*Duplicate Image Detection Using Spark*

Clair Houlihan  
*Computer Science*  
*University of Texas at San Antonio*  
San Antonio, United States  
clairhoulihan@yahoo.com

Thomas Vasquez  
*Computer Science*  
*University of Texas at San Antonio*San Antonio, United States  
[zrm836@my.utsa.edu](mailto:zrm836@my.utsa.edu)

Julia Bowers  
*Computer Science*  
*University of Texas at San Antonio*San Antonio, United States  
juliabowers97@yahoo.com

Justin McCann  
 *Computer Science*  
*University of Texas at San Antonio*San Antonio, United States  
guz232@my.utsa.edu

***Abstract*—Duplicate image detection aims to detect images that are the same in composition. The process of finding duplicate images can take time by running on one system, especially if there are multiple images to be checked. By combining duplicate image detection software with Spark, the process of finding images of the same components can be shortened.**

***Keywords—Spark, Duplicate, Software***

# Introduction

Spark is a program that allows a master virtual machine to delegate tasks to slave virtual machines in order to split up the workload of performing calculations. Within our application that we are building, Spark will be one of the main components to use in our program to speed up the process of image detection instead of the use of one system. We will use multiple virtual machines to speed up this process.

Throughout this document we will go over the methods used for the program to get into it’s working state. And how those methods benefited us in reaching the goal have speeding up the process of image detection. Also, we will go over how easy it is to use the program that we created and what you can do with it and how it works.

At the end of how to use our program and describing the features and method we used, we will show the results of our testing and how that compares to other programs trying to accomplish the same goal and how much spark has improved the task trying to complete.

A discussion of what were the challenges we faced and how we solved those challenges in trying to design a program to work with spark and multiple virtual machines to complete our task of detecting duplicate images.

Our conclusions of our results will also be discussed to show what we came across while working on this project and what could have been done to improve the program even more if more time was allowed and the knowledge that we gained from this first testing run of the program.

References will of course be provided at the end of this report as well as a collection of abbreviations used and of course the contributions that were made to achieve this report and the information provided by the authors.

# Methods & Tools

## PySpark

Pyspark is a library available for python that uses a master virtual machine (VM), and at least one slave virtual machine(VM). The master virtual machine(VM) in this case runs the Python script, while the slave virtual machine(VM) runs the commands given to it by the master script(by use of the pyspark library). The pyspark class and it’s parameters that were available to us:

class pyspark.SparkContext (

master = None,

appName = None,

sparkHome = None,

pyFiles = None,

environment = None,

batchSize = 0,

serializer = PickleSerializer(),

conf = None,

gateway = None,

jsc = None,

profiler\_cls =<class'pyspark.profiler.BasicProfiler'>)

The way we used pyspark to help with the efficiency of detecting duplicate images is that the pyspark library gave us access to use multiple virtual machines to split the task of detecting images within a folder we designated as image. We only used one master and one slave virtual machine to test our program.

* 1. Pillow Library

Pillow is a python library that checks two images for similarities and determines whether the images are duplicates. This is only one function of the library, it also allows for general image processing, image archives, and displaying images as well. There is a simple function that is used to read an image with this library which is displayed here:

from PIL import Image

with Image.open("hopper.ppm")as im: …

We chose the one function as that one function met our testing needs for detecting the duplicate images the other methods although would be very helpful in the interest of time we decided on just getting the core of the program working and possibly adding the extras to the program at a later time.

* 1. OS library

The Operating System library (OS) is a library that is installed along with python. It is useful for a wide array of tasks when python needs to interact with the operating system of a computer.

Of course, the OS is always needed for any python programs this helps with the standard functions being used and enables to complete work without hassle no matter the Operating system to ensure that we can work without having to be bugged with systems that won’t work with simple syntax and have to debug which would slow us down.

* 1. Google Cloud

Google cloud is a cloud service that provides a platform as a service (PaaS). It is mainly used as it has several resources allocated towards it such as memory.

Google cloud came into play because it offered a free platform where we could test our program in the cloud with the two virtual machines we set up to test the program on Google’s cloud servers to see if there was any difference from running just from a research server where we did the same testing with a slightly different program.

* 1. Chameleon Cloud

Chameleon is an experimental platform built to support Computer Sciences systems research. Community projects range from systems research developing new operating systems, virtualization methods, performance variability studies, and power management research to projects in software defined networking, artificial intelligence, and resource management.

The Chameleon cloud was the main testing server for our program it used an easy to setup interface for our program and was widely accessible form a variety of different systems since logging onto the main virtual machines were so seamlessly and anyone with an ssh or secure shell as it is known as, can log in and access and work on our virtual machines given the right credentials.

With this simple way to work it helped during the time that we are where we can physically be there to work on the same program together so this tool was one of the big helps during the building of this program.

* 1. Python

A programming language which is used and is becoming more popular. We chose this language because with the pySpark library it only works with python language and instead of trying to reinvent the wheel with trying to get another language working we used the tools that were already available in order to create the program and just configure it to use multiple virtual machines.

* 1. SSH

SSH or secure shell is what it is known as is a tool that is used to securely log into a server or virtual machine in order to work off of or just log into different machines and set up other machines.

We used this standard tool in order to connect to the different virtual machines in order to work on them more efficiently and this has no problems since we can work on the project from anywhere there is a stable connection and we can securely use ssh. Even on mobile it is possible.

* 1. Python Time Function

Within the python standard library there is a timer class that we used in order to test the difference between the times of a regular duplicate image program and the one we created for the virtual machines to use. We also tested the google cloud version of our program with our chameleon version to see if there was any difference that would occur.

The Standard function for the time function that we used is displayed here which is:

from timer import Timer

from reader import feed

def main():

"""Print the latest tutorial from Real Python"""

t = Timer()

t.start()

t.stop()

With this simple function we were able to gather a time result to see if the method we are trying to would decrease the time it would take to find the duplicate images or if there was really not a difference to it. But this simple function made it easier to give us a testing mark to go against.

F. Imgur Album Downloader

We found a program on Github that allowed us to download entire albums from the image hosting website imgur. The program allowed us to download a large album to use in our testing. The program was written by Alex Gisby (alexgisby) and the repository can be found at <https://github.com/alexgisby/imgur-album-downloader>

.spark implementation provided my professor Lama and Chameleon cClo

# Discussion & difficulties

We first thought of the proposal to work on the detecting of duplicating images since it seems like a fun project to do and also will add to each of our skills set with being able to work on multiple virtual machine would be a great benefit in the future and present workloads if the needs for such a skill set

When we first started we set up two virtual machines as a base more like two mpi clusters if we are being more correct. Once we had the clusters up and running we moved onto try different implementations in order to achieve our goal

The first implementation went very well. We were able to use the pillow library to successfully detect duplicate images. We thought that was the end of the program but we were looking at the program and it was only running on one cluster. We decided to review our original proposal and discovered that we did not accomplish what we set out to initially accomplish. After several days of work we came up with a new program that better matched what we originally proposed.

1. Difficulties with two Virtual Machines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Methods | Run 1 | Run 2 | Run 3 | Avg. |
| With Spark | 10.29 | 10.32 | 10.14 | 10.25 |
| Without Spark | 2.97 | 2.79 | 2.79 | 2.85 |

When we looking more into how the first implementation went after looking at the log and seeing if the detection program we wrote worked on both the other virtual machine(VM) was not running at the same time and also was not splitting up the jobs so that was the first problem we ran into it took a while to fix but we ended up fixing the problem seemed to be with our pyspark not running the second virtual machine once that bug was fixed both seemed to be working.

1. Cloud and Virtual Machines Difficulties

During the work of the project the cloud servers to our virtual machines went down and could not be accessed either from the actual chameleon site or the SSH shells we were using.

Come later to find out that the servers that the chameleon supply for our virtual machines had a global wide shut down which really slowed us down as we could not move along with our plan to add more clusters in.

Because of this set back we had to move to another service to finish up our project

1. Google Clouds Difficulties

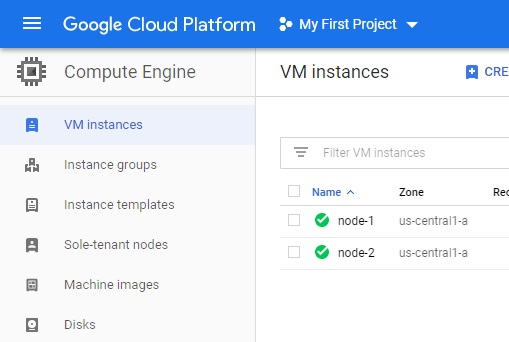
We decided to move over to Google's cloud services but because of this we add to redo the setup of our mpi cluster but since this service was different we had to work around the service.

The google cloud was bit difficult since we had to learn how it worked and how-to setup the Virtual machines which was a little different than setting up the virtual machines on the chameleon servers but once we got the running adding users and accessing it was about the same as the chameleon servers.

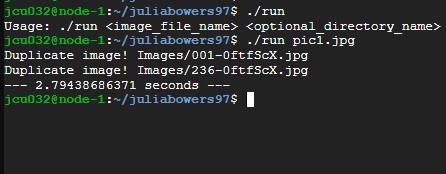
Once we started to add our program to it we found that even though we fixed the previous version of the program to work with the two virtual machines. It ended up being a problem again with the google virtual machines we set up not running together.

Thus we had another problem of trying to fix the program once again to run both virtual machines to get the duplicates detected with both virtual machines running at the same time.

Screenshot of our Google Cloud VM’s:



Screenshot of Version1.0 running on the Google VM:



1. Version 2.0

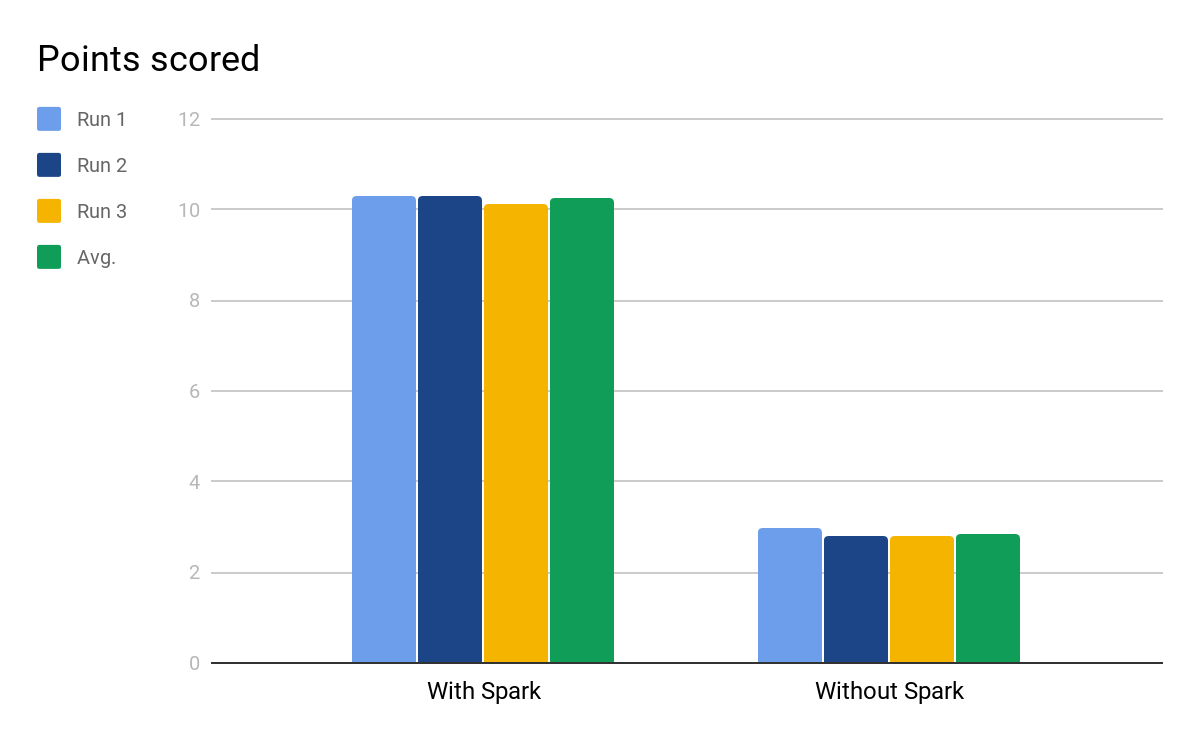
Our second iteration of the program functions just as we originally proposed. First it takes an image as the input parameter. Then it searches the image database for all image files and collects the hash of each file. Once all hashes are collected they are evenly distributed amongst the virtual machines in our cluster. Each process then checks the hashes for similarity and notes all files that are similar enough to consider them duplicates. After all processes finish running the program then prints out all images that are considered duplicates.

Even though we had to rework our entire program it was a learning experience for all of us. Our new program correctly utilizes the cluster we implemented during class to detect duplicate image files.

# Results

The results from the program came in about 10.25 seconds. Using the pillow module with spark context correctly worked at finding duplicate images. Running the pillow module without spark took 2.85 seconds to complete and came back with the same results as the pillow library using Spark.

Below is a table of the runs we have to see if there were differences of running it multiple times to get an avg of how much faster the image duplicates can be found with our program. Below are the results:

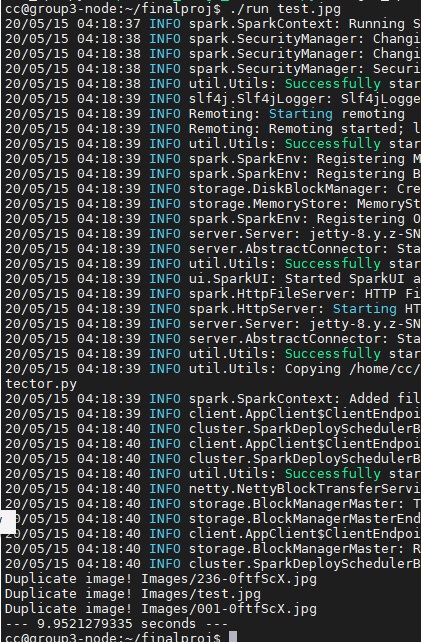


Running the program with Spark caused the program to run for a longer time since it had to set up the cluster before checking the images. The program without the Spark library only needed to open the directory containing the images and check each of the images against the original image.

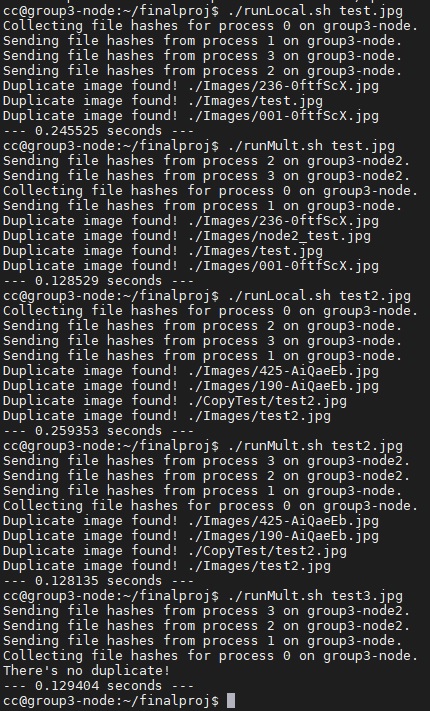
Also we kept a record of points received with the detection of duplicate images to see how many were scored with right while having it run with mpi clusters and without the help of the multiple clusters which shows that adding multiple virtual machines to the spark program will definitely help in detecting images faster than the standard of just running it on a standard python editor to find duplicate images.

Version 2.0 of our duplicate image detecting program better utilized the virtual machines within our cluster to accomplish its goal. Version 2.0 does not use the Python Pillow library that version 1.0 did, but this is for the better. Since version 2.0 did not use Apache Spark PySpark it also had greater efficiency since it did not have to boot. Using the hash of each image to check for duplicates also increased our efficiency.

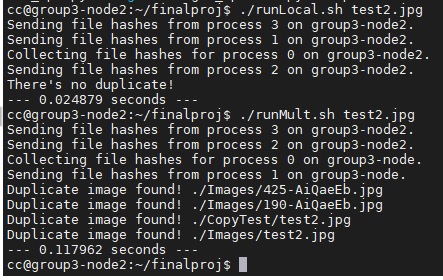
Screenshot of version 1.0 running:



Screenshot of version 2.0 running:



Screenshot of version2.0 running from node-2:



# CONCLUSIONS

Checking for duplicate images can be a time consuming process depending on how many images that are being checked. It is clear through the results of this experiment that running the program with a small number of images is inefficient, and going without the Spark cluster would be a better use of time in both the coding aspect and the execution. Both programs did exactly as expected, and got the correct images that were duplicates, so running with or without Spark had little to no effect on correct execution.

Even though the accuracy was high for finding duplicate images we could still see the improvement of the clusters being added as it made finding the duplicate images even faster than without the clusters as shown in our results section with the graphs displayed.

## Authors and Affiliations

Clair Houlihan did part of the tools section of the report, the conclusion and results portion, and provided some of the code for the project.

Justin McCann helped with formatting the report, added information about our second iteration into the report, worked on the presentation, and helped to work out bugs in the code.

Julia Bowers helped with the main program in developing most of the code for the project. she fixed a lot of the issues we came across and was able to implement what we needed for the project and ensure that the clusters worked.

Thomas Vasquez helped with Most of the report and formatted most of it while fixing some errors, tested the project and tried to fix some of the errors in the code and also ran it to test it.

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

[1] “pyspark,” *PyPI*. [Online]. Available: https://pypi.org/project/pyspark/. [Accessed: 15-May-2020]