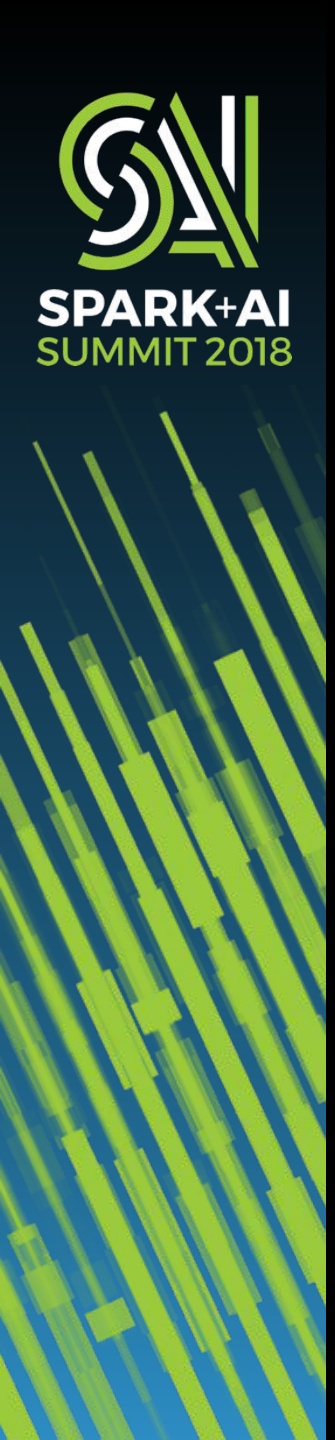


redhat

SCALABLE MONITORING USING PROMETHEUS WITH APACHE SPARK

Diane Feddema, Principal Software Engineer
Zak Hassan, Software Engineer

#Radanalytics

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

SOFTWARE ENGINEER - AI/ML CENTER OF EXCELLENCE, CTO OFFICE

- Currently focused on developing analytics platform on OpenShift and leveraging Apache Spark as the analytics engine. Also, developing data science apps and working on making metrics observable through cloud-native technology.
- Previously worked as a Software Consultant in the financial services and insurance industry, building end-to-end software solutions for clients.

redhat.

OVERVIEW

OBSERVABILITY

- Motivation
- What Is Spark?
- What Is Prometheus?
- Our Story
- 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

OUR STORY

- Instrumented spark jvm to expose metrics in a kubernetes 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 kubernetes pod

Nov 7, 2017 - https://github.com/radanalyticsio/openshift-spark/pull/35

- Added agent to report prometheus metrics endpoint in kubernetes pod

WHAT IS PROMETHEUS

- Open source monitoring
- in 2016 prometheus become the 2nd member of the CNCF
- scrapes metrics from a endpoint.
- Client libraries in **Go, Java, Python, etc**.
- Kubernetes 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.

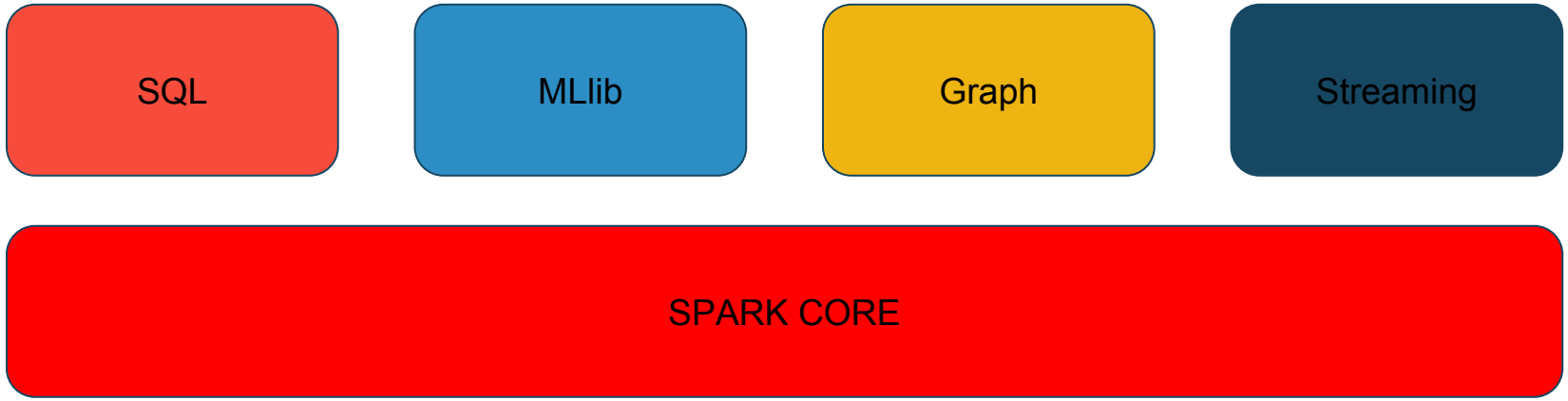
WHAT IS APACHE SPARK

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

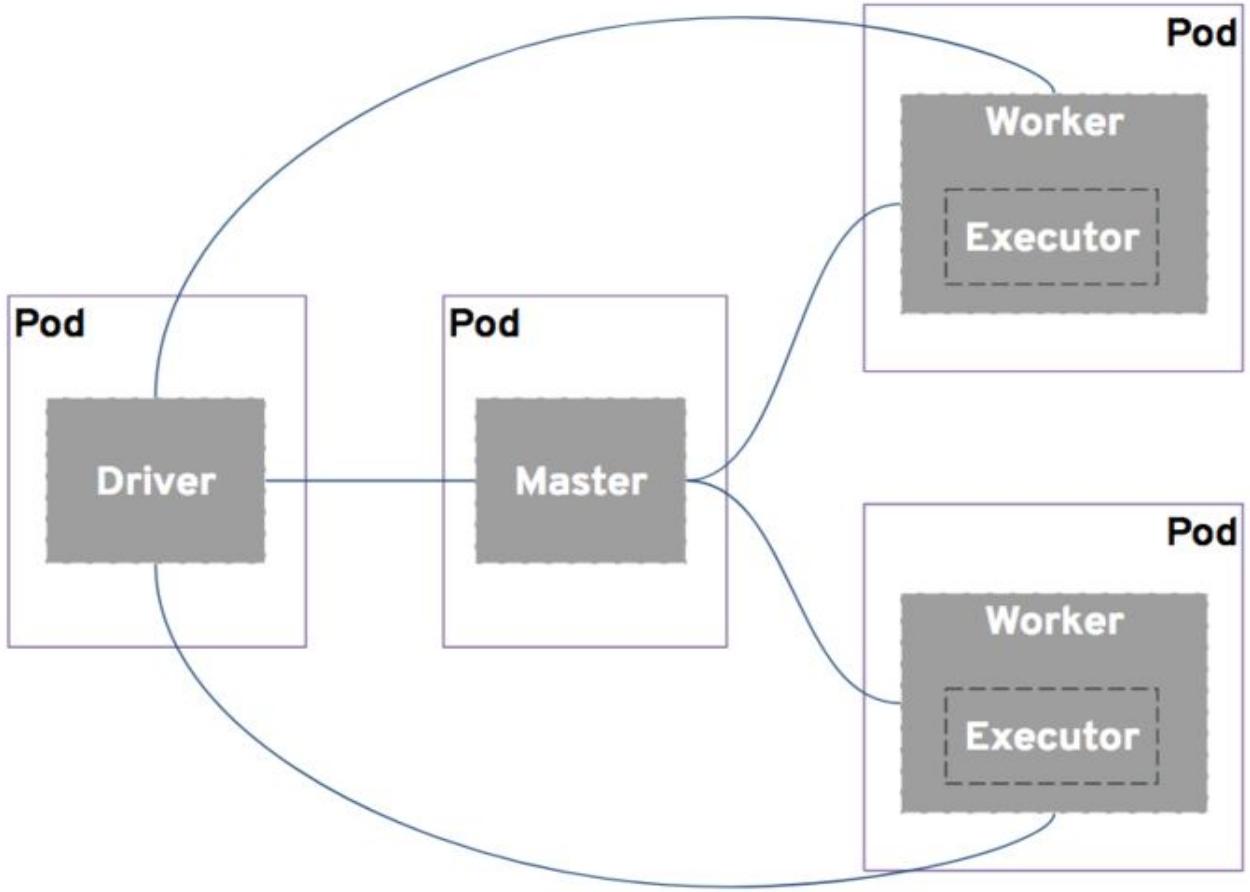
SPARK FEATURES

- 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.

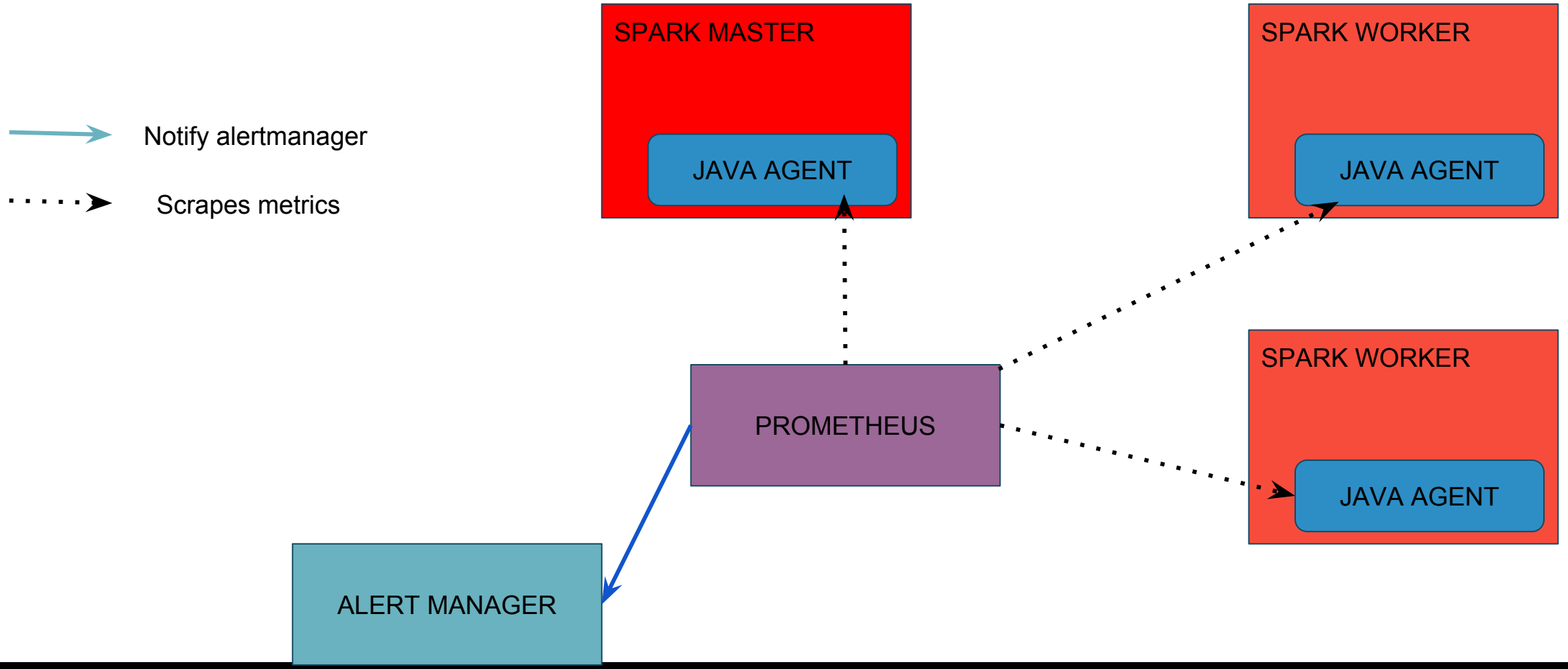


SPARK APPLICATION

SPARK IN CONTAINERS



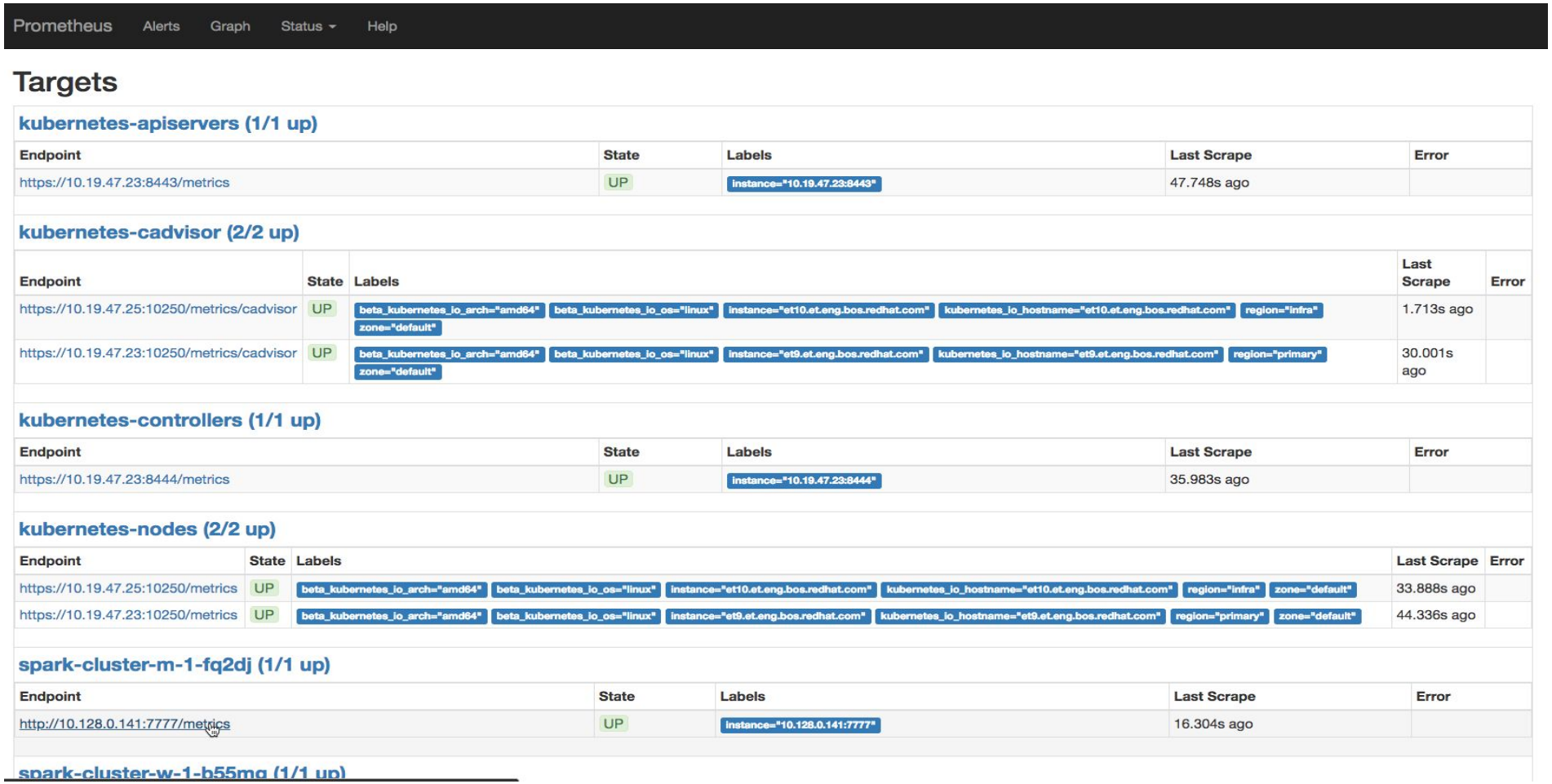
SPARK CLUSTER INSTRUMENT



INSTRUMENT JAVA AGENT

```
30  elif [ ${SPARK_METRICS_ON} == "prometheus" ]; then
31      JAVA_AGENT=" -javaagent:$SPARK_HOME/agent-bond.jar=$SPARK_HOME/conf/agent.properties"
32      metrics=" with prometheus metrics enabled"
33  else
34      JAVA_AGENT=" -javaagent:$SPARK_HOME/jolokia-jvm-1.3.6-agent.jar=port=7777,host=0.0.0.0"
35      metrics=" with jolokia metrics enabled (deprecated, set SPARK_METRICS_ON to 'prometheus')"
36  fi
37
38  if [ -z ${SPARK_MASTER_ADDRESS+_} ]; then
39      echo "Starting master$metrics"
40      exec $SPARK_HOME/bin/spark-class$JAVA_AGENT org.apache.spark.deploy.master.Master
41  else
42      echo "Starting worker$metrics, will connect to: $SPARK_MASTER_ADDRESS"
43      while true; do
44          echo "Waiting for spark master to be available ..."
45          curl --connect-timeout 1 -s -X GET $SPARK_MASTER_UI_ADDRESS > /dev/null
46          if [ $? -eq 0 ]; then
47              break
48          fi
49          sleep 1
50      done
51      exec $SPARK_HOME/bin/spark-class$JAVA_AGENT org.apache.spark.deploy.worker.Worker $SPARK_MASTER_ADDRESS
```

PROMETHEUS TARGETS



PULL METRICS

- 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']
```

ALERTMANAGER

- 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) .

simple_rule.yml

```
groups:
- name: spark.rules
  rules:
  - alert: SparkOutage
    expr: up == 0
    for: 5s
    labels:
      severity: critical
    annotations:
      description: erik spark cluster is down and out
      summary: erik spark Instance down
```

PROMQL

- 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]**

PART 2: Tuning Spark jobs with Prometheus

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.

Our Example Application

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

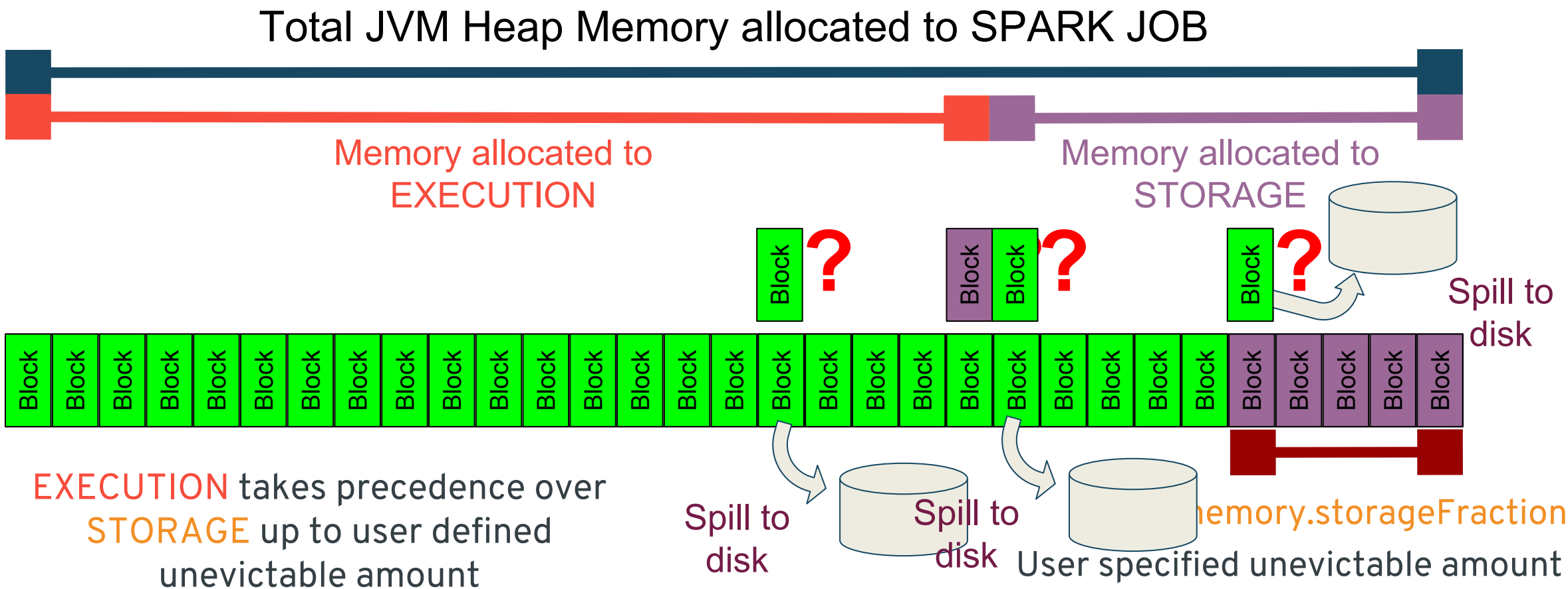
A little Background

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



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

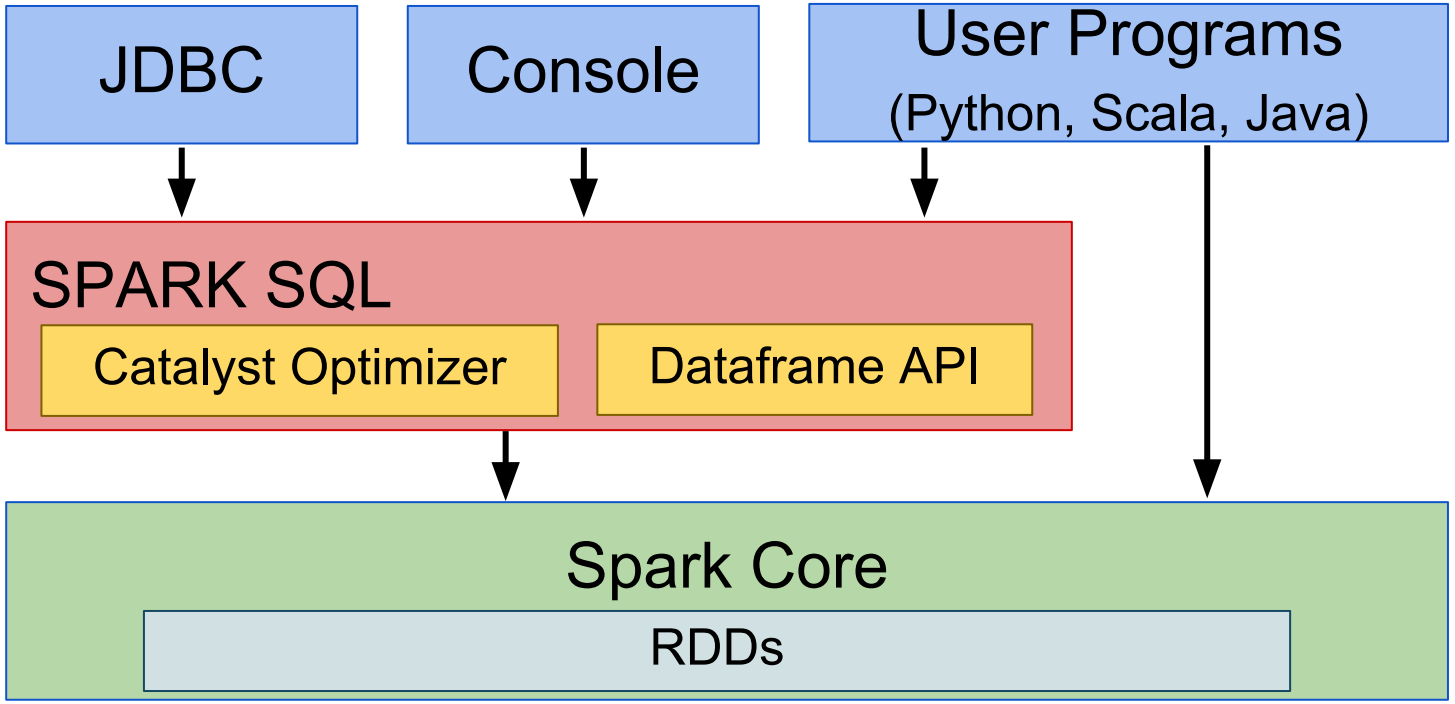
APIs:

- 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

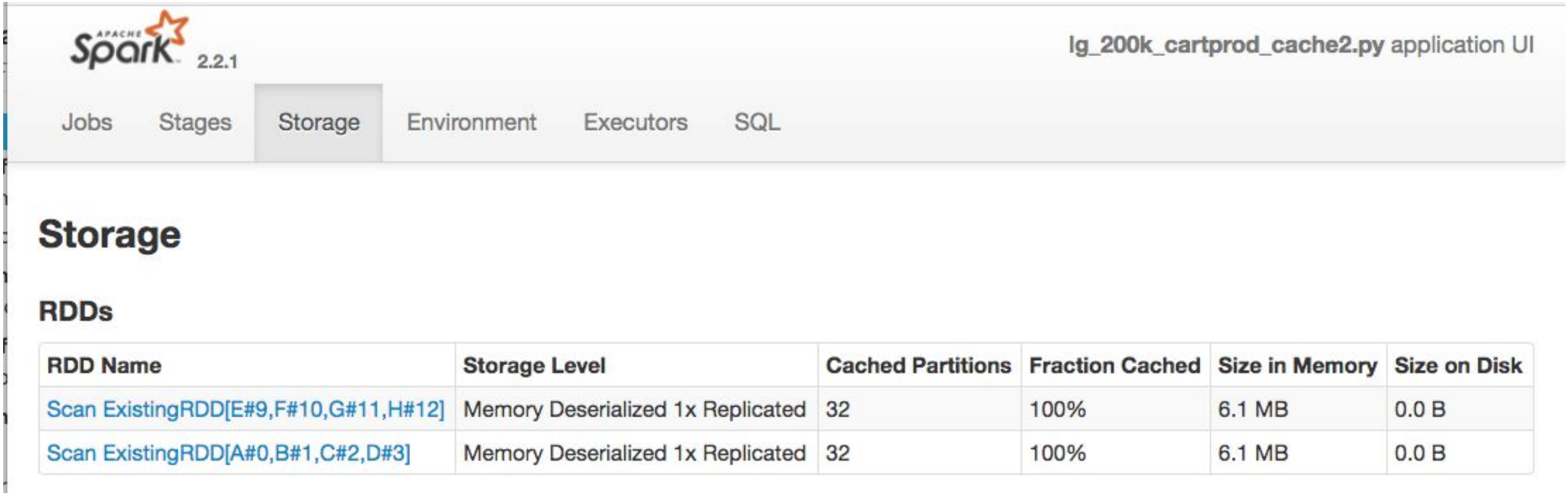
Specific code example:

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



Example: Code for non-cached run

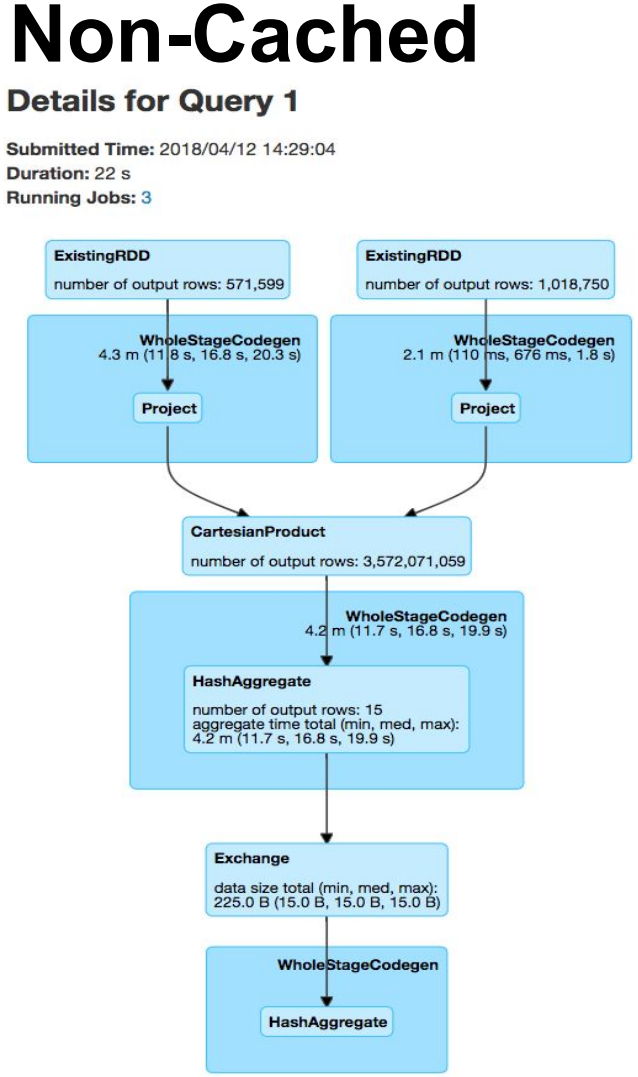
```
rdd1 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
seed = 3
rdd2 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
sc = spark.sparkContext
# convert each tuple in the rdd to a row
randomNumberRdd1 = rdd1.map(lambda x: Row(A=float(x[0]), B=float(x[1]), C=float(x[2]), 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

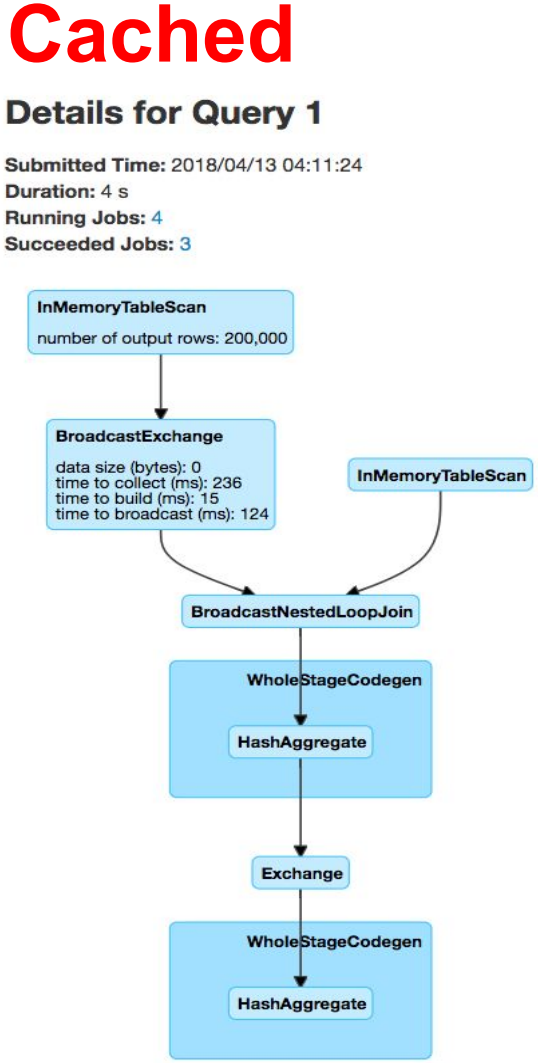
```
rdd1 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
seed = 3
rdd2 = RandomRDDs.normalVectorRDD(spark, nRow, nCol, numPartitions, seed)
sc = spark.sparkContext
# convert each tuple in the rdd to a row
randomNumberRdd1 = rdd1.map(lambda x: Row(A=float(x[0]), B=float(x[1]), C=float(x[2]), 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



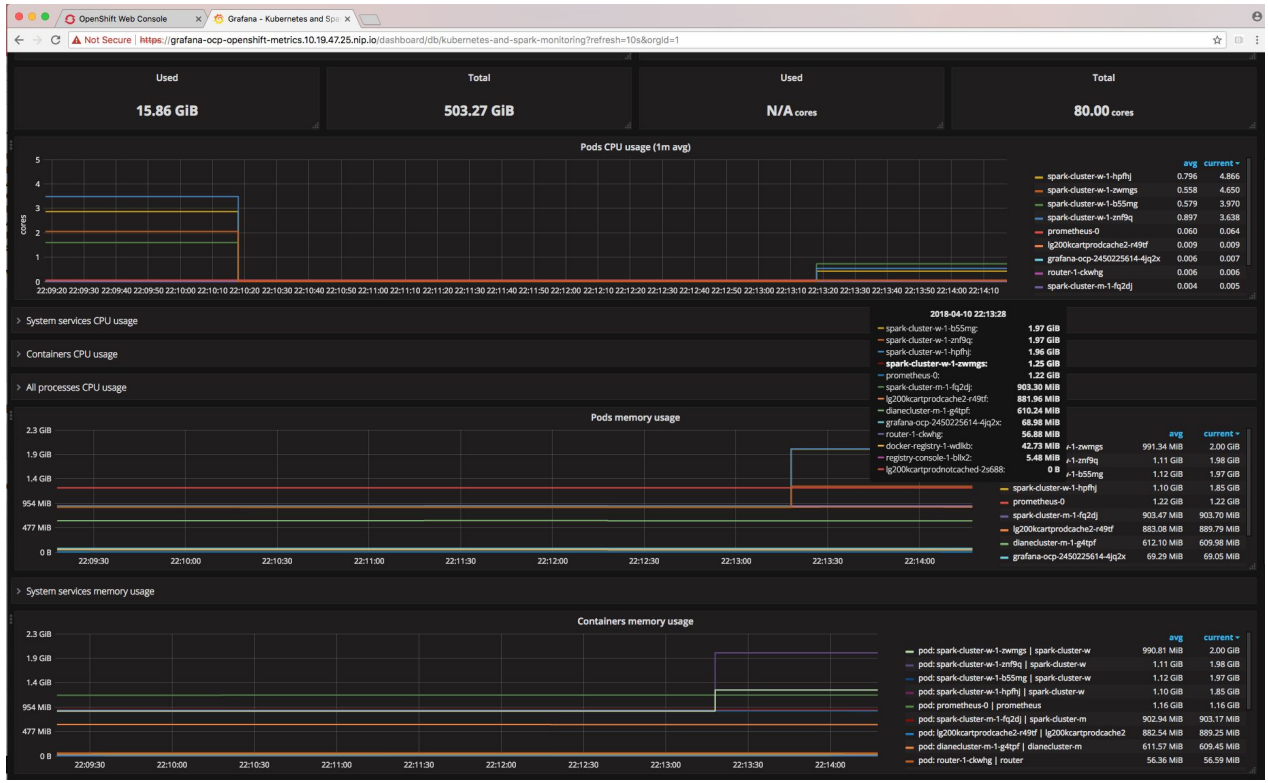
Cached



Example: Comparing cached vs non-cached runs

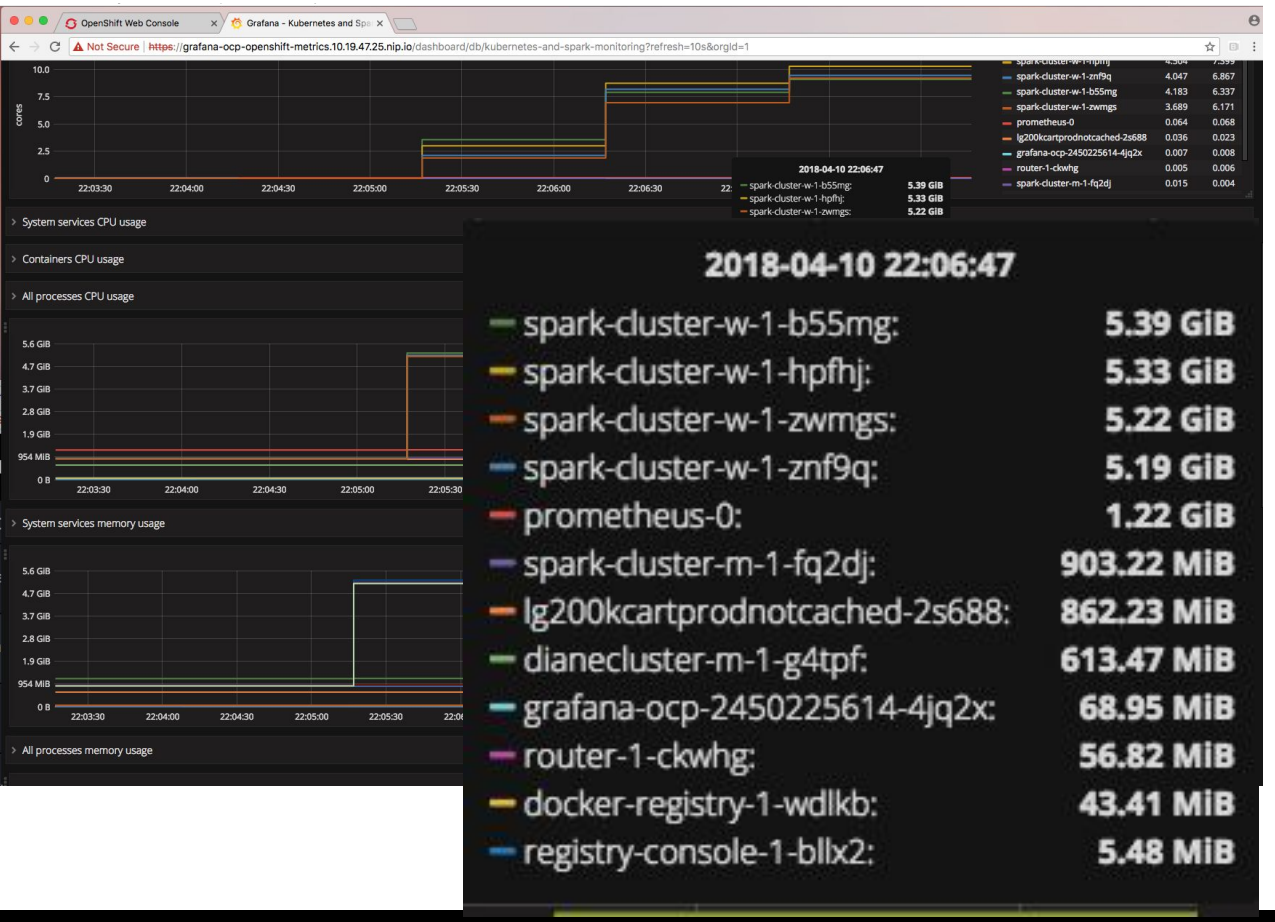
Prometheus dashboard: non-cached

Prometheus dashboard: cached

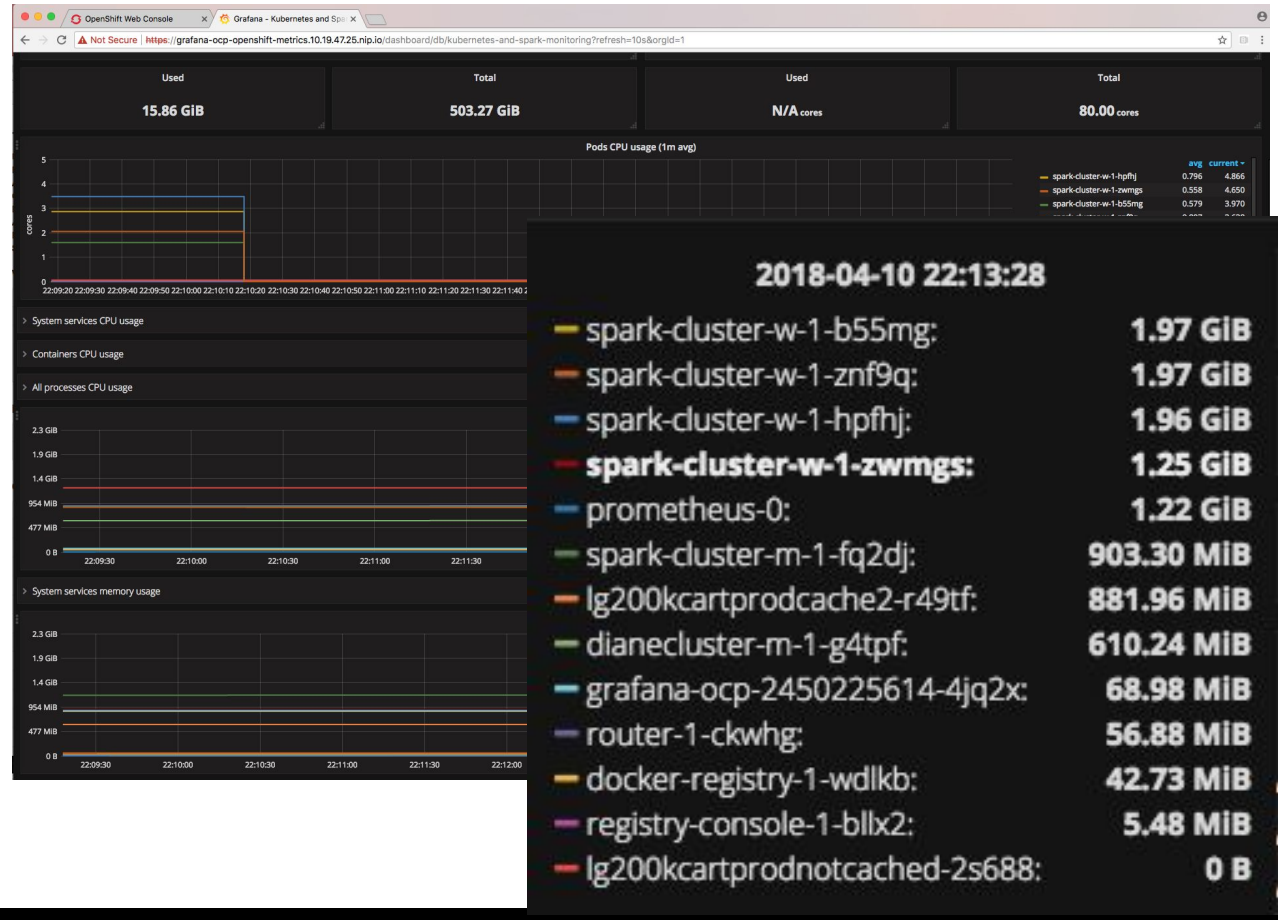


Example: Comparing cached vs non-cached runs

Prometheus dashboard: non-cached



Prometheus dashboard: cached



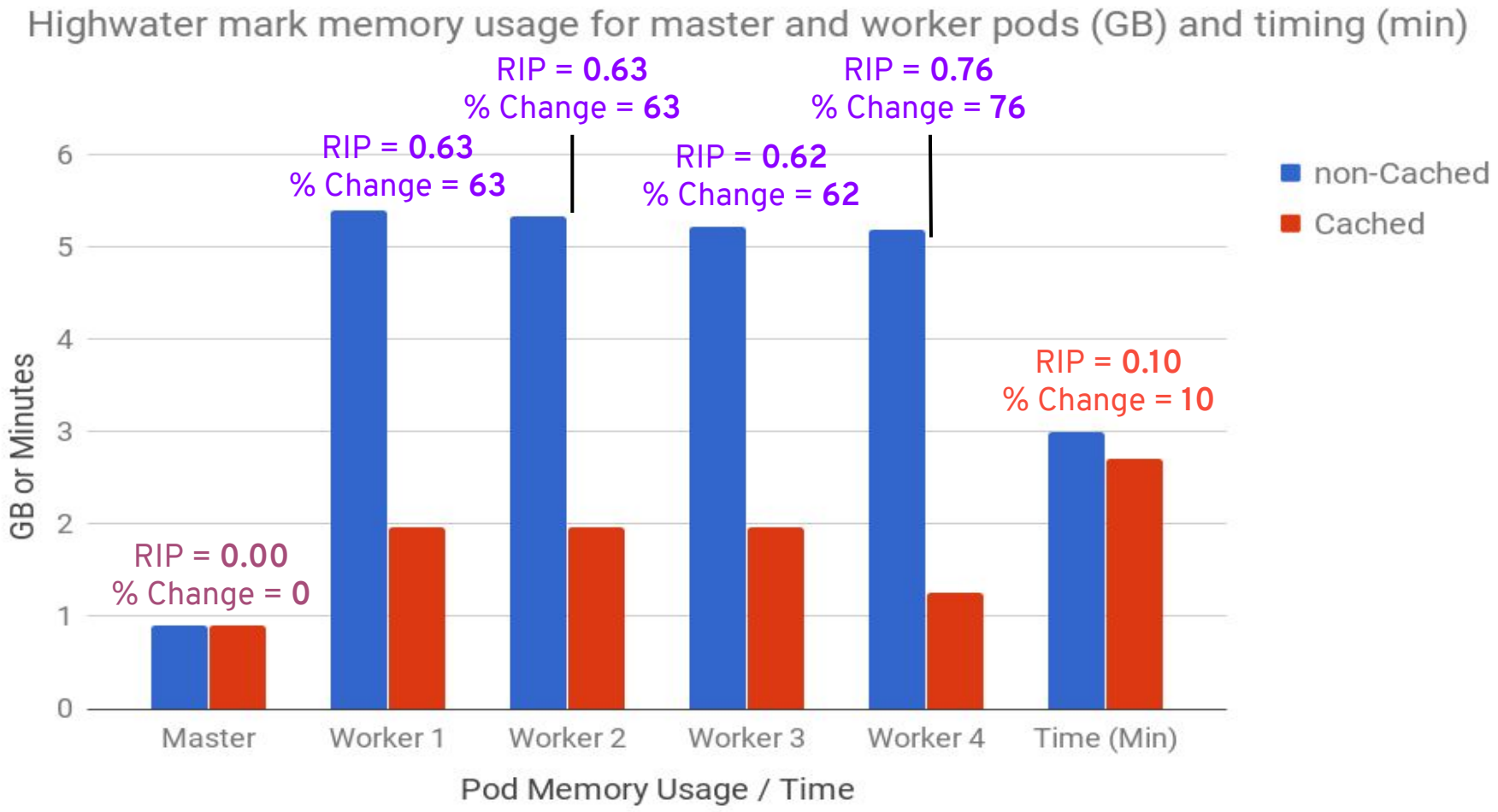
Comparing non-cached vs cached runs

RIP (Relative Index of Performance)

RIP: 0 to 1 = Improvement

0 to -1 = Degradation

% Change: negative values = Improvement



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?