PERFORMANCE ANALYSIS OF PROCESSING GHCN CLIMATE DATA USING CLUSTER COMPUTING AND APACHE SPARK

Madison Gipson and Lannie Hough
Department of Mathematics and Computer Science
Stetson University
DeLand, FL 32723
mgipson@stetson.edu, ldhough@stetson.edu

ABSTRACT

"Big data" describes a set of tools and methodologies designed to extract meaning from extremely large datasets. The size of the datasets in question often necessitate non-traditional approaches and tools like cluster computing and specialized frameworks like Apache Spark in order to process them efficiently. Big data is an increasingly important tool used to solve problems in areas such as medicine, commerce, advertising, and climate change. This paper demonstrates the use of big data techniques in order to analyze a GHCN climate dataset obtained from the National Oceanic and Atmospheric Administration. Some of the challenges often faced in big data are addressed, including the source and quality of data and how to sanitize it, as well as how to analyze and then visualize data and results. Furthermore, the findings from thorough investigations of the speed of the program performing computations on the dataset are presented here.

INTRODUCTION AND RELATED WORKS

Large dataset processing has many benefits, many of which stem from the fact that statistical power is increased as the size of a dataset increases. However, it also comes with certain challenges, largely due to the sheer size of data. To handle this, techniques like parallel computing are used, in which a dataset is divided and computed on in parallel. Frameworks like Apache Spark [5] have been created in order to aid developers and make this task less daunting, and in some cases more specialized tools have been developed or are in development, such as "Climate Spark." [1]

While it would seem apparent that increasing parallelism results in an increase in speed, there is a tradeoff between parallelism and performance in the form of delay in data transfer over a network between nodes. Considering details such as how many and what nodes to use, as well as larger-picture software design and algorithm implementation can help minimize overhead and significantly improve performance of parallel computing. There is active development in this area, including a proposal to use a variant of bin-packing algorithms. [4] The simulation detailed

later in this paper takes into consideration the impact of both the partitioning of data and the number of threads on parallelization performance.

The application of big data techniques specific to the field of climate research is unfortunately rather scarce compared to other areas such as marketing and e-commerce. [2] This is potentially due to the inherent complexity and analysis difficulty of climate data; however, huge climate datasets already exist and interest in this area is growing with climate change and global warming becoming more pressing societal issues. [3]

IMPLEMENTATION DETAILS

The intention of this simulation was to investigate the performance of processing a big dataset using cluster computing, parallel processing, and Apache Spark. We chose to observe the effect of partition and thread count on speed.

In terms of hardware, the cluster was comprised of four quad-core Raspberry Pis with 1 GB of RAM each, all running Raspbian. During the set-up process, Java 8, Spark, PySpark, FindSpark, and Pandas were installed. The cluster was managed with Apache Spark via SSH and the master-slave node relationship, with the other installations needed for program-specific computations.

The initial GHCN dataset obtained from NOAA consisted of a .dat file approximately 150 MB in size, which had lines containing information that represented the country of origin for the measurement, measuring station, and monthly temperature measurements for a given year, with the maximum range of years spanned from 1900 to 2020. However, not all countries had complete data; some had data for only the last several decades, some were sporadically missing measurements for months or years leaving gaps, some had an increase in data input due to measuring stations added overtime. This inconsistency in data was challenging to handle in a way that would produce accurate statistics, speaking to the aforementioned “complexity and analysis difficulty of climate data.”

The initial data format was suboptimal to operate on, so before making a program to process it, we first wrote a Python script to reformat it as a single line .json file, where each line represents a single JSON object. This is the optimal format for reading JSON with PySpark because it is easier to partition than a traditional JSON file. Each object in the file contained data for a single country and year, with a “month” field whose value contained nested arrays that present measurements from all stations for that month and year; reformatting the data in this way reduced the file size to approximately 100 MB.

Ultimately, we wanted to extract the average temperature change per year for a certain country over a given range of years, as well as the average temperature change during that time (average

change per year * number of years). To accomplish this, the 100 MB of JSON data was read into a single PySpark data frame, then filtered to only include the data relevant to the country and year range specified by the user. The array of months was split so that each month would have a column, with the different station readings within each column, then all the station readings for each month were averaged; this produced a highly consolidated data frame, with a maximum size of 120 rows by 14 columns, that was much easier to process in further calculations. This consolidated PySpark data frame was then converted to a Pandas data frame; this was done for better memory usage and for the later translation of the data frame to PyPlot.

country	months	year	Avg	JanAvg	FebAvg	MarAvg	AprAvg	MayAvg	JunAvg	JulAvg	AugAvg	SepAvg	OctAvg	NovAvg	DecAvg
US	[[5.82, 7.24, 4.4...	1900	11.46	0.26	-2.06	3.54	11.12	16.68	21.1	23.0	23.52	19.18	14.5	5.36	0.95
US	[[6.85, 6.1, 5.16...	1901	10.87	-0.73	-2.84	4.36	9.51	15.91	20.85	24.92	22.73	17.72	12.85	4.47	-0.98
US	[[5.58, 3.5, 7.29...	1902	11.05	-1.62	-1.82	5.89	10.3	17.09	19.87	22.67	21.56	17.25	12.74	7.53	-0.76
US	[[6.17, 4.34, 5.4...	1903	10.63	-1.07	-1.09	6.83	10.39	16.26	18.74	22.47	21.34	17.59	12.44	4.21	-1.69
US	[[4.21, 6.88, 4.7...	1904	10.4	-3.51	-1.99	4.86	9.07	16.09	19.96	21.96	21.24	18.45	12.22	5.93	-0.63
US	[[3.06, 2.0, 2.79...	1905	10.72	-3.65	-3.67	7.07	10.49	15.88	20.46	22.34	22.26	18.97	11.44	5.79	0.31
US	[[8.3, 4.84, 8.54...	1906	11.2	0.91	0.25	1.84	11.8	15.91	20.22	22.39	22.59	19.6	11.81	5.21	1.2
US	[[11.28, 12.52, 1...	1907	10.78	-0.56	0.36	7.57	8.05	13.53	19.02	22.74	21.54	18.16	11.71	5.22	1.62
US	[[5.84, 7.23, 7.3...	1908	11.4	-0.03	-0.17	6.55	11.54	15.7	19.94	22.99	21.55	19.05	11.7	6.3	1.18
US	[[8.81, 10.27, 10...	1909	10.87	-0.09	1.67	4.37	9.59	14.86	20.66	22.27	22.62	17.93	11.36	7.76	-3.14
US	[[6.34, 7.71, 6.1...	1910	11.17	-1.54	-1.7	9.6	12.0	14.9	19.91	23.18	21.57	18.62	13.4	4.65	-0.81
US	[[9.03, 10.04, 8....	1911	11.34	0.18	1.01	6.02	9.88	17.1	21.66	22.85	21.7	19.03	11.83	3.27	1.08
US	[[3.49, 5.31, 3.3...	1912	10.24	-5.08	-1.39	1.94	10.77	16.34	19.3	22.5	21.03	17.71	12.25	5.98	1.25
US	[[10.19, 11.11, 8...	1913	11.15	0.01	-1.47	4.18	10.88	15.69	20.46	22.94	22.93	17.75	11.42	7.43	1.48
US	[[7.22, 8.63, 6.9...	1914	11.1	1.19	-1.92	4.56	10.53	16.54	21.09	23.29	22.1	17.99	13.36	6.39	-2.53
US	[[4.95, 6.92, 4.3...	1915	10.94	-1.85	2.25	2.76	12.92	14.81	19.06	21.63	20.88	18.35	13.12	6.45	0.46
US	[[10.03, 12.04, 1...	1916	10.47	-1.61	-0.19	4.66	10.11	15.53	18.79	23.56	22.27	17.51	11.48	4.98	-1.61
US	[[9.09, 10.7, 14....	1917	9.8	-1.9	-1.69	3.78	9.3	12.93	19.26	23.19	21.43	17.42	9.73	5.91	-2.36
US	[[7.37, 3.26, 2.5...	1918	10.89	-5.33	0.49	7.25	9.35	16.49	21.09	22.09	22.56	16.18	13.35	5.16	1.85
US	[[5.14, 6.31, 7.5...	1919	10.85	0.01	0.09	5.04	10.43	15.38	20.91	23.34	21.83	18.6	11.96	4.29	-2.16

Consolidated data frame showing the individual month columns containing the monthly averages for each year.

By iterating through the Pandas data frame for the range of years specified and collecting monthly temperature changes, averaging those to produce an average monthly temperature change, then taking the mean of the average monthly temperature changes, the average yearly temperature change for the specified country is produced. This value was then multiplied by the number of years in the range of years specified to produce the average temperature change over the given range of years.

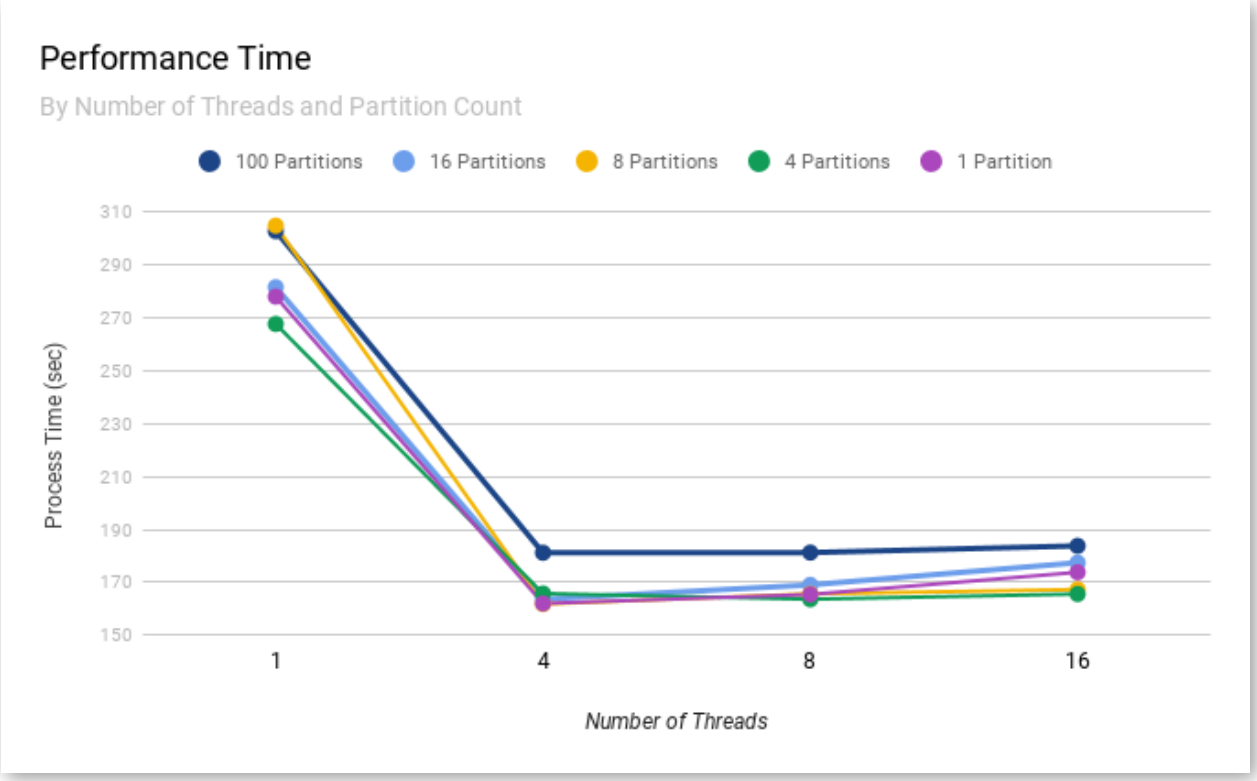
It is important to note that due to the relatively inconsistent and sometimes incomplete nature of the data source, we took a few extra steps to ensure as accurate a result as possible. We checked the monthly change before producing an average yearly temperature to account for years in which some months have no measurements. We also ignored skipped pairs of years in which one of the years had no or invalid measurements. Higher complexity data can lead to more false discoveries and results, and we attempted to accommodate for this.

To present the results, the reduced data frame, average yearly temperature increase and average temperature change over the given range of years was printed to the console. Additionally, PyPlot was used to visualize the results graphically, allowing easier trend discernment. These outputs can be seen in the “Results” section.

This program was tested on a cluster consisting of four Raspberry Pis, all running Raspbian, as well as multiple computers with differing specs running macOS Catalina. The primary goal of the tests was to determine the effect partition count and thread count have on speed of parallel computing.

RESULTS

The speed was tested with a varying number of threads (x-axis) and partition counts (various lines), which delineate the number of tasks the program can be split into.



The partition count seemed to have a minimal impact on performance at 1-16 partitions, but there was an increase in process time at 100 partitions. The increase at 100 partitions is presumably because of overhead involved in creating the partitions, and no way to take advantage of having the data partitioned that way. The number of threads, however, did have a significant impact on performance. One thread, regardless of partition count, performed significantly slower; this is likely due to the single thread missing out on the benefits of parallel computing experienced by multiple threads. *Holistically, 4 threads produced the best performance, with 8 partitions on 4 threads being the absolute best, and 1 partition on 4 threads performing well too.*

As expected, the speed on macOS laptops was much faster than on the Raspberry Pi Cluster (under 30 sec. compared to > 300 sec.); there were also no issues encountered with limited RAM on these machines as there were on the Raspberry Pis.

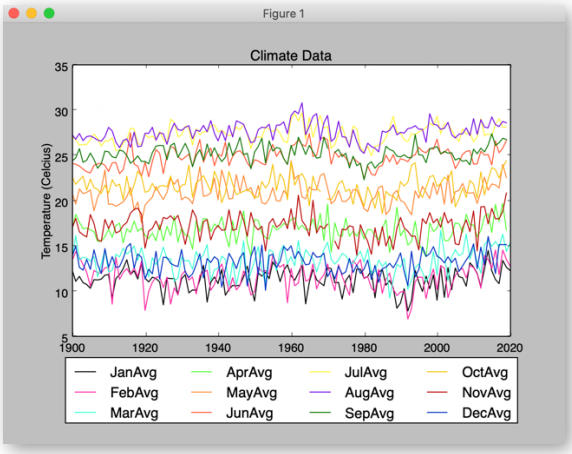
In addition to the results on computing performance, the results of the data computations are worth noting. Some countries, including the United States and Canada, appeared to be getting quite a bit *colder*, in contrast to the general scientific consensus of climate warming over time. After considering what could be causing this disparity in data results and known fact, we assumed it must be due to changes in what stations were taking measurements, and we assume that as time went over and more interest went into researching the climate, more measuring stations were established in colder areas.

This was not something initially anticipated when filtering the data, and to try and see if this was the case, we specifically analyzed parts of the data that looked more reliable. By selecting small island nations with consistent data input and a small number of measuring stations, the program returned data more similar to what would be expected. For example: Cyprus, a small island off the coast of Greece, showed a 1.7 degree C increase over the last 120 years, which is close to many scientific estimates of global temperature increase over the same period. The US from 2000-2020 showed an increase of .44 C, which is close to other estimates of temperature increases in that timeframe, and quite different from the 120-year apparent decrease in temperatures.

```
('You indicated country code ', 'CY')
No start and end years specified, using 1900-2019 as the time frame.
+-------+--------------------+----+-----+------+------+------+------+------+------+------+------+------+------+------+
|country|              months|year|  Avg|JanAvg|FebAvg|MarAvg|AprAvg|MayAvg|JunAvg|JulAvg|AugAvg|SepAvg|OctAvg|NovAvg|DecAvg|
+-------+--------------------+----+-----+------+------+------+------+------+------+------+------+------+------+------+------+
|     CY|[[12.88, 11.73, 1...|1900|19.17| 12.41| 13.44| 13.47| 16.91| 20.91| 24.15| 27.12| 27.35| 24.42| 22.71|  17.0| 13.88|
|     CY|[[9.93, 13.37, 10...|1901|20.41| 11.15| 14.41| 16.21| 18.54| 19.68| 24.09| 26.79| 26.84| 24.78| 21.86|  18.1| 15.04|
|     CY|[[11.55, 9.8, 10....|1902|19.53| 11.69| 13.81| 13.53| 17.68| 21.35| 23.87| 26.18| 27.47|  25.8| 22.34| 16.61| 12.69|
|     CY|[[11.8, 9.25, 10....|1903|18.93|  11.0| 12.99|  13.8| 16.95| 20.83| 23.54| 26.25| 26.69| 24.29| 22.75| 16.28| 12.38|
|     CY|[[10.6, 9.2, 9.96...|1904|18.88|  10.8| 12.15|  13.7| 17.02| 20.97| 23.24| 26.25| 27.13| 24.52| 21.23| 16.05| 13.56|
|     CY|[[10.1, 9.25, 12....|1905|18.89| 10.38|  10.6| 13.38| 16.83| 21.24| 23.13| 26.78|  27.2| 25.31| 22.43| 18.52| 12.46|
|     CY|[[10.33, 10.05, 1...|1906|19.15| 11.33| 12.35| 13.88| 16.64| 18.94| 23.95| 27.42| 27.01|  24.6| 21.52| 17.54| 14.63|
|     CY|[[11.18, 9.85, 9....|1907|18.54|  11.3|  12.1| 11.64| 15.55| 20.42| 24.12| 26.94| 26.74| 23.64| 21.29| 15.73| 13.01|
|     CY|[[11.33, 9.49, 10...|1908|18.54| 11.39| 12.04| 13.07| 15.59| 21.29|  24.2| 26.08| 26.63| 24.13| 20.84| 15.56| 12.01|
|     CY|[[11.83, 10.95, 1...|1909| 19.8| 11.83| 12.64| 14.58| 16.07| 23.43| 23.96| 26.66| 27.13| 25.52| 21.79| 17.99| 14.76|
|     CY|[[12.45, 10.05, 1...|1910|18.58| 11.89| 12.68| 12.17|  16.9| 19.66| 23.28| 26.63| 28.15| 25.45| 20.88| 16.97|  12.7|
|     CY|[[9.8, 8.09, 9.8,...|1911|18.47|  9.33|  8.63| 12.36|  16.0| 19.67| 24.23|  27.0| 27.75|  24.9| 22.37| 17.86| 13.88|
|     CY|[[11.54, 10.47, 1...|1912|19.47| 11.45| 13.82| 14.15| 17.31| 19.44| 24.27| 25.61| 26.52| 24.78| 21.66| 16.49| 12.68|
|     CY|[[11.65, 12.28, 1...|1913|19.48| 11.88| 11.37| 13.83| 17.57| 19.65| 23.04|  25.6| 25.91| 25.64| 21.88| 16.67| 12.77|
|     CY|[[12.7, 11.61, 13...|1914|18.99| 12.54| 12.12| 14.83| 15.39| 19.92| 23.49| 26.44| 26.69| 25.52| 20.16| 17.31| 13.67|
|     CY|[[13.5, 12.72, 13...|1915|20.04| 13.53| 12.55| 14.15| 16.95| 20.49| 25.22| 26.81| 26.88| 25.04| 22.75| 17.65| 14.92|
|     CY|[[12.04, 11.6, 10...|1916|19.92| 11.89|  11.9|  14.5| 16.56| 21.51| 27.54| 28.05| 25.94| 24.18| 20.62| 18.69| 15.32|
|     CY|[[12.25, 12.75, 1...|1917|19.24| 12.32| 12.65| 15.04| 17.81| 19.02|  24.1| 27.97| 27.62| 24.52|  21.8| 19.08| 12.58|
|     CY|[[11.55, 10.65, 1...|1918|19.19|  11.4| 11.68| 13.46| 16.66| 19.89| 23.86| 26.35| 26.01|  25.0| 23.33| 19.23| 14.15|
|     CY|[[13.1, 13.65, 12...|1919| 19.2| 13.13| 13.36| 14.87| 17.01| 17.63| 22.72| 26.24| 26.08| 24.39| 22.74| 18.57| 13.82|
+-------+--------------------+----+-----+------+------+------+------+------+------+------+------+------+------+------+------+
only showing top 20 rows

===== COUNTRY IS: CY =====
Total average yearly change in temperature from 1900 to 2019: 0.0144125427949 degrees C
Total average change in temperature from 1900 to 2019: 1.71509259259 degrees C
===== ===== ===== ===== ==
```

Data frame and summary of temperature change for Cyprus, 1900-2020.



PyPlot for Cyprus, 1900-2020.

CONCLUSION:

Increasing parallelization in processing generally helped, and the program was always faster running on more than a single thread, usually significantly faster. However, it is not always the case that more threads are better, because using eight and sixteen threads was consistently slower than using only four, albeit not by much. Once exceeding four threads, the overhead and idle time caused by I/O waiting outweighed the benefit from the extra threads, leaving four threads as optimal. The dataset was relatively large, but not massive, at approximately 100 MB in size, and still saw substantial benefits from computing on a cluster.

Per the results, we determined that parallel computing is suitable for processing large datasets where the benefit from parallel computing exceeds the costs from I/O and overhead of creating threads. If the machines on a cluster have a small amount of memory, and the program being ran is designed around these limitations, there is potential to see benefits in parallel computing.

Parallelism and cluster computing are powerful tools for computing on large datasets, but it is important to actually consider the task at hand, as well as which tools and techniques are optimal. It is not necessarily the case that more threads or more machines will result in a faster program, because overhead and I/O must be considered.

REFERENCES

[1] Fei Hu, Chaowei Yang, John Schnase, Daniel Duffy, Mengchao Xu, Michael Bowen, Tsengdar Lee, Weiwei Song, “ClimateSpark: An in-memory distributed computing framework for big climate data analytics,” *Computers & Geosciences Vol 115, June 2018, Pages 154-166*

[2] James H. Faghmous and Vipin Kumar.Big Data.Sep 2014.155-163. http://doi.org/big.2014.0026

[3] James Ford, Simon Tilleard, Lea Berrang-Ford, Malcolm Araos, Robbert Besbroek, Alexandra C. Lesnikowski, Graham K. MacDonald, Angel Hsu, Chen Chen, and Livia Bizikova, “Opinion: Big data has big potential for applications to climate change adaptation,” *Proceedings of the National Academy of Sciences of the United States of America*, September 27, 2016

[4] G. Jung, N. Gnanasambandam and T. Mukherjee, "Synchronous Parallel Processing of Big-Data Analytics Services to Optimize Performance in Federated Clouds," *2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, 2012, pp. 811-818.

[5] James Shanahan, Laing Dai, “Large Scale Distributed Data Science using Apache Spark,” *KDD ‘15: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*