Yarns about YARN: Migrating to MapReduce v2
Kathleen Ting, kate@cloudera.com
Miklos Christine, mwc@cloudera.com
Strata Hadoop San Jose, 19 February 2015
Whoami

Kathleen Ting
• Joined Cloudera in 2011
• Former customer operations engineer
• Technical account manager
• *Apache Sqoop Cookbook* co-author

Miklos Christine
• Joined Cloudera 2013
• Former customer operations engineer
• Systems engineer
• Apache Spark expert
Cloudera and Apache Hadoop

• Apache Hadoop is an open source software framework for distributed storage and distributed processing of Big Data on clusters of commodity hardware.

• Cloudera is revolutionizing enterprise data management by offering the first unified Platform for Big Data, an enterprise data hub built on Apache Hadoop.
  • Distributes CDH, a Hadoop distribution.
  • Teaches, consults, and supports customers building applications on the Hadoop stack.
  • The world-wide Cloudera Customer Operations Engineering team has closed tens of thousands of support incidents over six years.
Outline

• YARN motivation
• Upgrading MR1 to MR2
• YARN upgrade pitfalls
• YARN applications
YARN motivation
Yet Another Resource Negotiator
One platform, many workloads: batch, interactive, real-time
Map-reduce finally explained (по ссылке от @scr4t)
An Apache YARN timeline

- **January 2008:** Yahoo started work on YARN
- **June 2012:** CDH 4.0.0 included YARN
- **August 2012:** YARN promoted to Apache Hadoop sub-project
- **October 2013:** Hadoop 2.0 GA
- **April 2014:** YARN/MR2 is default in CDH 5
MapReduce v1

JobTracker
- Schedule Jobs
- Manage Jobs
- Store Jobs

TaskTracker
- Map Task
- Reduce Task

TaskTracker
- Map Task
- Reduce Task
MapReduce v2

Resource Manager
Schedule Jobs

NodeManager
Container
Map Task

NodeManager
Container
App Master
Manage Jobs

Container
Job History Server
Store Jobs

Container
Reduce Task
<table>
<thead>
<tr>
<th></th>
<th>MR1</th>
<th>MR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>JobTracker tracks all jobs, tasks</td>
<td>Split up tracking between ResourceManager, ApplicationMaster</td>
</tr>
<tr>
<td></td>
<td>Max out at 4k nodes, 40k tasks</td>
<td>Scale up to 10k nodes, 100k tasks</td>
</tr>
<tr>
<td>Availability</td>
<td>JT HA</td>
<td>RM HA &amp; for per-application basis</td>
</tr>
<tr>
<td>Utilization</td>
<td>Fixed size slots for map, reduce</td>
<td>Allocate only as many resources as needed, allows cluster utilization &gt; 70%</td>
</tr>
<tr>
<td>Multi-tenancy</td>
<td>N/A</td>
<td>Cluster resource management system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data locality &amp; lowered operational costs from sharing resources between frameworks</td>
</tr>
</tbody>
</table>
Upgrading MR1 to MR2
MR1 to MR2 functionality mapping

- Completely revamped architecture in MR2 on YARN
- While most translate, some configurations don’t
<table>
<thead>
<tr>
<th>MR2 on YARN</th>
<th>Applications on YARN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory &gt; heap to account for overhead:</td>
<td>Mem/CPU thresholds:</td>
</tr>
<tr>
<td>Memory per Container: mapreduce.[map</td>
<td>reduce].memory.mb (1.5 GB)</td>
</tr>
<tr>
<td>Map/Reduce Task Maximum Heap Size:</td>
<td>Container Memory Maximum: yarn.scheduler.maximum-allocation-memory</td>
</tr>
<tr>
<td>mapreduce.[map</td>
<td>reduce].java.opts.max.heap (1GB)</td>
</tr>
<tr>
<td>CPU per Container:</td>
<td>Container Virtual CPU Cores Maximum: yarn.scheduler.maximum-allocation-vcores</td>
</tr>
<tr>
<td>mapreduce.[map</td>
<td>reduce].cpu.vcores</td>
</tr>
</tbody>
</table>
# YARN compatibility

<table>
<thead>
<tr>
<th>Migration path</th>
<th>Binary support</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR1 (CDH4) to MR2 (CDH5)</td>
<td>✔</td>
</tr>
<tr>
<td>MR1 (CDH4) to MR1 (CDH5)</td>
<td>✔</td>
</tr>
<tr>
<td>MR2 (CDH4) to MR1/MR2 (CDH5)</td>
<td>✖</td>
</tr>
</tbody>
</table>

CDH has complete binary/source compatibility for almost all programs.

Virtually every job compiled against MR1 in CDH 4 will be able to run without any modifications on an MR2 cluster.

- Migrating to MR2 on YARN

- Getting MR2 Up to Speed
  - [http://blog.cloudera.com/blog/2014/02/getting-mapreduce-2-up-to-speed/](http://blog.cloudera.com/blog/2014/02/getting-mapreduce-2-up-to-speed/)

- Avoiding YARN Gotchas
YARN upgrade pitfalls
MapReduce v2

Resource Manager
- Schedule Jobs

NodeManager
- Container
  - Map Task
  - App Master
  - Manage Jobs

Job History Server
- Store Jobs

NodeManager
- Container
  - Reduce Task
MapReduce v2

Resource Manager
Schedule Jobs

NodeManager
Container
Map Task
Container
App Master
Manage Jobs

Job History Server
Store Jobs

HDFS

NodeManager
Container
Reduce Task
Logs
# General log related configuration properties

<table>
<thead>
<tr>
<th>Log configuration parameter</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>yarn.nodemanager.log-dirs</td>
<td>Determines where the container-logs are stored on the node when the containers are running. Default is <code>${yarn.log.dir}/userlogs</code>. For MapReduce applications, each container directory will contain the files stderr, stdin, and syslog generated by that container.</td>
</tr>
<tr>
<td>yarn.log-aggregation-enable</td>
<td>Whether to enable log aggregation or not. If disabled, NMs will keep the logs locally and not aggregate them.</td>
</tr>
</tbody>
</table>
YARN applications
Llama, Slider, Spark
YARN applications

• Llama (Low Latency Application MAtser)
  • Reserves memory in YARN for short-lived processes (e.g. Impala)
  • Registers one long-lived AM per YARN pool
  • Caches resources allocated by YARN for a short time, so that they can be quickly re-allocated to Impala queries
  • Long-term solution is to run Impala on YARN but currently recommend setting up admission control
YARN applications

• Apache Slider (incubating) née Hoya
  • Runs long-lived persistent services on YARN (e.g. HBase)
  • Not currently recommended as it doesn’t provide IO isolation
Spark on YARN
Spark Overview

- Application corresponds to an instance of the SparkContext class
- Executors are long lived processes
- Applications take up resource until the app completes
Why Spark on Yarn?

• Built in scheduler for resource management (Isolation, Prioritization)
• Sharing resources within a cluster (MapReduce, Spark)
• YARN is the only cluster manager for Spark that supports security (Kerberized Hadoop).
Configuring YARN for Spark

• Designed for interactive queries and iterative algorithms
  • In-memory caching, DAG engine, and APIs
• Set yarn.scheduler.maximum-allocation-mb as high as 64G on a machine with 192GB of memory
• Won’t run with small (< 1 GB) containers due to overhead

Reference:
http://blog.cloudera.com/blog/2014/05/apache-spark-resource-management-and-yarn-app-models/
# Deploying Spark Jobs

<table>
<thead>
<tr>
<th></th>
<th>YARN Cluster</th>
<th>YARN Client</th>
<th>Spark Standalone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driver runs in:</strong></td>
<td>Application Master</td>
<td>Client</td>
<td>Client</td>
</tr>
<tr>
<td><strong>Who requests resources?</strong></td>
<td>Application Master</td>
<td>Application Master</td>
<td>Client</td>
</tr>
<tr>
<td><strong>Who starts executor processes?</strong></td>
<td>YARN NodeManager</td>
<td>YARN NodeManager</td>
<td>Spark Slave</td>
</tr>
<tr>
<td><strong>Persistent services</strong></td>
<td>YARN ResourceManager and NodeManagers</td>
<td>YARN ResourceManager and NodeManagers</td>
<td>Spark Master and Workers</td>
</tr>
<tr>
<td><strong>Supports Spark Shell?</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Spark – YARN-CLIENT
Spark – YARN-CLUSTER
Spark Scheduling

- Fair scheduler for resource sharing
  - spark.yarn.queue
- Standalone cluster mode currently only supports a simple FIFO scheduler across applications

Dynamic Resource Pools

<table>
<thead>
<tr>
<th>Name</th>
<th>Weight</th>
<th>%</th>
<th>Virtual Cores Min / Max</th>
<th>Memory Min / Max</th>
<th>Max Running Apps</th>
<th>Scheduling Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>1</td>
<td>100.0%</td>
<td>- / -</td>
<td>- / -</td>
<td>-</td>
<td>DRF</td>
</tr>
<tr>
<td>default</td>
<td>1</td>
<td>12.5%</td>
<td>- / -</td>
<td>- / -</td>
<td>-</td>
<td>DRF</td>
</tr>
<tr>
<td>spark-prod</td>
<td>3</td>
<td>37.5%</td>
<td>1 / 5</td>
<td>1000MB / 10000MB</td>
<td>5</td>
<td>DRF</td>
</tr>
<tr>
<td>spark-test</td>
<td>1</td>
<td>12.5%</td>
<td>1 / 5</td>
<td>2000MB / 5000MB</td>
<td>3</td>
<td>DRF</td>
</tr>
<tr>
<td>mapred-prod</td>
<td>3</td>
<td>37.5%</td>
<td>5 / 50</td>
<td>5000MB / 50000MB</td>
<td>10</td>
<td>DRF</td>
</tr>
</tbody>
</table>
Spark Not Running On Yarn?

**Symptom:**
- Use spark-submit to run a python job, but only see the resources being used on one machine.
Spark Not Running On Yarn?

• Workaround:
  • Ensure that you have the options in the right position
  • Cause:
    • $ spark-submit pi.py --master yarn-client
  • Fix:
    • $ spark-submit --master yarn-client pi.py 1000
• Usage:
  spark-submit [options] <app jar | python file> [app options]
• Lot of improvements made to Spark 1.2 for spark-submit SPARK-1652
PySpark on Yarn Limitation

• **Symptom:**

   $ spark-submit --master yarn-cluster pi.py 1000

   Error: Cluster deploy mode is currently not supported for python applications.
   Run with --help for usage help or --verbose for debug output
PySpark on Yarn Limitation

**Workaround:**

```bash
$ spark-submit --master yarn-client pi.py 1000
...
Pi is roughly 3.132290
15/02/11 09:41:34 INFO SparkUI: Stopped Spark web UI at http://sparktest-1.ent.cloudera.com:4040
15/02/11 09:41:34 INFO DAGScheduler: Stopping DAGScheduler
```

**Future work:** SPARK-5162 / SPARK-5173
Lost Spark Executors

• **Symptom:**
  - Spark Driver WARN Messages
    14/12/08 17:11:08 WARN scheduler.TaskSetManager: Lost task 205.0 in stage 2.0 (TID 352, test-1.cloudera.com: ExecutorLostFailure (executor lost)

• NodeManager Logs
  Current usage: 26.2 GB of 26 GB physical memory used;
  27.1 GB of 54.6 GB virtual memory used. Killing container.
Lost Spark Executors

• **Workaround:**
  • Increase
    `spark.yarn.[executor|driver].memoryOverhead`
    • Test for your specific use case. 1GB to 4GB
Spark Improvements

• Spark 1.2 / CDH 5.3 : Prefer RDDs that are cached locally in HDFS

• Spark 1.2 : Dynamically release unused resources via spark.
  dynamicAllocation.enabled
  • Only support via YARN currently.

• Spark Streaming save incoming data to a WAL (write-ahead log) on HDFS,
  preventing any data loss on driver failure.
Conclusion
YARN performance

• Improved cluster utilization
  • Can run more jobs in smaller clusters
  • Run in uber mode for smaller jobs (reduces AM overhead)
• Dynamic resource sharing between frameworks
  • One framework can use the entire cluster
• Tom White’s *Hadoop: The Definitive Guide 4th Ed* (book signing @6:30pm)
  • Chapter 4 is on YARN
Hello, Cloudera Customers and Users!

These community forums are intended for developers and admins using Cloudera's Apache Hadoop–based platform to welcome your suggestions and feedback here.

Join this community to get a 40% discount for O'Reilly Media print books, and 50% for e-books and videos (bundles not included) — as well as

To participate in upstream open source projects, use their respective upstream mailing lists.

Ask a Question

Type your question here...

Continue

Community

News (2 Items)

Title | Posts
---|---
Community Guidelines & News | 5
Latest Post – This community is now mobile-friendly
Release Announcements | 40
Latest Post – Announcing: New Cloudera ODBC drivers for Impala a...
The fastest way to get started with Hadoop
Visit us at Booth #809

HIGHLIGHTS:

Apache Kafka is now fully supported with Cloudera

Learn why Cloudera is the leader for security in Hadoop
Questions?
@kate_ting
@miklos_c
Spark Tuning Parameters

- spark.shuffle.consolidateFiles=true
- spark.yarn.executor.memoryOverhead
- spark.yarn.driver.memoryOverhead
- spark.shuffle.manager=SORT
- spark.rdd.compress=true
- spark.serializer=org.apache.spark.serializer.KryoSerializer
YARN vs Mesos: Resource Manager’s role

<table>
<thead>
<tr>
<th>YARN</th>
<th>Mesos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asks for resources</td>
<td>Offers resources</td>
</tr>
<tr>
<td>Evolved into a resource manager</td>
<td>Evolved into managing Hadoop</td>
</tr>
<tr>
<td>Written in Java</td>
<td>Written in C++</td>
</tr>
<tr>
<td>Locality aware</td>
<td>More customizable</td>
</tr>
</tbody>
</table>