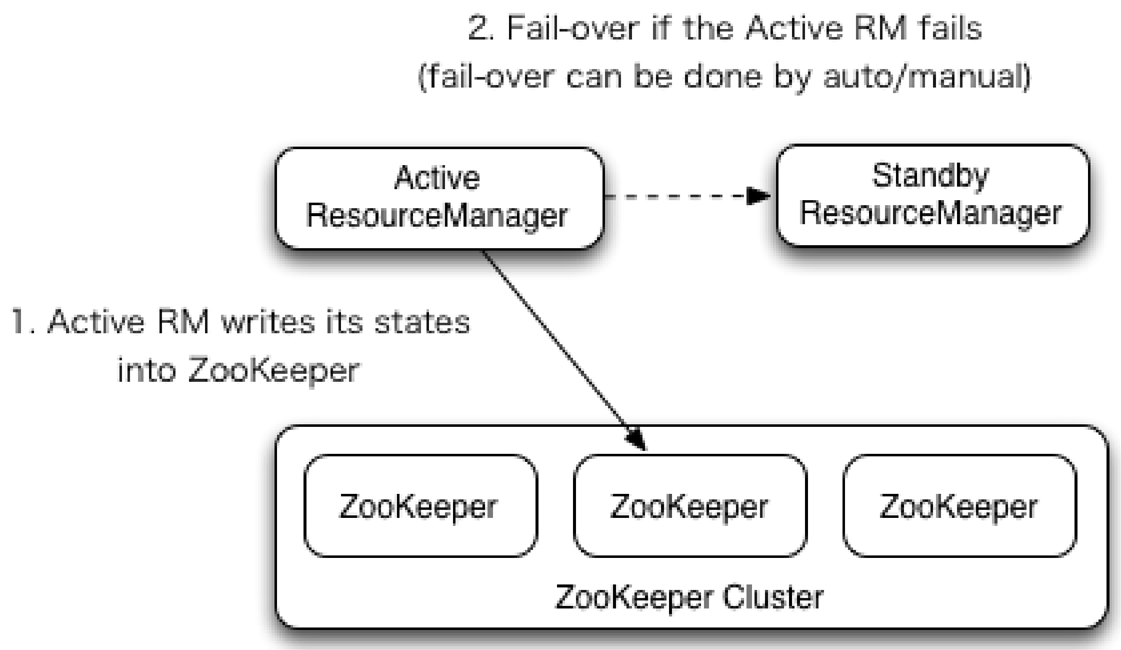
Dynamic Resource Allocation (Cont'd)

- For Spark On YARN, container allocation is based on locality preference
- Best effort algorithm to calculate node-locality ratio and request allocation



Resilient to Service Restart/Failure

- Resilient to ResourceManager restart/failure
  - RM is a single point of failure
  - Configuring *yarn.resourcemanager.ha.enabled* to enable RM HA
  - Enable *yarn.resourcemanager.recovery.enabled* to be resilient to RM change or restart
    - Non work preserving RM restart (Hadoop 2.4)
    - Work preserving RM restart (Hadoop 2.6)
- Resilient to NodeManager restart
  - Enable *yarn.nodemanager.recovery.enabled* to be resilient to NM restart (Hadoop 2.6)


2. Fail-over if the Active RM fails
(fail-over can be done by auto/manual)
Active ResourceManager
Standby ResourceManager
1. Active RM writes its states into ZooKeeper
ZooKeeper
ZooKeeper
ZooKeeper
ZooKeeper Cluster

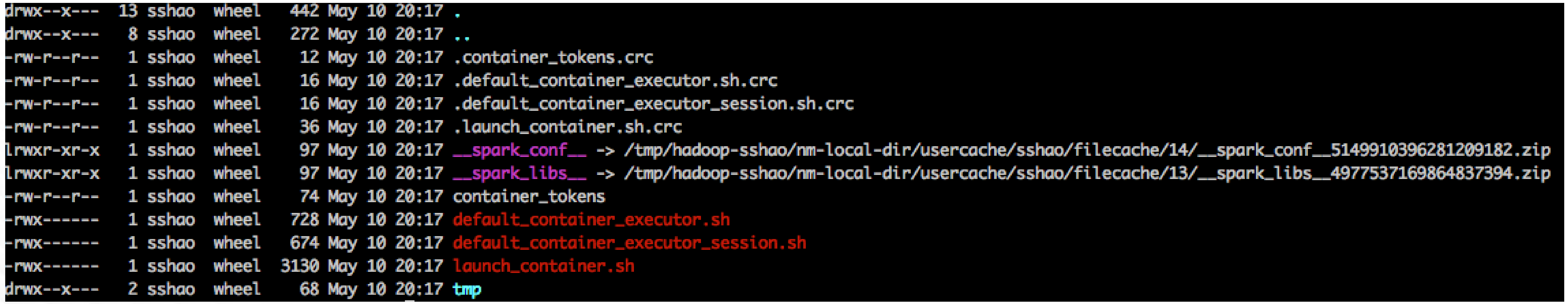
Access Kerberized Environment

- Spark on YARN supports accessing Kerberized Hadoop environment by Kerberos.
- It will automatically get tokens from NN if the Hadoop environment is security enabled.
- To retrieve the delegation tokens for non-HDFS services when security is enabled, configure *spark.yarn.security.tokens.{hive|hbase}.enabled*.
- You could also specify *--principal* and *--keytab* to let Spark on YARN to do kinit and token renewal automatically, this is useful for long running service like Spark Streaming.

Fast Debug YARN Application

- What we usually meet when running Spark on YARN?
  - Classpath problem
  - Java parameters does not work
  - Configuration doesn‘t take affect.

${yarn.nodemanager.local-dirs}/usercache/${user}/appcache/application_${appid}/container_${contid}



```
drwx--x---  13 sshao  wheel   442 May 10 20:17 .
drwx--x---   8 sshao  wheel   272 May 10 20:17 ..
-rw-r--r--   1 sshao  wheel    12 May 10 20:17 .container_tokens.crc
-rw-r--r--   1 sshao  wheel    16 May 10 20:17 .default_container_executor.sh.crc
-rw-r--r--   1 sshao  wheel    16 May 10 20:17 .default_container_executor_session.sh.crc
-rw-r--r--   1 sshao  wheel    36 May 10 20:17 .launch_container.sh.crc
lrwxr-xr-x   1 sshao  wheel    97 May 10 20:17 __spark_conf__ -> /tmp/hadoop-sshao/nm-local-dir/usercache/sshao/filecache/14/__spark_conf__5149910396281209182.zip
lrwxr-xr-x   1 sshao  wheel    97 May 10 20:17 __spark_libs__ -> /tmp/hadoop-sshao/nm-local-dir/usercache/sshao/filecache/13/__spark_libs__4977537169864837394.zip
-rw-r--r--   1 sshao  wheel    74 May 10 20:17 container_tokens
-rwx------   1 sshao  wheel   728 May 10 20:17 default_container_executor.sh
-rwx------   1 sshao  wheel   674 May 10 20:17 default_container_executor_session.sh
-rwx------   1 sshao  wheel  3130 May 10 20:17 launch_container.sh
drwx--x---   2 sshao  wheel    68 May 10 20:17 tmp
```

Future works of Spark On YARN

More Advanced Dynamic Resource Allocation

- The problem of current algorithm:
  - Current algorithm is based on the load of tasks, the more tasks submitted the more executor will be requested. This will introduce resource starvation for other applications.
  - Current algorithm is an universal algorithm doesn't consider the specific features of cluster manager.
- To improve current dynamic resource allocation:
  - Make current algorithm pluggable, to be adapted to customized algorithms.
  - Incorporate more YARN specific features to improve the current algorithm
    - Container resizing
    - Cooperative preemption
    - …

HORTONWORKS®

Integrate YARN Application Timeline Server

- Timeline Server - Storage and retrieval of applications' current as well as historic information in a generic fashion is solved in YARN through the Timeline Server. This serves two responsibilities:
  - Generic information about completed applications
  - Per-framework information of running and completed applications

- We're working on integrating Spark's history server with YARN application timeline server.
- Technical preview version is already released with HDP.

Better Long Running Support

- Spark applications can be divided into two categories: batch and long running.
- For long running applications, currently there's are several problems when running on YARN:
  - Fault tolerance of long running services.
  - Security support for long running services.
  - Log handling of long running services.
- What is done and missing in current Spark:
  - Window based attempt counting for AM is already supported (*spark.yarn.am.attempFailuresValidityInterval*) – SPARK-10739.
  - Window based executor failure counting for executor (SPARK-6735).
  - Token renewal is already supported for long running Spark applications, but has some small bugs.
  - Log handling for long running Spark application is not supported now.

Auxiliary Services Isolation

How to support different versions of Spark external shuffle services?

How to isolate the classpath of different shuffle services?

This is a problem when we want to support different major releases of Spark to coexist

YARN-4577 to solve this by isolating the classpath for different auxiliary services

Thank You

HORTONWORKS®
POWERING THE FUTURE OF DATA™