SCALABLE MONITORING WITH APACHE SPARK

Diane Feddema, Principal Software Engineer, CTO Office
Zak Hassan, Senior Software Engineer, CTO Office
YOUR SPEAKERS

DIANE FEDDEMA
PRINCIPAL SOFTWARE ENGINEER - AI/ML CENTER OF EXCELLENCE, CTO OFFICE
- Currently focused on developing and applying Data Science and Machine Learning techniques for performance analysis, automating these analyses and displaying data in novel ways.
- Previously worked as a performance engineer at the National Center for Atmospheric Research, NCAR, working on optimizations and tuning in parallel global climate models.

ZAK HASSAN
SENIOR SOFTWARE ENGINEER - AI/ML CENTER OF EXCELLENCE, CTO OFFICE
- Currently focused on developing analytics platform on OpenShift and leveraging Open Source ML Frameworks: Apache Spark, Tensorflow and more. Designing high performance and scalable ML platform that exposes metrics through cloud-native technology, Prometheus and Kubernetes.
OVERVIEW

OBSERVABILITY

● Motivation
● Integrating:
  ○ Apache Spark with radanalytics.io
  ○ Prometheus
  ○ Kubernetes
  ○ Grafana
● Spark Cluster JVM Instrumentation

PERFORMANCE TUNING

● Tuning Spark jobs
● Spark Memory Model
● Prometheus as a performance tool
● Comparing cached vs non-cached dataframes
● Demo
MOTIVATION

- Rapid experimentation with data science apps
- Identify bottlenecks
- Improve performance
- Resolve incidents more quickly
- Improving memory usage to tune spark jobs
Instrumented spark JVM to expose metrics in an OpenShift pod.
- Added ability to monitor spark with Prometheus
- Experimented with using Grafana with Prometheus to provide more insight
- Sharing our experiments and experience with using this to do performance analysis of spark jobs.
- Demo at the very end

June 1, 2017 - https://github.com/radanalyticsio/openshift-spark/pull/28
- Added agent to report jolokia metrics endpoint in openshift pod

- Added agent to report Prometheus metrics endpoint in openshift pod
SPARK APPLICATION

Source Data  Processing  Results
WHAT IS PROMETHEUS

- Open source monitoring
- In 2016 Prometheus became the 2nd member of the CNCF
- Scrapes metrics from an endpoint.
- Client libraries in Go, Java, Python, etc.
- Openshift comes instrumented out of the box with Prometheus endpoints.
- If you don’t have native integration with Prometheus there are lots of community exporters that allow lots of things to expose metrics in your infrastructure to get monitored.
WHAT IS APACHE SPARK

Apache Spark is an in-demand data processing engine with a thriving community and steadily growing install base

- Supports interactive data exploration in addition to apps
- Batch and stream processing
- Machine learning libraries
- Distributed
- Separate storage and compute (in memory processing)
- new external scheduler kubernetes
SPARK FEATURES

- Can run standalone, with yarn, mesos or **Kubernetes** as the cluster manager
- Has language bindings for Java, Scala, Python, and R
- Access data from JDBC, HDFS, S3 or regular filesystem
- Can persist data in different data formats: parquet, avro, json, csv, etc.
SPARK IN CONTAINERS
SPARK CLUSTER INSTRUMENT

- Notify alertmanager
- Scrapes metrics

SPARK MASTER
- JAVA AGENT

PROMETHEUS
- JAVA AGENT

ALERT MANAGER

SPARK WORKER
- JAVA AGENT
elif [ ${SPARK_METRICS_ON} == "prometheus" ]; then
    JAVA_AGENT="-javaagent:${SPARK_HOME}/agent-bond.jar=${SPARK_HOME}/conf/agent.properties"
    metrics="with prometheus metrics enabled"
else
    JAVA_AGENT="-javaagent:${SPARK_HOME}/jolokia-jvm-1.3.6-agent.jar=port=7777,host=0.0.0.0"
    metrics="with jolokia metrics enabled (deprecated, set SPARK_METRICS_ON to 'prometheus')"
fi

if [ -z ${SPARK_MASTER_ADDRESS+_} ]; then
    echo "Starting master$metrics"
    exec ${SPARK_HOME}/bin/spark-class$JAVA_AGENT org.apache.spark.deploy.master.Master
else
    echo "Starting worker$metrics, will connect to: ${SPARK_MASTER_ADDRESS}"
    while true; do
        echo "Waiting for spark master to be available ..."
        curl --connect-timeout 1 -s -X GET $SPARK_MASTER_UI_ADDRESS > /dev/null
        if [ $? -eq 0 ]; then
            break
        fi
        sleep 1
    done
    exec ${SPARK_HOME}/bin/spark-class$JAVA_AGENT org.apache.spark.deploy.worker.Worker ${SPARK_MASTER_ADDRESS}
<table>
<thead>
<tr>
<th>Targets</th>
<th>Endpoint</th>
<th>State</th>
<th>Labels</th>
<th>Last Scrape</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>kubernetes-apiservers (1/1 up)</strong></td>
<td><a href="https://10.19.47.23:8443/metrics">https://10.19.47.23:8443/metrics</a></td>
<td>UP</td>
<td><strong>instance=10.19.47.23</strong></td>
<td>47.748s ago</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.19.47.23</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.19.47.23</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>kubernetes-controllers (1/1 up)</strong></td>
<td><a href="https://10.19.47.23:8444/metrics">https://10.19.47.23:8444/metrics</a></td>
<td>UP</td>
<td><strong>instance=10.19.47.23</strong></td>
<td>35.983s ago</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.19.47.23</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.19.47.23</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=arn:st</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>spark-cluster-m-1-fq2dj (1/1 up)</strong></td>
<td><a href="https://10.128.6.141:7777/metrics">https://10.128.6.141:7777/metrics</a></td>
<td>UP</td>
<td><strong>instance=10.128.6.141</strong></td>
<td>16.504s ago</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.128.6.141</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>spark-cluster-w-1-b55mq (1/1 up)</strong></td>
<td></td>
<td>UP</td>
<td><strong>instance=10.128.6.141</strong></td>
<td>16.504s ago</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>instance=10.128.6.141</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prometheus lets you configure how often to scrape and which endpoints to scrap. The Prometheus server will pull in the metrics that are configured.

```
# prometheus.yaml

global:
  scrape_interval: 15s
  evaluation_interval: 15s

alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - alertmanager:9093

rule_files:
  - "simple_rule.yml"

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```
• PromQL query is used to create rules to notify you if the rule is triggered.
• Currently alertmanager will receive the notification and is able to notify you via email, slack or other options (see docs for details).

```yaml
# simple_rule.yml
1 groups:
2   - name: spark.rules
3     rules:
4       - alert: SparkOutage
5         expr: up == 0
6         for: 5s
7         labels:
8           severity: critical
9         annotations:
10          description: erik spark cluster is down and out
11          summary: erik spark Instance down
```
- Powerful query language to get metrics on kubernetes cluster along with spark clusters.
- What are gauges and counters?

  Gauges: Latest value of metric
  Counters: Total number of event occurrences. Might be suffix "*total".

You can use this format to get the last minute `prom_metric_total[1m]`
Things we would like to know when tuning Spark programs:

- How much memory is the driver using?
- How much memory are the workers using?
- How is the JVM begin utilized by spark?
- Is my spark job saturating the network?
- What is the cluster view of network, cpu and memory utilization?

We will demonstrate how Prometheus coupled with Grafana on Kubernetes can help answer these types of questions. Visit our blog

“How to Gather and Display Metrics in Red Hat OpenShift”

https://red.ht/2CZAAhN
Focus on Memory:
Efficient Memory use is Key to good performance in Spark jobs.

How:
We will create Prometheus + Grafana dashboards to evaluate memory usage under different conditions?

Example:
Our Spark Python example will compare memory usage with and without caching to illustrate how memory usage and timing change for a PySpark program performing a cartesian product followed by a groupby operation.
A little Background

Memory allocation in Spark

- Spark is an "in-memory" computing framework
- Memory is a limited resource!
- There is competition for memory
- Caching reusable results can save overall memory usage under certain conditions
- Memory runs out in many large jobs forcing spills to disk
Spark Unified Memory Model

LRU eviction and user defined memory configuration options

Total JVM Heap Memory allocated to SPARK JOB

Memory allocated to EXECUTION

Memory allocated to STORAGE

EXECUTION takes precedence over STORAGE up to user defined unevictable amount

spill to disk

memory.storageFraction
Using Spark SQL and Spark RDD API together in a tuning exercise

**We want to use Spark SQL to manipulate dataframes**

Spark SQL is a component of Spark

- it provides structured data processing
- it is implemented as a library on top of Spark

**APIs:**

- SQL syntax
- **Dataframes**
- Datasets

**Backend components:**

- Catalyst - query optimizer
- Tungsten - off-heap memory management eliminates overhead of Java Objects
Performance Optimizations with Spark SQL

Spark SQL performance benefits:
- Catalyst compiles Spark SQL programs down to an RDD
- Tungsten provides more efficient data storage compared to Java objects on the heap
- Dataframe API and RDD API can be intermixed
Using Prometheus + Grafana for performance optimization

Specific code example:
Compare non-cached and cached dataframes that are reused in a groupBy transformation

When is good idea to use cache in a dataframe?
- when a result of a computation is going to be reused later
- when it is costly to recompute that result
- in cases where algorithms make several passes over the data
Determining memory consumption for dataframes you want to cache

<table>
<thead>
<tr>
<th>RDD Name</th>
<th>Storage Level</th>
<th>Cached Partitions</th>
<th>Fraction Cached</th>
<th>Size in Memory</th>
<th>Size on Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan ExistingRDD[E#9,F#10,G#11,H#12]</td>
<td>Memory Deserialized 1x Replicated</td>
<td>32</td>
<td>100%</td>
<td>6.1 MB</td>
<td>0.0 B</td>
</tr>
<tr>
<td>Scan ExistingRDD[A#0,B#1,C#2,D#3]</td>
<td>Memory Deserialized 1x Replicated</td>
<td>32</td>
<td>100%</td>
<td>6.1 MB</td>
<td>0.0 B</td>
</tr>
</tbody>
</table>
Example: Code for non-cached run

```python
rdd1 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
seed = 3
rdd2 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
sc = spark.sparkContext
# convert each tuple in the rdd to a row
randomNumberRdd1 = rdd1.map(lambda x: Row(A=float(x[0]), B=float(x[1]), C=float(x[2]), D=float(x[3])))
randomNumberRdd2 = rdd2.map(lambda x: Row(E=float(x[0]), F=float(x[1]), G=float(x[2]), H=float(x[3])))
# create dataframe from rdd
schemaRandomNumberDF1 = spark.createDataFrame(randomNumberRdd1)
schemaRandomNumberDF2 = spark.createDataFrame(randomNumberRdd2)
cross_df = schemaRandomNumberDF1.crossJoin(schemaRandomNumberDF2)
# aggregate
results = schemaRandomNumberDF1.groupby("A").agg(func.max("B"), func.sum("C"))
results.show(n=100)
print "----------Count in cross-join-------------- {0}".format(cross_df.count())
```
Example: Code for **cached** run

```python
rdd1 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
seed = 3
rdd2 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
sc = spark.sparkContext
# convert each tuple in the rdd to a row
randomNumberRdd1 = rdd1.map(lambda x: Row(A=float(x[0]), B=float(x[1]), C=float(x[2]), D=float(x[3])))
randomNumberRdd2 = rdd2.map(lambda x: Row(E=float(x[0]), F=float(x[1]), G=float(x[2]), H=float(x[3])))
# create dataframe from rdd
schemaRandomNumberDF1 = spark.createDataFrame(randomNumberRdd1)
schemaRandomNumberDF2 = spark.createDataFrame(randomNumberRdd2)
# cache the dataframe
schemaRandomNumberDF1.cache()
schemaRandomNumberDF2.cache()
cross_df = schemaRandomNumberDF1.crossJoin(schemaRandomNumberDF2)
# aggregate
results = schemaRandomNumberDF1.groupBy("A").agg(func.max("B"), func.sum("C"))
results.show(n=100)
print "----------Count in cross-join------------- (0)".format(cross_df.count())```
Query plan comparison

Non-Cached

Details for Query 1
Submitted Time: 2018/04/12 14:29:04
Duration: 22 s
Running Jobs: 3

Cached

Details for Query 1
Submitted Time: 2018/04/13 04:11:24
Duration: 4 s
Running Jobs: 4
Succeeded Jobs: 3
Example: Comparing cached vs non-cached runs
Example: Comparing cached vs non-cached runs

Prometheus dashboard: non-cached

Prometheus dashboard: cached
Comparing non-cached vs cached runs

**RIP (Relative Index of Performance)**
- **RIP:**
  - 0 to 1 = Improvement
  - 0 to -1 = Degradation

**% Change:**
- negative values = Improvement

<table>
<thead>
<tr>
<th>Pod Memory Usage / Time</th>
<th>non-Cached</th>
<th>Cached</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>RIP = 0.76</td>
<td>% Change = 76</td>
</tr>
<tr>
<td>Worker 1</td>
<td>RIP = 0.63</td>
<td>% Change = 63</td>
</tr>
<tr>
<td>Worker 2</td>
<td>RIP = 0.63</td>
<td>% Change = 63</td>
</tr>
<tr>
<td>Worker 3</td>
<td>RIP = 0.62</td>
<td>% Change = 62</td>
</tr>
<tr>
<td>Worker 4</td>
<td>RIP = 0.00</td>
<td>% Change = 0</td>
</tr>
<tr>
<td>Time (Min)</td>
<td>RIP = 0.10</td>
<td>% Change = 10</td>
</tr>
</tbody>
</table>
Demo Time!

SPARK JOB + PROMETHEUS + GRAFANA DEMO
Recap

You learned:

- About our story on spark cluster metrics monitoring with prometheus
- Spark Features
- How prometheus can be integrated with apache spark
- Spark Applications and how memory works
- Spark Cluster JVM Instrumentation
- How do I deploy a spark job and monitor it via grafana dashboard
- Performance difference between cache vs non-cached dataframes
- Monitoring tips and tricks
Thank You!

Questions?
Where To Find Us?

Try this at home: https://red.ht/2CZAAhN

Zak Hassan
Twitter: @Propect1010

Diane Feddema
Twitter: @DianeFeddema