Breaking Spark
Top 5 Mistakes to avoid when using Apache Spark in Production

Neelesh Srinivas Salian
Outline

1. Introduction to Spark
2. Information gathering
3. Classification of the Issues
   1. Description
   2. Remedies
4. Q&A
Information Gathering

Source:
• Cloudera Customer Opened Cases
• Posts in Cloudera Community Forums
• Spark use cases for the following:

<table>
<thead>
<tr>
<th>Spark Core</th>
<th>Spark Streaming</th>
<th>Spark on YARN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Spark Applications and memory configuration problems</td>
<td>Problems in the Ingest pipelines</td>
<td>Usually Resource Management issues</td>
</tr>
<tr>
<td>Problems in interacting with Kafka</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Classification

1. Scaling of the Architecture
2. Memory Configurations
3. End user Code
4. Incompatible Dependencies
5. Administration/Operation related issues
1. Scaling of the Architecture

1. Increase in the cluster Size
2. Utilization of the Cluster
1. Scale of the Architecture: Larger Datasets

1. Understand the initial setup and current one

2. Look at logs / console to find the error

3. Determine parameters

4. Tune away
2. Memory Configurations

YARN

Remember:
Container Memory
• yarn.nodemanager.resource.memory-mb

Container Memory Maximum
• yarn.scheduler.maximum-allocation-mb

Container Memory Minimum
• yarn.scheduler.minimum-allocation-mb

Standalone

Java Heap Size of Worker in Bytes
• worker_max_heapsize

Java Heap Size of Master in Bytes
• master_max_heapsize
2. Memory Configurations: Insufficient Memory

Symptoms:

**Executor error:** ERROR 
util.SparkUncaughtExceptionHandler: Uncaught exception in thread Thread[Executor task launch worker-0,5,main] 
java.lang.OutOfMemoryError: Java heap space

**Driver error:** ERROR actor.ActorSystemImpl: Uncaught fatal error from thread [sparkDriver-akka.actor.default-dispatcher-60] shutting down ActorSystem [sparkDriver] 
java.lang.OutOfMemoryError: Java heap space

Cause:

The driver or executor is configured with a memory limit that is too low for the application being run.
2. Memory Configurations: Insufficient Memory

1. Understand Spark Configuration Order of Precedence
   1. Spark configuration file: Parameters are passed to SparkConf.
   2. Command-line arguments are passed to spark-submit, spark-shell, or pyspark.
   3. Cloudera Manager UI properties set in the spark-defaults.conf configuration file.

2. Verify the Driver and Executor Memory Used
   1. Environment tab in History Server
   2. Check driver and executor memory usage values

3. Increase the Heap
   Caveat: do not set the spark.driver.memory config using SparkConf, use --driver-memory
   - spark.driver.memory
   - spark.executor.memory
3. End User Code

1. Why chose Spark code over MapReduce?

2. Expensive Operations
   1. GroupByKey vs ReduceByKey

3. Syntax / Command Errors
4. Incompatible Dependencies

1. Incorrectly built jars

2. Version mismatch
4. Incompatible Dependencies: Classpath Issues

Symptoms

Error: Could not find or load main class
org.apache.spark.deploy.yarn.Application
Master

java.lang.ClassNotFoundException:
org.apache.hadoop.mapreduce.MRJobCon
fig

java.lang.NoClassDefFoundError: org/
apache/hadoop/fs/FSDataInputStream

ImportError: No module named
qt_returns_summary_wrapper
4. Incompatible Dependencies: Classpath Issues

What to Do?

1. Verify if an application is using classes that are not part of the framework (Hadoop/Spark)

2. Verify that an application is built using correct artifacts.
5. Administration/Operation Related issues

1. Management of Cluster
2. User-related issues
3. Security/Permissions
4. No Configurations setup
6. Bonus: People

Left Blank Intentionally
Summary

• Learn from the Mistakes

• Be prepared

• References:
  • Tuning Blogs Cloudera
  • Tuning YARN Cloudera
  • Spark Documentation
Credits

- Wilfred
- Paula
- Bjorn
- Mark
- Anand
- Sean
- Attila
- Dat
- Anthony
Thank you

Neelesh Srinivas Salian
LinkedIn
Email: neeleshssalian@gmail.com
Twitter: NeelS7