Apache Spark Resource Management and YARN App Models

May 30, 2014 (http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/) | By Sandy Ryza (http://blog.cloudera.com/blog/author/sandy-ryza/) | 13 Comments (http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/#comments)
Categories: Spark (http://blog.cloudera.com/blog/category/spark/) YARN (http://blog.cloudera.com/blog/category/yarn/)

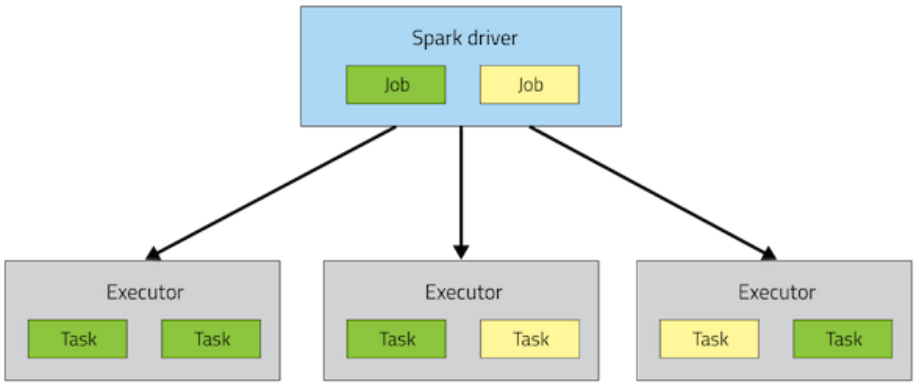
A concise look at the differences between how Spark and MapReduce manage cluster resources under YARN

The most popular Apache YARN application after MapReduce itself is Apache Spark. At Cloudera, we have worked hard to stabilize Spark-on-YARN (SPARK-1101 (https://issues.apache.org/jira/browse/SPARK-1101)), and CDH 5.0.0 added support for Spark on YARN clusters.

In this post, you'll learn about the differences between the Spark and MapReduce architectures, why you should care, and how they run on the YARN cluster ResourceManager.

Applications

In MapReduce, the highest-level unit of computation is a job. The system loads the data, applies a map function, shuffles it, applies a reduce function, and writes it back out to persistent storage. Spark has a similar job concept (although a job can consist of more stages than just a single map and reduce), but it also has a higher-level construct called an "application," which can run multiple jobs, in sequence or in parallel.



Spark application architecture

For those familiar with the Spark API, an application corresponds to an instance of the `SparkContext` class. An application can be used for a single batch job, an interactive session with multiple jobs spaced apart, or a long-lived server continually satisfying requests. Unlike MapReduce, an application will have processes, called *Executors*, running on the cluster on its behalf even when it's not running any jobs. This approach enables data storage in memory for quick access, as well as lightning-fast task startup time.

Executors

MapReduce runs each task in its own process. When a task completes, the process goes away. In Spark, many tasks can run concurrently in a single process, and this process sticks

Most Popular (http://blog.cloudera.com/most-popular/) Most Popular in /spark/ (http://blog.cloudera.com/most-popular-in-spark/)

Tags

analysis (http://blog.cloudera.com/blog/tag/analysis/) analytics (http://blog.cloudera.com/blog/tag/analytics/) apache (http://blog.cloudera.com/blog/tag/apache/) apache hadoop (http://blog.cloudera.com/blog/tag/apache-hadoop/) Apache HBase (http://blog.cloudera.com/blog/tag/apache-hbase/) apache hive (http://blog.cloudera.com/blog/tag/apache-hive/) beta (http://blog.cloudera.com/blog/tag/beta/) Big Data (http://blog.cloudera.com/blog/tag/big-data/) CDH (http://blog.cloudera.com/blog/tag/cdh/) cloudera (http://blog.cloudera.com/blog/tag/cloudera/) Cloudera Manager (http://blog.cloudera.com/blog/tag/cloudera-manager/) Community

around for the lifetime of the Spark application, even when no jobs are running.

The advantage of this model, as mentioned above, is speed: Tasks can start up very quickly and process in-memory data. The disadvantage is coarser-grained resource management. As the number of executors for an app is fixed and each executor has a fixed allotment of resources, an app takes up the same amount of resources for the full duration that it's running. (When YARN supports container resizing (https://issues.apache.org/jira/browse/YARN-1197), we plan to take advantage of it in Spark to acquire and give back resources dynamically.)

Active Driver

To manage the job flow and schedule tasks Spark relies on an active driver process. Typically, this driver process is the same as the client process used to initiate the job, although in YARN mode (covered later), the driver can run on the cluster. In contrast, in MapReduce, the client process can go away and the job can continue running. In Hadoop 1.x, the JobTracker was responsible for task scheduling, and in Hadoop 2.x, the MapReduce application master took over this responsibility.

Pluggable Resource Management

Spark supports pluggable cluster management. The cluster manager is responsible for starting executor processes. Spark application writers do not need to worry about what cluster manager against which Spark is running.

Spark supports YARN, Mesos, and its own "standalone" cluster manager. All three of these frameworks have two components. A central master service (the YARN ResourceManager, Mesos master, or Spark standalone master) decides which applications get to run executor processes, as well as where and when they get to run. A slave service running on every node (the YARN NodeManager, Mesos slave, or Spark standalone slave) actually starts the executor processes. It may also monitor their liveliness and resource consumption.

Why Run on YARN?

Using YARN as Spark's cluster manager confers a few benefits over Spark standalone and Mesos:

(http://blog.cloudera.com/blog/tag/community/) configuration (http://blog.cloudera.com/blog/tag/configuration/) data (http://blog.cloudera.com/blog/tag/data/) developer (http://blog.cloudera.com/blog/tag/developer/) developers (http://blog.cloudera.com/blog/tag/developers/) development (http://blog.cloudera.com/blog/tag/development/) events (http://blog.cloudera.com/blog/tag/events-2/) Flume (http://blog.cloudera.com/blog/tag/flume/) Hadoop (http://blog.cloudera.com/blog/tag/hadoop/) hadoop world (http://blog.cloudera.com/blog/tag/hadoop_world/) HBase (http://blog.cloudera.com/blog/tag/hbase/) HDFS (http://blog.cloudera.com/blog/tag/hdfs/) Hive (http://blog.cloudera.com/blog/tag/hive/) Hue (http://blog.cloudera.com/blog/tag/hue/) impala (http://blog.cloudera.com/blog/tag/impala-2/) installation (http://blog.cloudera.com/blog/tag/installation/) java (http://blog.cloudera.com/

- YARN allows you to dynamically share and centrally configure the same pool of cluster resources between all frameworks that run on YARN. You can throw your entire cluster at a MapReduce job, then use some of it on an Impala query and the rest on Spark application, without any changes in configuration.
- You can take advantage of all the features of YARN schedulers (http://hadoop.apache.org/docs/r2.4.0/hadoop-yarn/hadoop-yarn-site/FairScheduler.html) for categorizing, isolating, and prioritizing workloads.
- Spark standalone mode requires each application to run an executor on every node in the cluster, whereas with YARN, you choose the number of executors to use.
- Finally, YARN is the only cluster manager for Spark that supports security. With YARN, Spark can run against Kerberized Hadoop clusters and uses secure authentication between its processes.

Running on YARN

When running Spark on YARN, each Spark executor runs as a YARN container. Where MapReduce schedules a container and fires up a JVM for each task, Spark hosts multiple tasks within the same container. This approach enables several orders of magnitude faster task startup time.

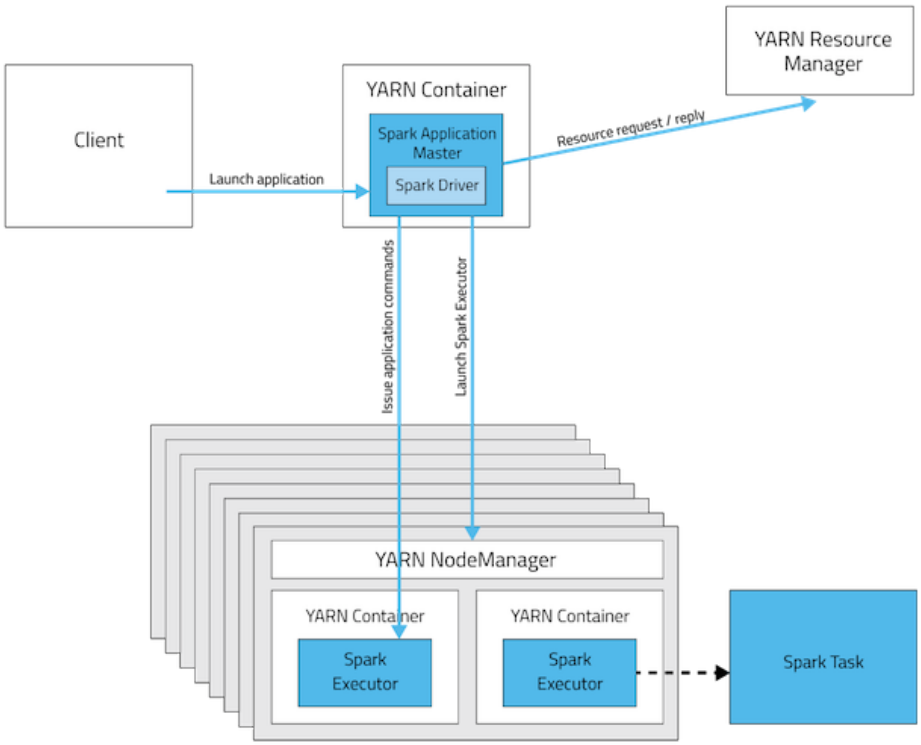
Spark supports two modes for running on YARN, "yarn-cluster" mode and "yarn-client" mode. Broadly, yarn-cluster mode makes sense for production jobs, while yarn-client mode makes sense for interactive and debugging uses where you want to see your application's output immediately.

Understanding the difference requires an understanding of YARN's *Application Master* concept. In YARN, each application instance has an Application Master process, which is the first container started for that application. The application is responsible for requesting resources from the ResourceManager, and, when allocated them, telling NodeManagers to start containers on its behalf. Application Masters obviate the need for an active client — the process

blog/tag/java/) log (http://blog.cloudera.com/blog/tag/log/) logs (http://blog.cloudera.com/blog/tag/logs/) MapReduce (http://blog.cloudera.com/blog/tag/mapreduce/) monitoring (http://blog.cloudera.com/blog/tag/monitoring/) open source (http://blog.cloudera.com/blog/tag/open-source/) Pig (http://blog.cloudera.com/blog/tag/pig/) platform (http://blog.cloudera.com/blog/tag/platform/) python (http://blog.cloudera.com/blog/tag/python/) questions (http://blog.cloudera.com/blog/tag/questions/) release (http://blog.cloudera.com/blog/tag/release/) REST (http://blog.cloudera.com/blog/tag/rest/) Search (http://blog.cloudera.com/blog/tag/search/) security (http://blog.cloudera.com/blog/tag/security/) sql (http://blog.cloudera.com/blog/tag/sql/) Support (http://blog.cloudera.com/blog/tag/support-2/) Testing (http://blog.cloudera.com/blog/tag/testing/) use

starting the application can go away and coordination continues from a process managed by YARN running on the cluster.

In yarn-cluster mode, the driver runs in the Application Master. This means that the same process is responsible for both driving the application and requesting resources from YARN, and this process runs inside a YARN container. The client that starts the app doesn't need to stick around for its entire lifetime.



yarn-cluster mode

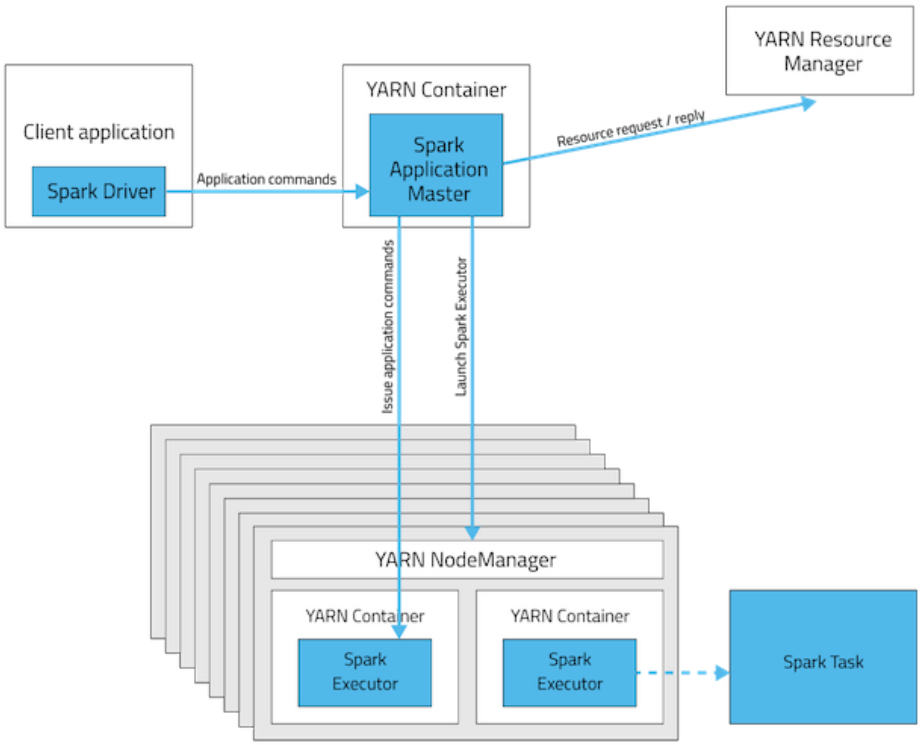
The yarn-cluster mode, however, is not well suited to using Spark interactively. Spark applications that require user input, like spark-shell and PySpark, need the Spark driver to run inside the client process that initiates the Spark application. In yarn-client mode, the Application Master is merely present to request executor containers from YARN. The client communicates with those containers to schedule work after they start:

cases (http://blog.cloudera.com/blog/tag/use-cases/)

Archives

Archives

Select Month



yarn-client mode

This table offers a concise list of differences between these modes:

	YARN Cluster	YARN Client	Spark Standalone
Driver runs in:	Application Master	Client	Client
Who requests resources?	Application Master	Application Master	Client
Who starts executor processes?	YARN NodeManager	YARN NodeManager	Spark Slave
Persistent services	YARN ResourceManager and NodeManagers	YARN ResourceManager and NodeManagers	Spark Master and Workers
Supports Spark Shell?	No	Yes	Yes

Key Concepts in Summary

- **Application:** This may be a single job, a sequence of jobs, a long-running service issuing new commands as needed or an interactive exploration session.
- **Spark Driver:** The Spark driver is the process running the spark context (which represents the application session). This driver is responsible for converting the application to a directed graph of individual steps to execute on the cluster. There is one driver per application.
- **Spark Application Master:** The Spark Application Master is responsible for negotiating resource requests made by the driver with YARN and finding a suitable set of

hosts/containers in which to run the Spark applications. There is one Application Master per application.

- **Spark Executor:** A single JVM instance on a node that serves a single Spark application. An executor runs multiple tasks over its lifetime, and multiple tasks concurrently. A node may have several Spark executors and there are many nodes running Spark Executors for each client application.
- **Spark Task:** A Spark Task represents a unit of work on a partition of a distributed dataset.

Further reading:

- Spark documentation (https://spark.apache.org/docs/latest/cluster-overview.html)
- Cloudera documentation for running Spark applications on YARN (http://blog.cloudera.com//content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/cdh5ig_running_spark_apps.html)

Sandy Ryza is a data scientist at Cloudera, and an Apache Hadoop committer.

Spark Summit 2014 is coming (June 30 – July 2)! Register here (https://sparksummit2014.eventbrite.com/?discount=Cloudera2014) to get 20% off the regular conference price.

apache (http://blog.cloudera.com/blog/tag/apache/) apache hadoop (http://blog.cloudera.com/blog/tag/apache-hadoop/) CDH (http://blog.cloudera.com/blog/tag/cdh/) cloudera (http://blog.cloudera.com/blog/tag/cloudera/) configuration (http://blog.cloudera.com/blog/tag/configuration/) data (http://blog.cloudera.com/blog/tag/data/) debugging (http://blog.cloudera.com/blog/tag/debugging/) Hadoop (http://blog.cloudera.com/blog/tag/hadoop/) impala (http://blog.cloudera.com/blog/tag/impala-2/) MapReduce

(http://blog.cloudera.com/blog/tag/mapreduce/) resource management (http://blog.cloudera.com/blog/tag/resource-management/) REST (http://blog.cloudera.com/blog/tag/rest/) scheduling (http://blog.cloudera.com/blog/tag/scheduling/) security (http://blog.cloudera.com/blog/tag/security/) Support (http://blog.cloudera.com/blog/tag/support-2/)

13 responses on "Apache Spark Resource Management and YARN App Models"

Andrey

May 31, 2014 at 3:32 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-45488)

Hi, thanks for great article. Just a little note: you've misused the pictures for cluster and client modes. :)

Justin Kestelyn (@kestelyn)

May 31, 2014 at 8:17 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-45491)

Thanks for pointing that out, fixed.

Chester (http://www.alpinenow.com)

June 29, 2014 at 1:34 pm

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-45752)

Sandy,
Excellent article. Very well explained.
Chester

Nit

July 2, 2014 at 6:26 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-45768)

Hey Sandy – Thanks for great talk in spark summit. I think slides are not out yet?… can you please post the command that you mentioned for preferredNodeLocationData.

Mukul Jain

August 4, 2014 at 12:54 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-45933)

Good article.

In the executor section it is worth mentioning/ highlighting that one other (important) disadvantage of Executor approach (beside the coarse-grained management) is that a misbehaving task can now impact not only itself but all tasks running under that executor. A task crashing the process thus not have impact on other tasks too but not other applications since one executor is still running tasks for same application.

Thanks for the article and explaining it nicely.

Gerry

September 11, 2014 at 11:19 pm

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-46437)

Great diagram to illustrate the difference between all these different modes. I'm trying to figure out how to run spark in yarn client mode where spark driver runs on my local machine and the yarn cluster is behind a firewall. I can ssh to a host inside the firewall, so in theory I can use ssh tunneling, but I haven't figured out the details yet. Alternatively, we could open up a set up ports for yarn on the firewall. Could you shed some light on these solutions?

Milan

March 26, 2015 at 4:49 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-53792)

Hi

I want to run a spark job (which is in my local machine), to be submitted to a YARN cluster in a different machine.

Any help is appreciated.

Vijay Ramisetty

August 5, 2015 at 11:48 pm

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-65943)

Nice article. well explained :)

Tudor-Lucian Lapusan

February 12, 2016 at 12:57 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-74517)

Nice explanations,
Now I have a better understanding of MapReduce vs Spark and how Spark is using YARN for resource requests.

Nice description of how Spark drives run when the application is ran in client or cluster mode.

Agrta Rawat

March 3, 2016 at 10:14 pm

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-74787)

Very good article, gives a complete understanding of Sprak proccess. :)
Thanks for such an article!

Rick Bross

June 23, 2016 at 6:08 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-76355)

Just curious about something that I can't seem to find a definitive statement on; we run an on-premises Spark cluster in Standalone mode. In this mode we run a Spark worker process on the master node, so all nodes run executors.
We are deploying in EMR, which automatically sets up Yarn. We only intend to run Spark on this cluster. Is there any way to have the Yarn "master" also run the NodeManager and spawn executors? Otherwise the resource usage for that node is so low that the node is almost wasted.

Justin Kestelyn

June 24, 2016 at 3:40 pm

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-76375)

Rick, I think that would be a question for Amazon; not sure what YARN config options are provided for EMR by AWS (if I understand your question correctly).

VipinK

September 21, 2016 at 3:46 am

(http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/comment-page-1/#comment-77223)

In a cloudera YARN cluster, is it possible to run Spark on few of the nodes. e.g. Out of 8 nodes cluster, I want to run Spark on 3 nodes. Rest of the nodes I want to keep for HBase. Is it possible on a YARN cluster?

‹ New SQL Choices in the Apache Hadoop Ecosystem: Why Impala Continues to Lead (http://blog.cloudera.com/blog/2014/05/new-sql-choices-in-the-apache-hadoop-ecosystem-why-impala-continues-to-lead/)

Apache Spark 1.0 is Released (http://blog.cloudera.com/blog/2014/05/apache-spark-1-0-is-released/) ›

Career (http://jobs.jobvite.com/cloudera/)
Contact (https://www.cloudera.com/contact-us.html)

United States: +1 888 789 1488
Outside the US: +1 650 362 0488