Matrix Computation with Apache Spark in the Cloud

Christopher Blöcker, Timotheus Kampik, Tobias Sundqvist

This repository (https://github.com/TimKam/wasp-cloud-course) contains documentation and example code for running matrix computations with Apache Spark in the cloud. The content was produced as part of the WASP Software Engineering and Cloud Computing Course 2019.

Problem

The problem to solve is scaling ***matrix computations*** with Apache Spark. Instead of just running them on a local machine, the computations are to be executed across several cloud instances to speed up execution time. To demonstrate the performance advantage of the cloud-based solution, any of the matrices that are available at https://sparse.tamu.edu/ and any of Spark's linear algebra methods can be picked. However, the choice of data set and methods should allow for a fruitful analysis.

Choice of Methods

To assess the performance of an Apache Spark cluster with different numbers of worker nodes for a linear algebra use case, we chose the following methods:

1. **Principal component analysis** (PCA): pyspark.mllib.linalg.distributed.RowMatrix.computePrincipalComponents;
2. **Multiply**: pyspark.mllib.linalg.distributed.RowMatrix.multiply

We picked the two methods, because we expect them to be commonly used in real-life scenarios.

Choice of Data Set

We selected a medium-sized data set by sorting the available data sets at https://sparse.tamu.edu/ by number of non-zero entries. The exact data set we selected is https://sparse.tamu.edu/PARSEC/Si87H76.

Setup

The Docker/Spark setup is taken from this blog post by Marco Villarreal.

Execution

Build the images with `make`. Then spin up the cluster with `docker-compose up -d`. Spin up a submit container to run something in the cluster with something like

```
docker run -ti --rm --network=$NETWORK -v <script path>:/scripts spark-
submit:2.3.1 /bin/bash
```

where `NETWORK` is the Docker network to which the containers are connected and `<script path>` is where the ***executables*** are. Find out the network with `docker network ls`.

To run the analysis code, execute:

```
python analysis.py -u <SparkUrl>
```

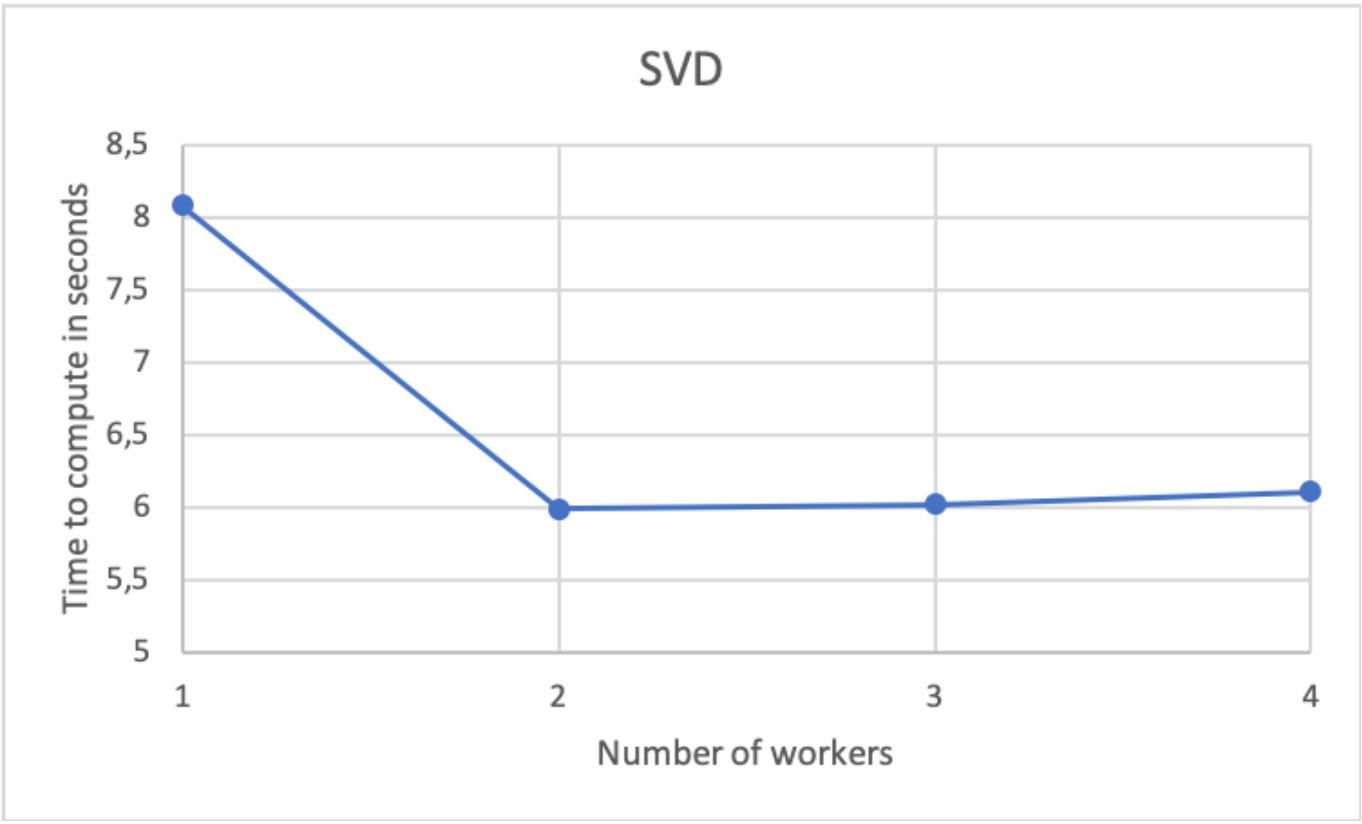
with `SparkUrl` being the URL to your Spark master, for example `spark://localhost:7077` if you run a local Spark cluster.

Resources

Analysis

We ran both operations on the aforementioned data set on 1, 2, 3, 4, and 5 workers in Google Cloud. Below are our results, averaged over 10 runs each:

	1 Worker	2 Workers	3 Workers	4 Workers
Multiply	0.7329313039779664 s	0.5070641040802002 s	0.8104101181030273 s	1.015389585494995 s
SVD	8.090139007568359 s	5.993102312088013 s	6.0258043050765995 s	6.112043023109436 s



As can be seen, ...