Word Count and Page Rank using PySpark

By Mikaela Montaos
Prepared under the direction of Professor Henry Chang

School of Engineering
Northwestern Polytechnic University
117 Fourier Ave, Fremont, CA 94539
April 2021

Table of Contents

Introduction

- Platform for this project is Google Cloud Platform
- Technology used
  - Google Kubernetes Engine (GKE)
  - PySpark
- Functions
  - Word Count
  - Page Rank

Design

- PySpark is a combination of Apache Spark and Python
- Apache Spark
  - Fast, in-memory data processing engine which allows data workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets
- Advantages of Spark
  - Run computations in memory
    - 100x faster in memory and 10x faster even when running on disk than MapReduce
  - Easy to combine different processing models seamlessly in the same application
- Python is a general purpose, high level programming language. It is commonly used for machine learning and real-time streaming analytics

Implementation

1. Create a cluster on GKE

```
gcloud container clusters create spark --num-nodes=1 --machine-type=e2-highmem-2 --region=us-west1
```

```
NAME   LOCATION   MASTER_VERSION   MASTER_IP     MACHINE_TYPE   NODE_VERSION     NUM_NODES   STATUS
spark  us-west1   1.18.16-gke.502  35.230.50.43  e2-highmem-2   1.18.16-gke.502  3           RUNNING
```

2. Install NFS server provisioner

```
helm repo add stable https://charts.helm.sh/stable
```

```
montaos19518@cloudshell:~ (cs571-306821)$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

```
helm install nfs stable/nfs-server-provisioner \
--set persistence.enabled=true,persistence.size=5Gi
```

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ helm install nfs stable/nfs-server-provisioner \
> --set persistence.enabled=true,persistence.size=5Gi

WARNING: This chart is deprecated
NAME: nfs
LAST DEPLOYED: Thu Apr 22 17:53:27 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The NFS Provisioner service has now been installed.

A storage class named 'nfs' has now been created
and is available to provision dynamic volumes.

You can use this storageclass by creating a `PersistentVolumeClaim` with the
correct storageClassName attribute. For example:

    ---
    kind: PersistentVolumeClaim
    apiVersion: v1
    metadata:
      name: test-dynamic-volume-claim
    spec:
      storageClassName: "nfs"
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 100Mi
```

Implementation

3. Create a persistent disk volume and a pod to use NFS

```
montaos19518@cloudshell:~ (cs571-new)$ cat spark-pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
    name: spark-data-pvc
spec:
    accessModes:
        - ReadWriteMany
    resources:
        requests:
            storage: 2Gi
    storageClassName: nfs
---
apiVersion: v1
kind: Pod
metadata:
    name: spark-data-pod
spec:
    volumes:
        - name: spark-data-pv
          persistentVolumeClaim:
            claimName: spark-data-pvc
    containers:
        - name: inspector
          image: bitnami/minideb
          command:
            - sleep
            - infinity
          volumeMounts:
            - mountPath: "/data"
              name: spark-data-pv
```

Implementation

4. Apply the YAML descriptor

```
kubectl apply -f spark-pvc.yaml
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f spark-pvc.yaml
persistentvolumeclaim/spark-data-pvc created
pod/spark-data-pod created
```

5. Create and prepare the app JAR file

```
docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
```

```
montaos19518@cloudshell:~ (cs571-new)$ docker run -v /tmp:/tmp -it bitnami/spark -- find /opt/bitnami/spark/examples/jars/ -name spark-examples* -exec cp {} /tmp/my.jar \;
Unable to find image 'bitnami/spark:latest' locally
latest: Pulling from bitnami/spark
f87be781ad7c: Pull complete
08a308c29bf9: Pull complete
0595528da3a0: Pull complete
a895f572d643: Pull complete
f1a363eccc8d: Pull complete
d833c8086411: Pull complete
65def5cdfce9: Pull complete
0c6b4e1e73e7: Pull complete
29e6a48f7b02: Pull complete
8693a54dced2: Pull complete
Digest: sha256:f5ea4708250bcc4cb6b97f9e7978c1f40bb1402108b5c9a80781bc97fc7d233d
Status: Downloaded newer image for bitnami/spark:latest

Welcome to the Bitnami spark container
Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-spark
Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-spark/issues
```

Implementation

6. Add a test file with multiple line of words for the word count

```
echo "Having knowledge but lacking the power to express it clearly is no better than never having any ideas at all" > /tmp/test.txt
```

7. Copy the JAR file containing the application and other required files to the PVC using a mount point

```
kubectl cp /tmp/my.jar spark-data-pod:/data/my.jar
kubectl cp /tmp/test.txt spark-data-pod:/data/test.txt
```

8. Make sure the files are in the persistent volume (PV)

```
kubectl exec -it spark-data-pod -- ls -al /data
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl exec -it spark-data-pod -- ls -al /data
total 1504
drwxrwsrwx 2 root root    4096 Apr 22 18:00 .
drwxr-xr-x 1 root root    4096 Apr 22 17:58 ..
-rw-r--r-- 1 1001 root 1527168 Apr 22 18:00 my.jar
-rw-r--r-- 1 1000 1001     109 Apr 22 18:00 test.txt
```

9. Deploy Apache Spark on Kubernetes using the shared volume

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ cat spark-chart.yaml
service:
    type: LoadBalancer
worker:
    replicaCount: 3
    extraVolumes:
        - name: spark-data
          persistentVolumeClaim:
            claimName: spark-data-pvc
    extraVolumeMounts:
        - name: spark-data
          mountPath: /data
```

10. Deploy Apache Spark using Bitnami Apache Spark Helm Chart and supply it with the configuration YAML file

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

```
montaos19518@cloudshell:~ (cs571-306821)$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

Implementation

```
helm install spark bitnami/spark -f spark-chart.yaml
```

```
montaos195188@cloudshell:~ (cm571-new)$ helm install spark bitnami/spark -f spark-chart.yaml
NAME: spark
LAST DEPLOYED: Thu Apr 22 18:04:21 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the Spark master WebUI URL by running these commands:

    NOTE: It may take a few minutes for the LoadBalancer IP to be available.
    You can watch the status of by running 'kubectl get --namespace default svc -w spark-master-svc'

  export SERVICE_IP=$(kubectl get --namespace default svc spark-master-svc -o jsonpath="{.status.loadBalancer.ingress[0]['ip', 'hostname'] }")
  echo http://$SERVICE_IP:80

2. Submit an application to the cluster:

  To submit an application to the cluster the spark-submit script must be used. That script can be
  obtained at https://github.com/apache/spark/tree/master/bin. Also you can use kubectl run.

  Run the commands below to obtain the master IP and submit your application.

  export EXAMPLE_JAR=$(kubectl exec -ti --namespace default spark-worker-0 -- find examples/jars/ -name 'spark-example*\.jar' | tr -d '\r')
  export SUBMIT_IP=$(kubectl get --namespace default svc spark-master-svc -o jsonpath="{.status.loadBalancer.ingress[0]['ip', 'hostname'] }")

  kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
    --image docker.io/bitnami/spark:3.1.1-debian-10-r42 \
    -- spark-submit --master spark://$SUBMIT_IP:7077 \
    --deploy-mode cluster \
    --class org.apache.spark.examples.SparkPi \
    $EXAMPLE_JAR 1000

** IMPORTANT: When submit an application the --master parameter should be set to the service IP, if not, the application will not resolve the master. **

** Please be patient while the chart is being deployed **
```

Implementation

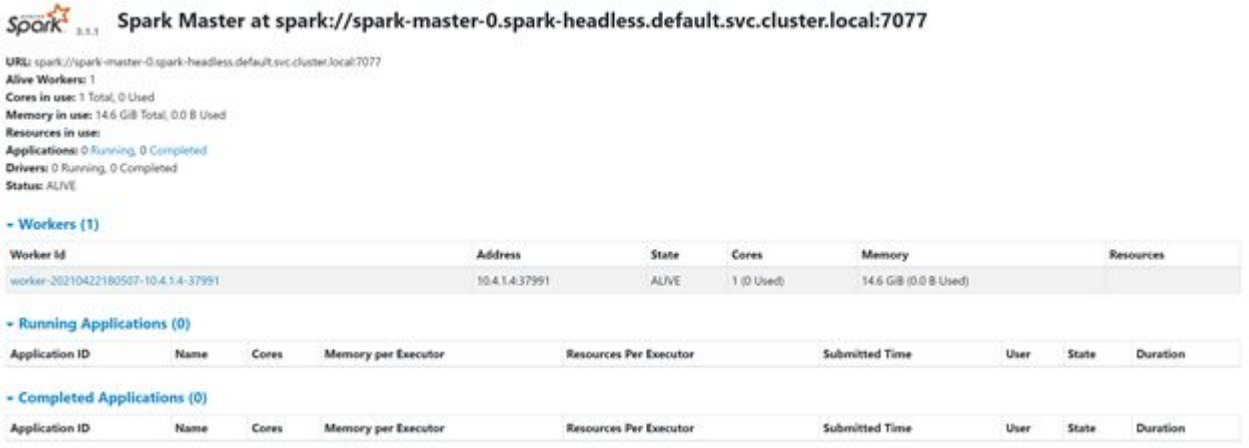
11. Get the external IP of the running pod

```
kubectl get svc -l "app.kubernetes.io/instance=spark,app.kubernetes.io/name=spark"
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get svc -l "app.kubernetes.io/instance=spark,app.kubernetes.io/name=spark"
NAME                TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)                      AGE
spark-headless      ClusterIP      None          <none>         <none>                       43s
spark-master-svc    LoadBalancer   10.7.250.8    34.83.40.52    7077:32236/TCP,80:30207/TCP  43s
```

12. Use your browser to open the external IP

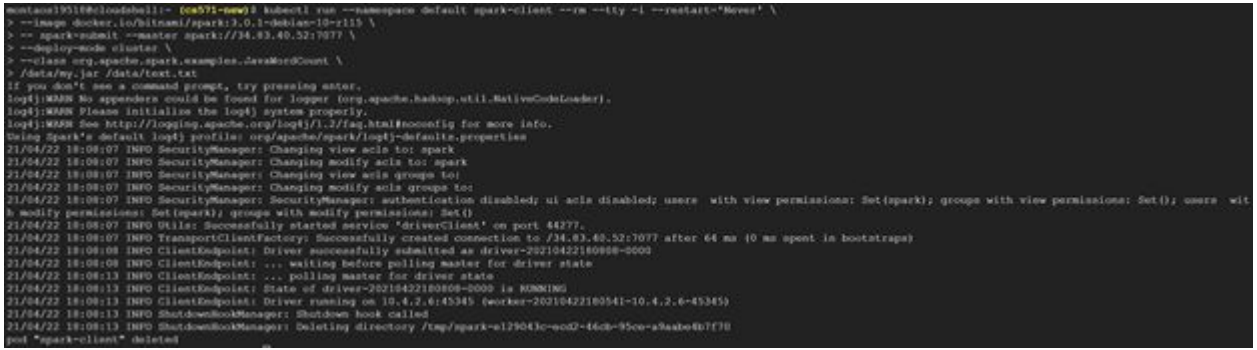
```
http://<external IP>
```



Test

1. Submit a word count task

```
kubectl run --namespace default spark-client --rm --tty -i --restart='Never' \
--image docker.io/bitnami/spark:3.0.1-debian-10-r115 \
-- spark-submit --master spark://<external IP>:7077 \
--deploy-mode cluster \
--class org.apache.spark.examples.JavaWordCount \
/data/my.jar /data/text.txt
```



Test

2. Refresh the browser to see the completed task

Spark Master at spark://spark-master-0.spark-headless.default.svc.cluster.local:7077

URL: spark://spark-master-0.spark-headless.default.svc.cluster.local:7077
Alive Workers: 3
Cores in use: 3 Total, 0 Used
Memory in use: 43.9 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 2 Completed
Drivers: 0 Running, 2 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210422180507-10.4.1.4-37991	10.4.1.4:37991	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210422180541-10.4.2.6-45345	10.4.2.6:45345	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	
worker-20210422180619-10.4.0.11-35343	10.4.0.11:35343	ALIVE	1 (0 Used)	14.6 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Running Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration

Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210422181814-0001	JavaWordCount	2	1024.0 MiB		2021/04/22 18:18:14	spark	FINISHED	13 s
app-20210422180813-0000	JavaWordCount	2	1024.0 MiB		2021/04/22 18:08:13	spark	FINISHED	2 s

Completed Drivers (2)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20210422181809-0001	2021/04/22 18:18:09	worker-20210422180507-10.4.1.4-37991	FINISHED	1	1024.0 MiB		org.apache.spark.examples.JavaWordCount

Test

3. Get the name of the worker node

```
kubectl get pods -o wide | grep <worker node address>
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get pods -o wide | grep 10.4.1.4
spark-worker-0                1/1     Running   0          15m   10.4.1.4    gke-spark-default-pool-69b5008a-7k9g   <none>           <none>
```

4. Execute the pod to see the result of the task

```
kubectl exec -it <spark-worker name> -- bash
cd /opt/bitnami/spark/work
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl exec -it spark-worker-0 -- bash
I have no name!@spark-worker-0:/opt/bitnami/spark$ cd /opt/bitnami/spark/work
I have no name!@spark-worker-0:/opt/bitnami/spark/work$ cat driver-20210422181809-0001/stdout
it: 1
is: 1
than: 1
ideas: 1
never: 1
having: 1
no: 1
all: 1
express: 1
to: 1
at: 1
lacking: 1
Having: 1
better: 1
any: 1
power: 1
but: 1
knowledge: 1
clearly: 1
the: 1
```

Test

5. Execute the spark master pod (You will need to do a portion of the Word Count tutorial to generate this pod)

   `kubectl exec -it spark-master-0 -- bash`

6. Start pyspark: `pyspark`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl exec -it spark-master-0 -- bash
I have no name!@spark-master-0:/opt/bitnami/spark$ pyspark
Python 3.6.13 (default, Apr 19 2021, 18:12:00)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/04/22 18:26:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Python version 3.6.13 (default, Apr 19 2021 18:12:00)
Spark context Web UI available at http://spark-master-0.spark-headless.default.svc.cluster.local:4040
Spark context available as 'sc' (master = local[*], app id = local-1619115966209).
SparkSession available as 'spark'.
```

7. Exit pyspark: `exit()`

```
>>> exit()
I have no name!@spark-master-0:/opt/bitnami/spark$
```

Test

8. Go to the directory where pagerank.py is located:

```
cd /opt/bitnami/spark/examples/src/main/python
```

9. Run pagerank.py using pyspark (/opt is the directory; 2 is the number of iterations to run pagerank)

```
spark-submit pagerank.py /opt 2
```

Test

```
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/iotanalytics/2017-11-27
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pytz/zoneinfo/america/argentina
file:/opt/bitnami/java/demo/jvmti/compiledmethodload/lib
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/cloudfront/2019-03-26
file:/opt/bitnami/spark/data/mllib/images/origin/kittens
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pytz
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/dlm/2018-01-12
file:/opt/bitnami/java/jre/bin
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pandas/tests/groupby/aggregate
file:/opt/bitnami/java/sample/jmx/jmx-scandir/src/com/sun/jmx/examples/scandir/config
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/waf-regional
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pandas/io/formats
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/pinpoint
file:/opt/bitnami/python/lib/python3.6/test/test_email
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/kafka/2018-11-14
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/mediatailor/2018-04-23
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/numpy/ma
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/xray
file:/opt/bitnami/java/demo/applets/moleculeviewer
file:/opt/bitnami/spark/examples/src/main/python/sql/streaming
file:/opt/bitnami/spark/examples/src/main/resources/dir1
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/docutils/writers/latex2e
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pandas/tests/extension/base
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/iotsecuretunneling/2018-10-05
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/serverlessrepo
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/numpy/random/tests
file:/opt/bitnami/java/demo/nbproject/management/jtop
file:/opt/bitnami/python/lib/python3.6/test/test_importlib/extension
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/ec2/2014-10-01
file:/opt/bitnami/python/lib/python3.6/test/subprocessdata
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/boto3/examples
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/resourcegroupstaggingapi/2017-01-26
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/iotsitewise/2019-12-02
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/greengrass/2017-06-07
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/boto3-1.17.53.dist-info
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/ec2
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/pandas/tests/indexes/period
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/personalize-events/2018-03-22
file:/opt/bitnami/python/lib/python3.6/test/cjkencodings
file:/opt/bitnami/java/demo/applets/wireframe
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/cur/2017-01-06
file:/opt/bitnami/spark/examples/src/main/java/org/apache/spark/examples/mllib
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/appstream/2016-12-01
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/secretsmanager/2017-10-17
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/awscli/examples/redshift
file:/opt/bitnami/spark/python/pyspark/python/pyspark
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/elb/2012-06-01
file:/opt/bitnami/spark/venv/lib/python3.6/site-packages/botocore/data/ec2/2015-03-01
```

Enhancement ideas

- Create a game
  - In-game events require quick responses
  - Apache Spark can handle the speed, variety and volume of the data

Conclusion

- PySpark is used in scalable data analysis, building machine learning pipelines, and creating ETLs for a data platform.
- PySpark is fast because of lazy execution
  - Lazy execution means that the operation is does not execute until it is needed

Bibliography

Bernardo P. M., Tao W., Lee J. (2018, April). *Apache Spark with Python - Big Data with PySpark and Spark* [Video]. O'Reilly Online Learning. https://learning.oreilly.com/library/view/apache-spark-with/9781789133394/?cohort=false

Sharma, R. (2021, January 10). *12 Exciting Spark Project Ideas & Topics For Beginners [2021]*. UpGrad Blog. https://www.upgrad.com/blog/spark-project-ideas-topics-for-beginners/

Weber, B. (2018, December 16). *A Brief Introduction to PySpark*. Towards Data Science. https://towardsdatascience.com/a-brief-introduction-to-pyspark-ff4284701873