APPROVAL SHEET

Title of Thesis: Efficient Scientific Big Data Aggregation through Parallelization and Subsampling

Name of Candidate: Savio Sebastian Kay
Master of Science in Information Systems, 2019

Thesis and Abstract Approved:

Dr. Jianwu Wang
Assistant Professor
Department of Information Systems
University of Maryland, Baltimore County

Date Approved: 07/26/2019

ABSTRACT

Title of Document:	EFFICIENT SCIENTIFIC BIG DATA AGGREGATION THROUGH PARALLELIZATION AND SUBSAMPLING
	Savio Sebastian Kay Master of Science, Information Systems, 2019
Directed By:	Dr. Jianwu Wang Assistant Professor Department of Information Systems

In the various scientific research study, experiments related to atmospheric physics and satellite data administration, processing and manipulation does take a considerable amount of time and resources depending on the size of the project. Due to the tremendous amount of data existing even in an essential use case, computing information does take a longer time. It is in the cause of multiple variables included in the substantial scientific dataset sizes of the Satellite specific files. One of the methods scientific researcher and developers' approach is to use more resources to manage the significant data ingestion and manipulation along with process parallelization like file-level parallelization and or day-level parallelization. It drastically reduces the time taken to process data. However, the concept of subsampling is known to diminish the period to a shorter span, which is suitable for a

lot of scientific study and experiments. In this thesis, the procedure of subsampling has tested and proposed to be an approach to decrease processing time radically. Experimental results show the Xarray python package; a modern python framework provides enough support to process large volumes of data in a shorter period, which is suitable for the scientific research study. We process One Month of Satellite data which constitutes to be 8928 HDF files with the size of about 1.154TB (Terabytes) of information. It includes 8928 HDF files of MYD03 (357.23GB) and 8928 HDF files of MYD06_L2 (797.71GB) MODIS satellite datasets. We evaluate the cloud property variable by aggregating Level 2 data to Level 3 format, and we achieve this via two primary approaches of subsampling and parallel processing. Our research and experiments show along with parallel computing on multiple compute nodes through XArray & Dask; subsampling technique can reduce system execution time dramatically with little to no data loss in the final computed information. The code for the research and study can be found over at the GitHub account of 'saviokay' with the repository name 'masters-thesis', it can be accessed via the link: **https://github.com/saviokay/masters-thesis .**

EFFICIENT SCIENTIFIC BIG DATA AGGREGATION THROUGH PARALLELIZATION AND SUBSAMPLING

by

Savio Sebastian Kay

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfillment
of the requirements for The Degree of
Master of Science in
Information Systems
2019

Advisory Committee:
Dr. Jianwu Wang, Chair/Advisor
Dr. Sisi Duan, Co-Chairperson
Dr. Zhibo Zhang

© Copyright by
Savio Sebastian Kay
2019

Dedication

I dedicate my dissertation work to my family and friends. A special feeling of gratitude to my loving parents, Sebastian Joseph Kay and Sheela Sebastian Kay whose constant words of encouragement and push of tenacity ring in my ears. My elder sister Sharon Ajay Matthew, who inspired me to pursue my Master Thesis and Research. I also dedicate this dissertation to my friends who have supported me throughout the process. I will always appreciate all they have done for me throughout the entire master program

Acknowledgement

I would like to deeply thank Prof. Jianwu Wang for not only introducing to the intriguing world of Big Data but to also be a being a constant guide in my research and pursuit to finding definitive results. His advice, guidance and especially persistent patience has helped me in completing this work on time. I would also like to mention Prof. George Karabatis in motivating me to pursue thesis and research completing my master's degree. The work is supported by a NASA CMAC project: Efficient and Flexible Aggregation and Distribution of MODIS Atmospheric Products based on Climate Analytics-as-a-Service Framework (grant number: 80NSSC18K0796 and 17-CMAC17-0004).

The hardware used in the study is from UMBC High Performance Computing Facility (HPCF) which is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258, CNS–1228778, and OAC–1726023) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See hpcf.umbc.edu for more information on HPCF and the projects using its resources.

I would also like to thank the committee members Prof. Zhibo Zhang, Prof. Sisi Duan and Prof. Jianwu Wang for being on the examination committee and for providing invaluable feedback. I am highly grateful to Deepak P, Hua Song, Chamara Raja and Redwan Walid for their crucial and immense contribution in developing the framework for processing level 3 data aggregation. Special thanks to my friends and colleagues in the Data Informatics Lab at UMBC for their tips and suggestions on the research ideas.

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

In the current scientific research development, computation, and processing, a large set of data has always been time-consuming and resource hungry. In the past, equating such large dataset for information took considerable period partly because the process conducted serially. Each file would be processed and equated one at a time to compute the final result. With the recent development, we can perform the same task in a parallel fashion with frameworks like Apache Spark, Python Dask, Hadoop, to name a few. However, along with the recent development of parallelization techniques, the larger datasets and research project is, the longer the overall execution period. New techniques are always welcomed to reduce processing time and produce the same result to save resources and time.

One of the techniques proposed is known as subsampling. In analytical chemistry, the concept of subsampling refers to taking a significantly smaller subset of the original dataset to represent its final values results. If the results from the subsampled dataset have a negligible difference between the original dataset, does it have any benefit to the procedure. There is a various way to subsample the dataset with a start-stop-step method, wherein the step represents the valid subsampling number. With the prediction of data loss, the resources required for processing such a sample would take a considerably low number of resource and hence even lower period. We can employ the subsampling along with data parallelization to improve the time efficiency of the system.

1.1 Significance of The Problem:

Some of the reasons to employ such techniques is because of a lot of scientific study and research use cases cannot render for a more extended period and may require quicker computation. Use cases such as Weather prediction, stock market prediction, and many more need computation at a much quicker pace with computation of considerably larger datasets. Xarray is an open-source python framework which processes heterogenous N-dimensionality datasets in a short period.

Some of the notable features:

Speed: Ingestion of Big and more massive Satellite dataset executes exponentially faster than its counterparts Xarray uses the concept and principles followed by Dask python library. Tasks working in parallel mode executes substantially faster than tasks working in serial mode. It uses dynamic task scheduling similar to the concepts followed in Airflow, Luigi, Celery, or Make but better enhance the interactive computational workload. Since it enables distributed computing in pure Python, it extends native support in speed and performance. Also, Dask is known to operate with very little overhead, low latency, and minimal serialization required for quick numerical algorithms.

Ease of Use: The XArray and Dask framework employs inspirations from the very familiar NumPy and Pandas Dataframe objects. Due to this feature, deploying functions and methods

from the framework is much more comfortable and requires fewer changes to the legacy code. Dask provides rapid feedback and diagnostics to aid understanding since the design is for interactive computing.

Compatible Everywhere: Dask runs on several platforms, such as HPC (High-Performance Computing) Environment, Python Interactive Mode, Python Standalone, and even in Cloud Environment. Dask runs resiliently on 1000 cores cluster or even a trivial configuration of a standalone laptop with a lower specification. In the study below, we run our system through High-performance computing provided by UMBC and in Binder cloud environment.

Combining the features of Xarray And Dask With the Subsampling Techniques, one can build a system that processes in real-time a large volume of the dataset and perform the computation to discover the cloud property and other various attributes of the atmospheric science research in the project.

1.2 Summary of The Approach:

In our approach, we have realized a system that uses XArray framework as the data structure helping in ingesting and manipulating the dataset with ease and the Dask framework to parallelize task to the workers through a scheduler. The scheduler maintains, manage the workers in the cluster, whether it be a local cluster of workers or a cloud-based cluster of workers.

We employ the subsampling technique to reduce the overall time for data ingestion, data manipulation, and overall data processing for the system. We use subsampling of variable instance to find the efficient instant. Exploring subsampling of 2 with a start-step-stop principle, wherein we would start with an entity record and take a step of 2 and register the entity record for further computation. It drastically deteriorates the period for the execution of the entire system. To efficiently process the large volume of data, we use the Dask python framework to process the incoming data in the real-time and compute the cloud property of the speculated date and time.

We use the HDFS (Hadoop File System) for data ingestion and even fitting the final output since the NASA EOS group supports and maintains the HDF File format along with the HDF Group for advancement in scientific research study and research.

1.3 Contribution of the Thesis:

This research study contains some unique contributions. They are summarized below:

- Creation of monthly cloud fraction mean algorithm code in Xarray pydata module.
- Calculation of the monthly cloud fraction mean using the subsampling technique for faster computation.
- Creation of the method to remote terminal use jupyterLab/jupyterNotebook with HPCF (UMBC) resources via ssh tunneling, port forwarding, and slurm jobs.
- Calculation of the data loss percentage of various subsampled models with the original dataset.
- Calculation of total execution time taken by the system to compute cloud property variable through various environments.

Chapter 2

Background

This is an introductory chapter that provides additional context for the findings and results discussed in the following chapters, including some essential notations. It begins with a brief review of our general approach and context for the workings implemented in this thesis. We go through the goals of data aggregation and present the main approach of subsampling used in this thesis.

2.1 HDF Standard & Files:

In today's world of a data-heavy centric medium where information is abundantly available and retrievable, one aspect which has not changed much is the data compression principle and the technologies associated with it of providing complete information in a faster and efficient way. The National Center for Supercomputing Applications while realizing this goal and principle with satellite information, decided to develop the HDF file format [1]. The NCSA gains enormous support from the HDF group, a non-profit corporation who have made it a goal to continue the development of HDF5 and related technologies. The goal was to create a high-performance data library and file format meant to process and store heterogeneous data efficiently.

HDF by itself supports n-dimensional dataset, and each element themselves can be complex objects. Hence it is widely used in aerospace, automotive, electronic instruments, financial services, government agency toward defense and national security and even medical and biotech industries. Scientific fields like physics, genomics, astronomy, computational fluid dynamics, and earth science reap the benefits of the file standard and file format.HDF file format enables to maintain data metadata, streamlining information lifecycles, and pipelines.

Projects dealing with massive datasets and attributes will benefit from the aspect of metadata along with the data which can be crucial at times [2]. Also, there is no limit or boundary in sizing the dataset or data objects in the group which promotes extensibility for big data. Additional features include cross-platform compatibility with several platforms, parallel systems, and high-level API with C++, Fortran 90, Java, and Python, to name a few interfaces it is used widely. HDF File format is known to possess high-performance input and output amidst an in-depth collection of unified performance features that permits for access time optimization.

2.1.1 HDF4:

Out of the array of formats supported by the HDF group, HDF4 is one of the older versions of them. It is known to support a wind range of data models, multidimensional arrays, raster images, and including tables[3]. Every single one of them defines a specific aggregate data type and provide an application programming interface for writing, reading and managing files, data, and metadata. A newer version of data models can be created by the HDF developer and or users. It is also known to be self-describing in nature, which means it allows the application to interpret and understand the structure and contents of the file without external information. Each HDF file has groups and or individual objects which can harbor a mix of related objects which can be accessed by the HDF developer or user.

Bearing in mind the advantages of the old format, it does possess many limitations. With a very unclear object model, continuous improvement and support can be tedious and difficult. The application program interfaces are rather complex with each interface style like tables and images, to name a few. One of the most significant limitations with HDF4 format was the use of 32-bit signed integer for limiting the files to a maximum of 2GB[4]. It renders its unusable in many modern scientific use cases and application.

.

2.1.2 HDF5:

With the limitation in the previous format, the newer format of HDF5 was designed to address and also anticipate future scientific requirements, use cases, and systems. The newer version developed with an updated structure with two types of objects, namely

- Datasets -- Multidimensional array of the homogenous type.
- Groups -- Container structures which can hold datasets.

The limitation in the previous format, the newer format of HDF5 was designed to address and also anticipate future scientific requirements, use cases, and systems [4]. The newer version developed with an updated structure with two types of objects, namely It is a genuinely hierarchical file system-like data format — metadata and attributes stored in the form of user-defined groups and datasets. HDF5 can be located using POSIX-like syntax for resources location -- /path/to/resource. Including HDF4. Like the predecessor, HDF5 possess complex application programming interface for writing, reading and managing data and metadata. The application programming interface was oriented concerning datasets, groups, attributes, and property list. One of the accessible scientific file formats used for various use cases is netCDF version

4, which is inspired by the framework of HDF5. HDF5 works well with stock prices series, 3D meteorological data and network networking data, in general, time-series data since it uses B-trees to the index table object. The data gets fitted in a straightforward array, and this ensures readability of data in a much faster rate than reasonable SQL database access [5]. The HDF5 storage mechanism can be simpler and faster than the star schema found in SQL.

2.2 XArray Architecture:

NumPy and Pandas python libraries stay widely used for data science that provides an array of use cases for data scientists and researchers worldwide. NumPy possesses powerful N-dimensional array objects with sophisticated functions for data manipulation and broadcasting, including tools for integration with older and much more convenient languages to data scientists like Fortran and C++. Simplified operations include linear algebra, Fourier transform, and random number and array generation capabilities for compelling use cases. Other than the scientific uses, NumPy is also known for the efficient multi-dimensional container of generic and user-defined data. In the case of Pandas, it is known to be a high-performance data structure popularly employed by data scientists for data analysis and research. One of the primary limitations of the python library, which is a deal breaker for our research study is the factor of having an equal number of entries columns and rows for the

following data structure. It possesses as a limitation to us since we do not have equal length of columns and rows in our satellite datasets [6]. However, we do need a lot of the functionalities of

NumPy and Pandas libraries with the flexibility of ingesting and manipulating variable dataset shapes efficiently.

Xarray does possess most of the properties and attributes of NumPy and Pandas python libraries along with the flexibility of having unusual dataset shapes and sizes. XArray, a derivative of pandas and NumPy libraries, maintains control for reading, writing, and manipulating the scientific Dataset to provide relevant results and research. The developers and high-level contributors to the specific library formulated the library to further scientific research with Satellite Dataset and Information.[7]

2.3 Subsampling:

Subsampling in the analytical chemistry sense is the procedure by which a small, representative sample from the whole larger sample [8]. The smaller sample is a subset of the larger dataset. The sample can be fewer record entities of the entire record set. Concerning scientific research, samples taken from the entire dataset followed the start-step-stop theory wherein we have a start with a record, take a step from that record and then stop at the above fourth record for processing. The explanation is

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 4

step. = 4

start = 1

stop = 100

The resulting array is a subset sample of the main array. We hypothesize in our evaluation, the data loss found in compared to the original dataset, is negligible and can be implemented with the subset. The running theory is representative sample should have comparatively low and maybe negligible systematic bias.

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 3

step. = 3

start = 1

stop = 100

The resulting array similar to the above example is a subset sample of the main array.

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 2

step. = 2

start = 1

stop = 100

The resulting array similar to the above example is a subset sample of the main array.

2.4 Taki Server: JupyterLab:

The steps outlined are for deploying a remote Jupyter Notebook session on the Taki Server to the local machine through SSH Tunneling and Port Forwarding.

For the current version and implementation of Jupyter Lab/Notebook on Taki/Maya Server can be deployed through a SLURM job with predefined Node configurations.

The Cluster configuration, down to the single node, can be modified by the SLURM file created for this intention. More details on modifying the Cluster Configuration on Taki Server view the Taki HPCF Documentation Page [9].

The SLURM Job File

The SLURM job file is at the folder location jianwu_common/MODIS_Aggregation/.

Navigate to the location and look for the file slurm_jupyter.sbatch

```
#!/bin/bash
## slurm_jupyter.sbatch
## Created Fri Apr 12 15:17:36 2019
## Author: saviokay

#SBATCH --job-name=tunnel
#SBATCH --partition=batch
#SBATCH --qos=medium+
#SBATCH --mem=16000
#SBATCH --output=jupyter-log-%J.txt
#SBATCH --nodes=4

XDG_RUNTIME_DIR=""
ipnport=$(shuf -i8000-9999 -n1)
ipnip=$(hostname -i)
## print tunneling instructions to jupyter-log-{jobid}.txt
echo -e "
Copy/Paste this in the local terminal to ssh tunnel with
remote
-------------
ssh -N -L $ipnport:$ipnip:$ipnport user@host
-------------
Then open a browser on the local machine to the following
address
```

```
------------------------------
localhost:$ipnport  (prefix w/ https:// if using password)
------------------------------
    "
## start an ipcluster instance and launch jupyter server
jupyter-notebook --no-browser --port=$ipnport --ip=$ipnip
```

Make sure to have appropriate file right permission to perform the above task.

Submit the slurm file as a job to the Taki server with command

```
sbatch slurm_jupyter. sbatch
```

The Jupyter-Log File

Once the job is submitted, check if the deployment is performed successfully with these steps:

Check the job status with squeue -u <username>

Check for the file jupyter-log-******.txt has been created the same directory.

Note that upon each successful deployment, a new jupyter-log-******.txt file. It has further instructions.

To view the latest file created,

```
savio1@taki-usr1 in MODIS_Aggregation : ls -la | grep jupyter

-rw-rw----  1 savio1   pi_jianwu     2744 Apr 18 00:33
#jupyter-log-331928.txt#
-rw-rw----  1 savio1   pi_jianwu     1576 May 16 21:52
jupyter-log-817138.txt
-rw-rw----  1 savio1   pi_jianwu     1573 May 16 21:52
jupyter-log-826118.txt
-rw-rw----  1 savio1   pi_jianwu     1576 May 16 22:07
jupyter-log-826299.txt
drwxr-xr-x  2 savio1   pi_jianwu       20 May 16 09:24
jupyter_txt
```

Open the latest jupyter-log-******.txt file based on the timestamp and follow the rest of the instruction in it.

```
For reference:
savio1@taki-usr1 in MODIS_Aggregation: cat jupyter-log-
826299.txt

    Copy/Paste this in the local terminal to ssh tunnel with
remote
-------------------------------------------------
ssh -N -L 8600:10.2.1.234:8600 user@host
-------------------------------------------------

Then open a browser on the local machine to the following
address

-----------------------------------------------
    localhost:8600  (prefix w/ https:// if using password)
-----------------------------------------------

[I 22:06:51.596 NotebookApp] JupyterLab extension loaded from
/umbc/xfs1/jianwu/common/anaconda3/lib/python3.7/site-
packages/jupyterlab
[I 22:06:51.599 NotebookApp] The Jupyter Notebook is running
at:
[I 22:06:51.599 NotebookApp]
http://10.2.1.234:8600/?token=02b5599708cf234994a8f24b23873e5
d7f7d51fb8c2c61e3
[I 22:06:51.599 NotebookApp] Use Control-C to stop this
server and shut down all kernels (twice to skip
confirmation).
[C 22:06:51.624 NotebookApp]

To access the notebook,
file:///home/savio1/.local/share/jupyter/runtime/nbserver-
27979-open.html
Or copy and paste:

http://10.2.1.234:8600/?token=02b5599708cf234994a8f24b23873e5
```

```
d7f7d51fb8c2c61e3
```

After this open a new terminal window and execute the following command:

```
ssh -N -L 8600:10.2.1.234:8600 user@host

For example
ssh -N -L 8600:10.2.1.234:8600 savio1@taki.rs.umbc.edu
```

Once the command is running, a prompt will ask the Taki password. (It is usually the same as your UMBC password). Enter the following and let it run in the background.

```
ssh -N -L 8600:10.2.1.234:8600 savio1@taki.rs.umbc.edu
WARNING: UNAUTHORIZED ACCESS to this computer violates
Criminal
savio1@taki.rs.umbc.edu's password:
```

NOTE: Once the password is entered, the terminal window will not output anything. However, **let it run in the background** as it is performing the SSH tunneling from the Taki server to the system with the command mentioned above. If the terminal window is closed, the connection will also be terminated immediately, and any saves and checkpoints will be invalid.

Once all steps are performed, access the Notebook through a web interface by entering this on the Web Browser:

```
http://localhost:8600
//Note this information is retrieved from the jupyter-log-
******.txt  file created earlier on taki.
```

For the current example, the web address specified is

```
http://10.2.1.234:8600/?token=02b5599708cf234994a8f24b23873e5
d7f7d51fb8c2c61e3
http://localhost:8600/?token=02b5599708cf234994a8f24b23873e5d
7f7d51fb8c2c61e3
//However we need to change 10.2.1.234 here with localhost.
If any prompts for password or token is requested, use the
token specified in the URL above.
```

Moreover, if the steps are followed accurately up till now, a functioning Jupyter Notebook should be deployed from the Taki server on the local machine.

Let us go a step further and get Taki Server Folder access and additional functionality, use Jupyter Lab. To achieve that change the URL from http://localhost:8600/?tree to

http://localhost:8600/?lab

NOTE:

Do remember to close all kernels and tabs.

Once completed, make sure to cancel all running jobs on Taki to avoid getting flagged by the administrator for User node Job Deployment Abuse.

Use `scancel <job_id>` to cancel a job on Taki Server.

Chapter 3

Methodology

In this chapter we will be reviewing the multiple aspects involved in the process of aggregating level 2 to level 3 satellite data with the help of UMBC High Performance Computing Taki Cluster[10]. The Cluster possess a behemoth of 324 nodes, 38 Graphical Processing Units, 40 Intel Phi Coprocessor with over 15TB of main memory which is appropriately distributed into four main categories of usage.[11]

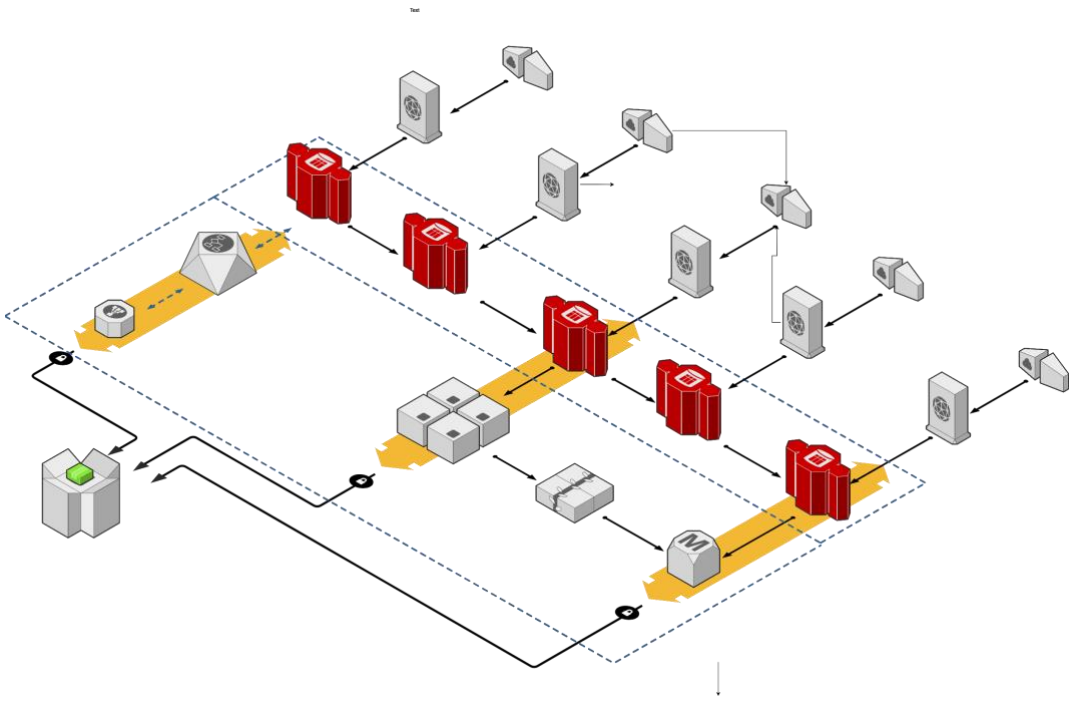


Figure 3.1 : Architecture Of Taki Cluster

The image provides a brief overview of the Taki Cluster architecture. The following gives us a basic understanding of processes and job transactions occurring during the

functioning of the server. The Architecture consists of terminals which are seeking for the resources from the server and are accessing either via the internal network of the server or are accessing it via remotely via SSH tunneling. Either of the cases, there are firewalls in place and specific categories of nodes accessible to hierarchical users of the server.

3.1 Overview of The Approach:

While the approach of the data aggregation from level 2 to level 3 can be obtained in several ways and with the use of various frameworks, we will be primarily looking to seek the output with the help of Xarray. The files are in HDF format, especially HDF4 format. NASA's Earth Observing System (EOS) maintains its HDF modification called HDF-EOS. As of now, NASA EOS group utilizes HDF4-EOS format for storage and management of satellite specific data. Recent developments support the HDF5-EOS format for the specific reading and transmitting of the instrument found in NASA's Aura Satellites [12]. Primarily we use the framework's function for reading all Satellite Scientific Datasets with additional arguments for data preprocessing and management. With the use of NumPy and pandas indexing, we extract the required variables and assign them to the appropriate user-defined variables.

Additional to these indexing, we also perform required bitwise computation to get the appropriate values [13]. The values are found to be compressed and assigned to the SDS (Scientific Dataset) to converse space and provide optimization. In the evaluation chapters, we

explore the effects of subsampling on the performance and the cheer data loss compared to without the subsampling. Accordingly, we assign the latitude and longitude indexes through the required data variables. Equating the total pixel available and the cloud pixel detected from the indexes thus created earlier.

With the total and cloud pixel calculated, we determine Cloud Fraction (CF) by dividing the cloud pixel array with the total pixel array. The resulting array is fit into Xarray Data Array data structure. With the assistance of the XArray saving libraries, we save the data structure into an HDF file. We plot the results with the help of the 'matplotlib' library to verify and validate the data. Further, we study and benchmark the effects of the subsampling with a comparison between the results with and without subsampling.

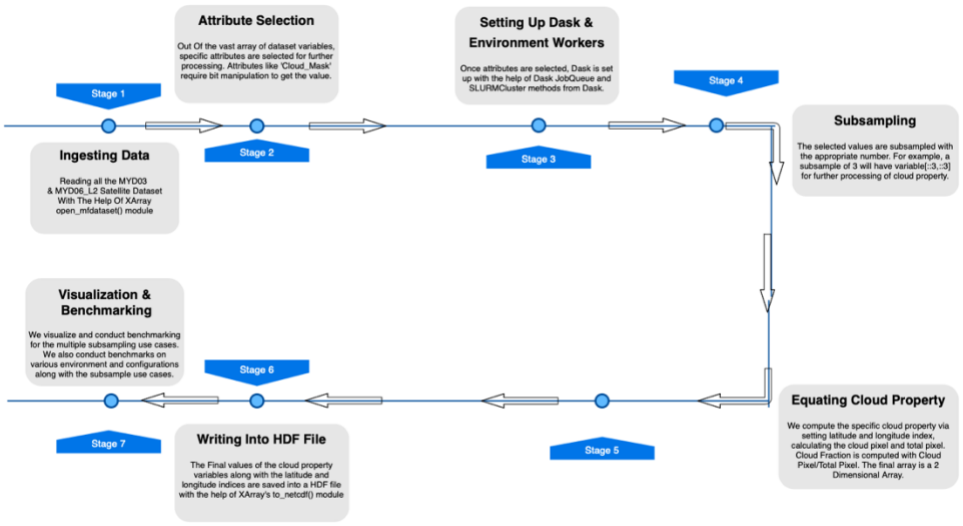


Figure 3.2: Procedure and Approach Overview

.

3.2 Dataset and Source Description:

The MODIS Satellite Atmosphere Imager instrument records data in the legacy MODIS standard products, including Aerosol (MYD04), Water Vapor (MYD05), Atmospheric Profile (MOD/MYD07), Cloud Mask (MOD/MYD35), Joint Atmosphere (ATML2), and Level-3 (MOD/MYD08) [13]. The products are in the continuous development of Terra and Aqua Mission. The Dataset in understanding is a set of Scientific Datasets (SDS) form compressed in 2040x1354 dataset shape. For the study, we pursue the MYD03 and MYD06_L2 datasets, which contain both geolocation information and cloud properties information. From the MYD03 Dataset; we extract the latitude and longitude information for each specific location for 1KM resolution (1x1KM). We extract the cloud properties 1KM resolution from the MYD06_L2 Dataset. The resolution of L2 cloud products is at 1x1 KM and 5x5 KM resolution for all of the variables of cloud properties. The filenames for the datasets follow standardized pattern and format. For the Terra MODIS Satellite, it follows the MOD06_L2.AYYYYDDD.HHMM.VVV.YYYYDDDHHMMSS.hdf and MYD06_L2.AYYYYDDD.HHMM.VVV.YYYYDDDHHMMSS.hdf for Aqua MODIS Satellite. [13]The format highlights the following standard:

```
MOD06/MYD06 = Earth Science Data Type name
L2 = Denotes a Level-2 product
A = Indicates the following date/time information is for the
acquisition (observation)
YYYYDDD = Acquisition year and day-of-year
HHMM = Acquisition Hour and minute start time
VVV = collection (e.g., '004' for Collection 4)
YYYYDDDHHMMSS = production data and time
HDF = denotes HDF file format.
```

The Time Zone followed is UTC times, not necessarily local and the most evident aspect is the MOD prefix expressly signifies Terra platform data granule and MYD for the Aqua platform data granule.

3.3 Cloud Property Variables & Bitwise Conversions:

The study explores the specific variable of Cloud Mask (CM) calculate the final Cloud Fraction (CF). The variable is from the MODO6_L2 dataset. However, certain variables are stored in the bit form and need conversion to obtain the actual value out of the dataset. The values in the dataset are stored in such a way to maintain smaller file sizes and compressions. Hence to obtain the actual value, the specific bitwise operation needs to be performed during data ingestion to evaluate the correct information.[13] For the current study, we choose the Cloud Mask 1x1 KM resolution variable stored as 'cloud_mask_1km.' We use the Xarray module of 'mf_opendataset ()' to ingest the dataset with the specific variable passed as an argument to extract it. The module has additional arguments to scan through and provide an extra level of granularity for filtration during data ingestion.

```
data = xr.open_mfdataset(M06[:],
parallel=True)['Cloud_Mask_1km'][:,:,:].values

data  = Variable Name
xr = XArray python library
open_mfdataset = XArray Dataset Read Module
M06[:] = File List With File Path and File Directory
['Cloud_Mask_1km'][:,:,0].values = Variable Selection.
```

Further to decode the actual value, we process the value with the following operation:

data_decoded = (np.array(data, dtype = "byte") & 0b00000110) >> 1

We use the AND operation in the "byte" version of ds06 compacted in a numpy array with the 0b00000110 with right shifting operation to the resulting array. Each Bit in the resulting entries represents particular bit value definitions.

Scientific Data Set (SDS): "Cloud_Mask_1KM"

Description: Cloud Mask QA Flags at 1x1 KM

Length: 2 bytes (16 bits)

Flag Name	Number Of Bits	Bit Values	Bit Value Definitions
Cloud Mask Cloudiness Flag	2	0 1 2 3	Confident Cloudy Probably Cloudy Probably Clear Confident Clear

3.4 Evaluating Cloud & Total Pixel:

For the Cloud fraction, cloud and total pixel array are calculated to equate the final result. We evaluate the cloud fraction by dividing the cloud pixel with the total pixel.

```
Cloud_Fraction Array = Cloud_Pixel Array/ Total_Pixel Array
```

To retrieve the cloud pixel array, we utilize the integer value converted latitude and longitude reshaped into the ingested dataset shape and dimension. We equated the total pixel based on the zip function with the latitude index and longitude index.

```
latitude_index = np.where(l_index > -1, l_index, 0)
longitude_index = np.where(ll_index > -1, ll_index, 0)
for i, j in zip(latitude_index , longitude_index ):
        total_pixel[i,j] += 1
```

After the total pixel, we tackle cloud pixel for which we again set the indexes and compute the respective latitude and longitude correlated to the cloud pixels with the zip function.

```
indicies = np.nonzero(data_decoded <= 0)
row_index = indicies[0]
column_index = indicies[1]
cloud_longitude = [longitude_index.reshape(data_decoded
.shape[0],data_decoded .shape[1])[i,j] for i, j in
zip(row_index, column_index)]
cloud_latitude = [latitude_index .reshape(data_decoded
.shape[0],data_decoded .shape[1])[i,j] for i, j in
zip(row_index, column_index)]

    for x, y in zip(cloud_latitude, cloud_longitude):
        cloud_pixel[int(x),int(y)] += 1
```

3.5 Equating Total Cloud Fraction:

With the final total pixel array and final cloud pixel array, we equate the total cloud fraction, referred to as CF in many instances. The division of cloud pixel array with the total pixel array results in the total cloud fraction.

For instance,

```
Cloud Fraction = Cloud Pixel Array / Total Pixel Array
```

The resulting cloud fraction is a two-dimensional array with zero/Nan values along with non-zero values. The resulting shape is 180x360 which represents the entire globe while plotting. Regardless of the time series selected for processing, the final cloud fraction array would be in the shape of 180x360 with corresponding latitude and longitude. All of the elements of the final cloud fraction array are of 1:1 ratio with the elements in the latitude and longitude values in MYD03 dataset. That suggests that values corresponding to a certain point or pixel will have the same value in the MYD03 dataset which possessed all of the geolocation data.

Data	Dataset Type	Dimension	Shape
Latitude	MYD03 SDS	2	2030 x 1354
Longitude	MYD03 SDS	2	2030 x 1354
Cloud Mask	MYD06_L2 SDS	3	2030 x 1354 x 2
Cloud Pixel	User Defined Array	2	180 x 360
Total Pixel	User Defined Array	2	180 x 360

Cloud Fraction	User Defined Array	2	180 x 360

The resulting array is equated with the 1x1 KM resolution values from the MYD03 SDS and the values may be varied if the 5x5 resolution values are employed. The final array is saved in an HDF file format with the help of to_netcdf () module from the XArray framework.

```
filename = xarray.DataArray(<cloud_fraction_variable>)
filename.to_netcdf("filepath/filename.hdf")
```

3.6 Plotting Cloud Fraction & Benchmarking Execution Time:

With the final file created with the xarray module, we can verify and validate the findings by benchmarking and plotting the HDF files. For this task, we use matplotlib python library. The library is used for data visualization and plotting various graphs for data analysis and research. The module of Pyplot in 'matplotlib' is used to create our plots for this research. We utilize by importing matplotlib. pyplot as plt. The visualization can be saved with module plt.savefig("<filepath/filename.png>").

```
plt.figure(figsize=(14,7))
plt.contourf(range(-180,180), range(-90,90),
cloud_fraction_variable, 100, cmap = "jet")
plt.xlabel("Longitude", fontsize = 14)
plt.ylabel("Latitude", fontsize = 14)
plt.title("<title_name>", fontsize = 16)
plt.colorbar()
plt.savefig("<filepath/filename.png>")
```

To benchmark the execution time, python's native time module is used to equate the

total time of execution. With one module instance created at the start of the code and one at the end of the code, the difference between the two made the total execution time. This gives a brief outlook of the duration of the code. Further benchmarking is performed in the evaluation and implementation section.

```
start_time = time.time()
end_time  = time.time()
print("Total Time Taken This Loop: ", end_time - start_time)
hours, rem = divmod(end_time-start_time, 3600)
minutes, seconds = divmod(rem, 60)
```

3.7 Subsampling:

Subsampling in the analytical chemistry sense is the procedure by which a small, representative sample from the whole larger sample. The smaller sample is a subset of the larger dataset. The sample can be fewer record entities of the entire record set. Concerning scientific research, samples taken from the entire dataset followed the start-step-stop theory wherein we have a start with a record, take a step from that record and then stop at the above fourth record for processing. The explanation is

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 4

step. = 4

start = 1

stop = 100

```
import numpy as np
np.arange(start = 1, stop = 100, step = 4).reshape((5,5))

>>array([[ 1,  5,  9, 13, 17],
       [21, 25, 29, 33, 37],
       [41, 45, 49, 53, 57],
       [61, 65, 69, 73, 77],
       [81, 85, 89, 93, 97]])
```

The resulting array is a subset sample of the main array. We hypothesize in our evaluation, the data loss found in compared to the original dataset, is negligible and

can be implemented with the subset. The running theory is representative sample should have comparatively low and maybe negligible systematic bias.

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 3

step. = 3

start = 1

stop = 100

```
import numpy as np
np.arange(start = 1, stop = 100, step = 3).reshape((3,11))

>>array([[ 1,  4,  7, ..., 25, ..., 31],
[34, 37, 40, 43, 46, ..., 61, 64],
[67, 70, 73, 76, ..., 94, 97]])
```

The resulting array similar to the above example is a subset sample of the main array.

Assuming a 10 x 10 2D matrix array with 100 record entities, with subsampling of 2

step. = 2

start = 1

stop = 100

```
import numpy as np
np.arange(start = 1, stop = 100, step = 2).reshape((2,25))

>>array([[ 1,  3,  5,  7, ..., 21, 23, 25, 27, 29, 31,33, 35, 37,
39, 41, 43, 45, 47, 49],
[51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79,
81,83, 85, 87, ..., 99]])
```

The resulting array similar to the above example is a subset sample of the main array.

Chapter 4

Implementation & Evaluation

The chapter mainly focuses on the implementation and evaluation of the prototype of the system that performs subsampling with the satellite dataset to reduce the time duration into a shorter span. To carry out the experimentation, we have used the MODIS satellite MYD03 and MYD06_L2 datasets provided by NASA's atmospheric and climate research website portal. The MYD06_L2 consists of parameters at a spatial resolution of 1 - KM (1 x 1 KM) or 5 - KM (5 x5 KM) [14]. Every file covers a five-minute time interval, and this results in 288 files for one day and 8,928 files for a month. Once the data gets ingested we create the subsamples for each use case and evaluate the overall performance of our system by running it in a various number of clusters and we found out that as we increase the subsampling and number of resources, the overall processing time for the system decreases.

4.1 Implementation:

The scientific dataset which we utilize in the research study is from NASA's MODAPPS Portal and is formatted in the HDF File format recognized by NASA's EOS group which manages and maintain the file format. For this evaluation, we will be tackling the cloud property of the cloud mask and cloud fraction from the vast array of variables, groups, and SDS (Scientific Datasets). We retrieve the geolocation from the MYD03 dataset and cloud property from MYD06_L2

dataset [14]. With the help of XArray python library, we ingest the data from both the dataset and create a Xarray data structure to further processing and manipulation.

Some of the attributes found in the dataset are described below:

Id	Name	Nb_Dim	Dim	Type	Nb_attributes
0	Latitude	2	[408, 270]	5	10
1	Longitude	2	[408, 270]	5	10
2	Scan_Start_Time	2	[408, 270]	6	10
3	Solar_Zenith	2	[408, 270]	22	10
4	Solar_Zenith_Day	2	[408, 270]	22	10
5	Solar_Zenith_Night	2	[408, 270]	22	10
6	Solar_Azimuth	2	[408, 270]	22	10
7	Solar_Azimuth_Day	2	[408, 270]	22	10
Id	**Name**	**Nb_Dim**	**Dim**	**Type**	**Nb_attributes**
8	Solar_Azimuth_Night	2	[408, 270]	22	10
9	Sensor_Zenith	2	[408, 270]	22	10
10	Sensor_Zenith_Day	2	[408, 270]	22	10
11	Sensor_Zenith_Night	2	[408, 270]	22	10
12	Sensor_Azimuth	2	[408, 270]	22	10
13	Sensor_Azimuth_Day	2	[408, 270]	22	10
14	Sensor_Azimuth_Night	2	[408, 270]	22	10
15	Brightness_Temperature	3	[7, 408, 270]	22	10
16	Surface_Temperature	2	[408, 270]	22	10
17	Surface_Pressure	2	[408, 270]	22	10

18	Cloud_Height_Method	2	[408, 270]	20	11
19	Cloud_Top_Height	2	[408, 270]	22	10
20	Cloud_Top_Height_Nadir	2	[408, 270]	22	10
21	Cloud_Top_Height_Nadir_Day	2	[408, 270]	22	10
22	Cloud_Top_Height_Nadir_Night	2	[408, 270]	22	10
23	Cloud_Top_Pressure	2	[408, 270]	22	10
24	Cloud_Top_Pressure_Nadir	2	[408, 270]	22	10
25	Cloud_Top_Pressure_Night	2	[408, 270]	22	10
26	Cloud_Top_Pressure_Nadir_Night	2	[408, 270]	22	10
27	Cloud_Top_Pressure_Day	2	[408, 270]	22	10
28	Cloud_Top_Pressure_Nadir_Day	2	[408, 270]	22	10
29	Cloud_Top_Temperature	2	[408, 270]	22	10
30	Cloud_Top_Temperature_Nadir	2	[408, 270]	22	10
31	Cloud_Top_Temperature_Night	2	[408, 270]	22	10
Id	**Name**	**Nb_Dim**	**Dim**	**Type**	**Nb_attributes**
32	Cloud_Top_Temperature_Nadir_Night	2	[408, 270]	22	10
33	Cloud_Top_Temperature_Day	2	[408, 270]	22	10

34	Cloud_Top_Temperature_Nadir_Day	2	[408, 270]	22	10
35	Tropopause_Height	2	[408, 270]	22	10
36	Cloud_Fraction	2	[408, 270]	20	10
37	Cloud_Fraction_Nadir	2	[408, 270]	20	10
38	Cloud_Fraction_Night	2	[408, 270]	20	10
39	Cloud_Fraction_Nadir_Night	2	[408, 270]	20	10
40	Cloud_Fraction_Day	2	[408, 270]	20	10
41	Cloud_Fraction_Nadir_Day	2	[408, 270]	20	10
42	Cloud_Effective_Emissivity	2	[408, 270]	20	10
43	Cloud_Effective_Emissivity_Nadir	2	[408, 270]	20	10
44	Cloud_Effective_Emissivity_Night	2	[408, 270]	20	10
45	Cloud_Effective_Emissivity_Nadir_Night	2	[408, 270]	20	10
46	Cloud_Effective_Emissivity_Day	2	[408, 270]	20	10
47	Cloud_Effective_Emissivity_Nadir_Day	2	[408, 270]	20	10
48	Cloud_Top_Pressure_Infrared	2	[408, 270]	22	10
49	Spectral_Cloud_Forcing	3	[5, 408, 270]	22	11
50	Cloud_Top_Pressure_From_Ratios	3	[5, 408, 270]	22	11
51	Radiance_Variance	2	[408, 270]	22	10
52	Cloud_Phase_Infrared	2	[408, 270]	20	11

53	Cloud_Phase_Infrared_Night	2	[408, 270]	20	11
54	Cloud_Phase_Infrared_Day	2	[408, 270]	20	11
Id	**Name**	**Nb_Dim**	**Dim**	**Type**	**Nb_attributes**
55	Cloud_Phase_Infrared_1km	2	[2040, 1354]	20	11
56	IRP_CTH_Consistency_Flag_1km	2	[2040, 1354]	20	11
57	os_top_flag_1km	2	[2040, 1354]	20	11
58	cloud_top_pressure_1km	2	[2040, 1354]	22	10
59	cloud_top_height_1km	2	[2040, 1354]	22	10
60	cloud_top_temperature_1km	2	[2040, 1354]	22	10
61	cloud_emissivity_1km	2	[2040, 1354]	20	10
62	cloud_top_method_1km	2	[2040, 1354]	20	11
63	surface_temperature_1km	2	[2040, 1354]	22	10
64	cloud_emiss11_1km	2	[2040, 1354]	22	10
65	cloud_emiss12_1km	2	[2040, 1354]	22	10
66	cloud_emiss13_1km	2	[2040, 1354]	22	10
67	cloud_emiss85_1km	2	[2040, 1354]	22	10
68	Cloud_Effective_Radius	2	[2040, 1354]	22	10
69	Cloud_Effective_Radius_PCL	2	[2040, 1354]	22	10
70	Cloud_Effective_Radius_16	2	[2040, 1354]	22	10

71	Cloud_Effective_Radius_16_PCL	2	[2040, 1354]	22	10
72	Cloud_Effective_Radius_37	2	[2040, 1354]	22	10
73	Cloud_Effective_Radius_37_PCL	2	[2040, 1354]	22	10
74	Cloud_Optical_Thickness	2	[2040, 1354]	22	10
75	Cloud_Optical_Thickness_PCL	2	[2040, 1354]	22	10
76	Cloud_Optical_Thickness_16	2	[2040, 1354]	22	10
77	Cloud_Optical_Thickness_16_PCL	2	[2040, 1354]	22	10
78	Cloud_Optical_Thickness_37	2	[2040, 1354]	22	10
Id	**Name**	**Nb_Dim**	**Dim**	**Type**	**Nb_attributes**
79	Cloud_Optical_Thickness_37_PCL	2	[2040, 1354]	22	10
80	Cloud_Effective_Radius_1621	2	[2040, 1354]	22	10
81	Cloud_Effective_Radius_1621_PCL	2	[2040, 1354]	22	10
82	Cloud_Optical_Thickness_1621	2	[2040, 1354]	22	10
83	Cloud_Optical_Thickness_1621_PCL	2	[2040, 1354]	22	10
84	Cloud_Water_Path	2	[2040, 1354]	22	10
85	Cloud_Water_Path_PCL	2	[2040, 1354]	22	10
86	Cloud_Water_Path_1621	2	[2040, 1354]	22	10

87	Cloud_Water_Path_1621_PCL	2	[2040, 1354]	22	10
88	Cloud_Water_Path_16	2	[2040, 1354]	22	10
89	Cloud_Water_Path_16_PCL	2	[2040, 1354]	22	10
90	Cloud_Water_Path_37	2	[2040, 1354]	22	10
91	Cloud_Water_Path_37_PCL	2	[2040, 1354]	22	10
92	Cloud_Effective_Radius_Uncertainty	2	[2040, 1354]	22	10
93	Cloud_Effective_Radius_Uncertainty_16	2	[2040, 1354]	22	10
94	Cloud_Effective_Radius_Uncertainty_37	2	[2040, 1354]	22	10
95	Cloud_Optical_Thickness_Uncertainty	2	[2040, 1354]	22	10
96	Cloud_Optical_Thickness_Uncertainty_16	2	[2040, 1354]	22	10
97	Cloud_Optical_Thickness_Uncertainty_37	2	[2040, 1354]	22	10
98	Cloud_Water_Path_Uncertainty	2	[2040, 1354]	22	10
Id	**Name**	**Nb_Dim**	**Dim**	**Type**	**Nb_attributes**
99	Cloud_Effective_Radius_Uncertainty_1621	2	[2040, 1354]	22	10
100	Cloud_Optical_Thickness_Uncertainty_1621	2	[2040, 1354]	22	10
101	Cloud_Water_Path_Uncertainty_1621	2	[2040, 1354]	22	10
102	Cloud_Water_Path_Uncertainty_16	2	[2040, 1354]	22	10

103	Cloud_Water_Path_Uncertainty_37	2	[2040, 1354]	22	10
104	Above_Cloud_Water_Vapor_094	2	[2040, 1354]	22	10
105	IRW_Low_Cloud_Temperature_From_COP	2	[2040, 1354]	22	10
106	Cloud_Phase_Optical_Properties	2	[2040, 1354]	20	11
107	Cloud_Multi_Layer_Flag	2	[2040, 1354]	20	11
108	Cirrus_Reflectance	2	[2040, 1354]	22	10
109	Cirrus_Reflectance_Flag	2	[2040, 1354]	20	11
110	Cloud_Mask_5km	3	[408, 270, 2]	20	11
111	Quality_Assurance_5km	3	[408, 270, 10]	20	11
112	Cloud_Mask_1km	3	[2040, 1354, 2]	20	11
113	Extinction_Efficiency_Ice	2	[12, 7]	5	11
114	Asymmetry_Parameter_Ice	2	[12, 7]	5	11
115	Single_Scatter_Albedo_Ice	2	[12, 7]	5	11
116	Extinction_Efficiency_Liq	2	[18, 7]	5	11
117	Asymmetry_Parameter_Liq	2	[18, 7]	5	11
118	Single_Scatter_Albedo_Liq	2	[18, 7]	5	11
119	Cloud_Mask_SPI	3	[2040, 1354, 2]	22	10
120	Retrieval_Failure_Metric	3	[2040, 1354, 3]	22	11

Id	Name	Nb_Dim	Dim	Type	Nb_attributes
121	Retrieval_Failure_Metric_16	3	[2040, 1354, 3]	22	11
122	Retrieval_Failure_Metric_37	3	[2040, 1354, 3]	22	11
123	Retrieval_Failure_Metric_1621	3	[2040, 1354, 3]	22	11
124	Atm_Corr_Refl	3	[2040, 1354, 6]	22	11
125	Quality_Assurance_1km	3	[2040, 1354, 9]	20	11
126	Statistics_1km_sds	1	17	5	4

Figure 4.2: Resulting Dataset, Groups And Variable With Sizes and Dimensions

Dataset/Array	Dimensional Size	Description
Proposed Cloud Mask After Subsampling Of 5	[2030 x1354] ~ [508 x 339]	The Resulting of having step of 5 with involving record entities for processing the cloud fraction. The resultant dataset is drastically smaller than the original dataset.

Figure 4.3: The Cloud Mask Array Pre And Post Subsampling of 5

Dataset/Array	Dimensional Size	Description
Proposed Cloud Mask After Subsampling Of 4	[2030 x1354] ~ [508 x 339]	The Resulting of having step of 4 with involving record entities for processing the cloud

		fraction. The resultant dataset is drastically smaller than the original dataset.

Figure 4.4: The Cloud Mask Array Pre And Post Subsampling of 4

Dataset/Array	Dimensional Size	Description
Proposed Cloud Mask After Subsampling Of 3	[2030 x1354] ~ [677 x 452]	The Resulting of having step of 3 with involving record entities for processing the cloud fraction. The resultant dataset is drastically smaller than the original dataset.

Figure 4.5: The Cloud Mask Array Pre And Post Subsampling of 3

Dataset/Array	Dimensional Size	Description
Proposed Cloud Mask After Subsampling Of 2	[2030 x1354] ~ [1015 x 677]	The Resulting of having step of 2 with involving record entities for processing the cloud fraction. The resultant dataset is drastically smaller than the original dataset.

Figure 4.6: The Cloud Mask Array Pre And Post Subsampling of 2

4.2 Evaluation:

The scientific dataset which we utilize in the research study is from NASA's MODAPPS Portal and is formatted in the HDF File format recognized by NASA's EOS group which manages and maintain the file format. For this evaluation, we will be tackling the cloud property of the cloud mask and cloud fraction from the vast array of variables, groups, and SDS(Scientific Datasets). We retrieve the geolocation from the MYD03 dataset and cloud property from MYD06_L2 dataset[15]. With the help of XArray python library, we ingest the data from both the dataset and create a xarray data structure to further processing and manipulation.

Some of the attributes found in the dataset are described below:

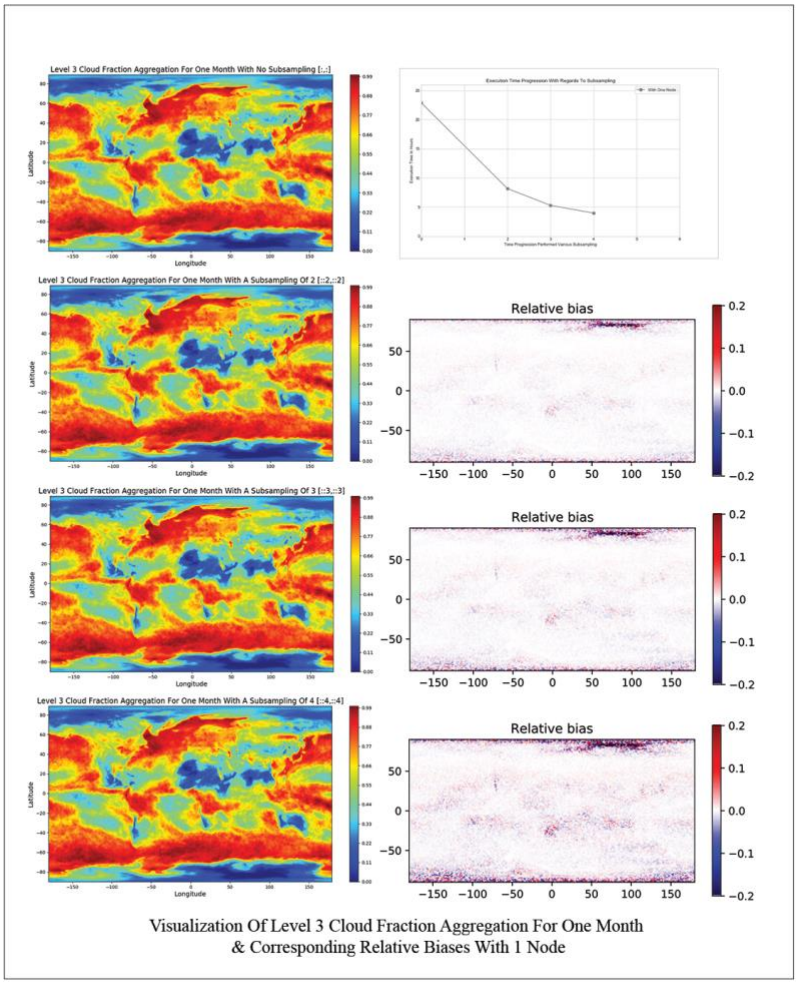
4.2.1 Subsampling Evaluation:

4.2.1.1 Evaluation Through Taki HPC Server:

Env.	Code File	SLURM File	Subsampling	# Of Nodes	Data Loss (%)	Execution Time
Taki	nosub1n_onemonth.py	nosub1n_onemonth.slurm	0	1	NA	82332.01 Sec (22.87 Hrs)
Taki	twosub1n_onemonth.py	twosub1n_onemonth.slurm	2	1	3.5370	29412.33 Sec (8.17 Hrs)
Taki	threesub1n_onemonth.py	threesub1n_onemonth.slurm	3	1	5.4871	19080.29 Sec (5.30 Hrs)
Taki	foursub1n_on	foursub1n_one	4	1	2.2154	14292.32

	emonth.py	month.slurm				Sec (3.97 Hrs)

Figure 4.11: System Execution Time And Data Loss Evaluation Table For Taki HPC With 1 Node



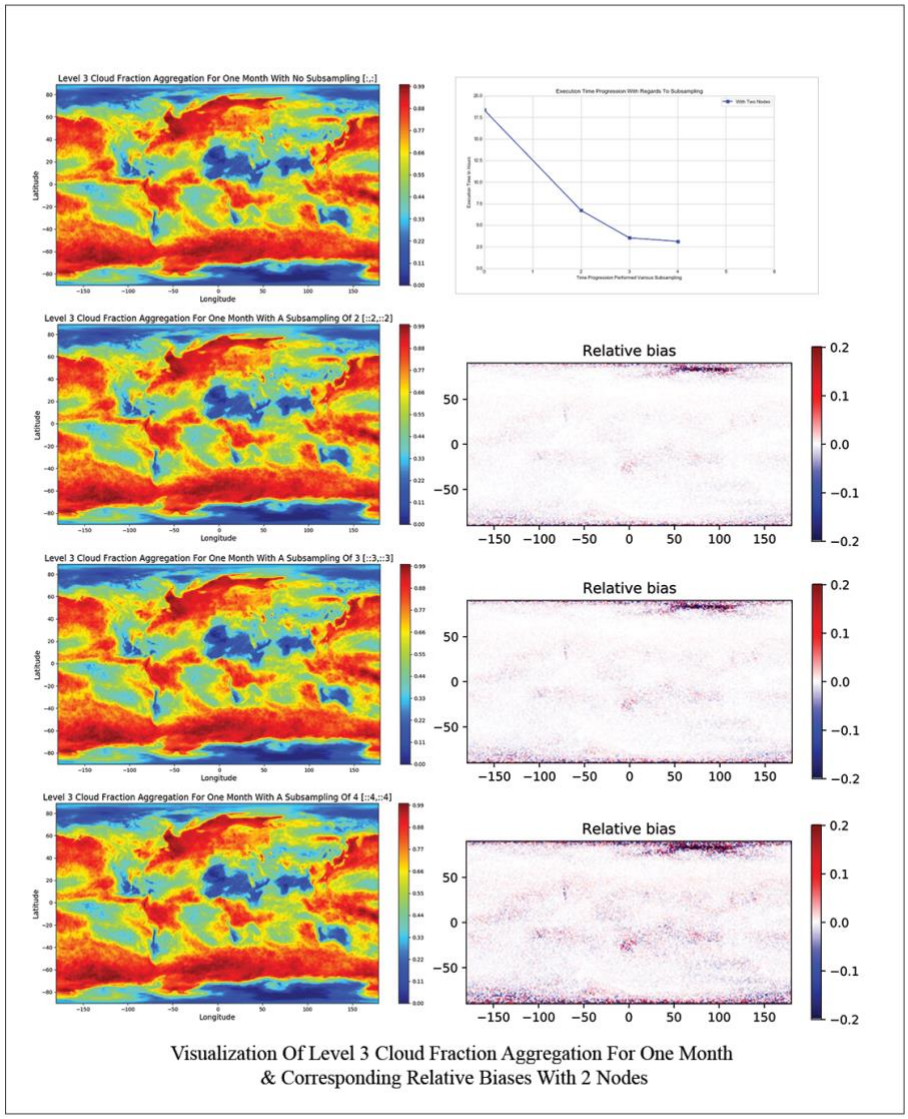
Visualization Of Level 3 Cloud Fraction Aggregation For One Month & Corresponding Relative Biases With 1 Node

Figure 4.1: Visualization Of Level 3 Cloud Fraction Aggregation With 1 Node

Env.	Code File	SLURM File	Subsampling	# Of Nodes	Data Loss (%)	Execution Time
Taki	nosub2n_onemonth.py	nosub2n_onemonth.slurm	0	2	NA	66066.01 Se (18.35 Hrs)
Taki	twosub2jn_onemonth.py	twosub2n_onemonth.slurm	2	2	3.5370	24192.33 Se (6.72 Hrs)

Taki	threesub2n_onemonth.py	threesub2n_onemonth.slurm	3	2	5.4871	12708.29 Se (3.53 Hrs)
Taki	foursub2n_onemonth.py	foursub2n_onemonth.slurm	4	2	2.2154	11232.32 Se (3.12 Hrs)

Figure 4.10: System Execution Time And Data Loss Evaluation Table For Taki HPC With 2 Nodes

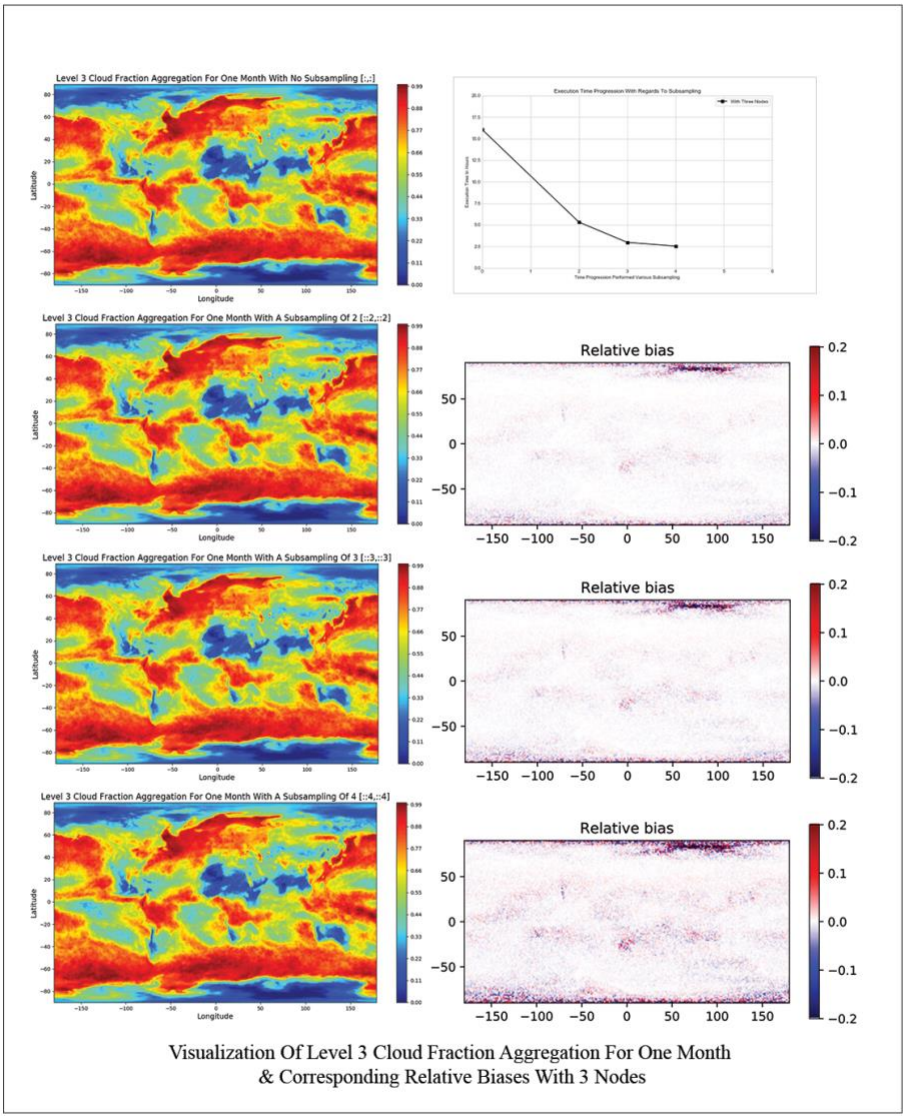


Visualization Of Level 3 Cloud Fraction Aggregation For One Month & Corresponding Relative Biases With 2 Nodes

Figure 4.2: Visualization Of Level 3 Cloud Fraction Aggregation With 2 Nodes

Env.	Code File	SLURM File	Subsampling	# Of Nodes	Data Loss (%)	Execution Time
Taki	nosub3n_onemonth.py	nosub3n_onemonth.slurm	0	3	NA	46836.01 Se (16.05 Hrs)
Taki	twosub3n_onemonth.py	twosub3n_onemonth.slurm	2	3	3.5370	16064.33 Se (5.32 Hrs)
Taki	threesub3n_onemonth.py	threesub3n_onemonth.slurm	3	3	5.4871	10692.29 Se (2.97 Hrs)
Taki	foursub3n_onemonth.py	foursub3n_onemonth.slurm	4	3	2.2154	9144.32 Se (2.54 Hrs)

Figure 4.9: System Execution Time And Data Loss Evaluation Table For Taki HPC With 3 Nodes

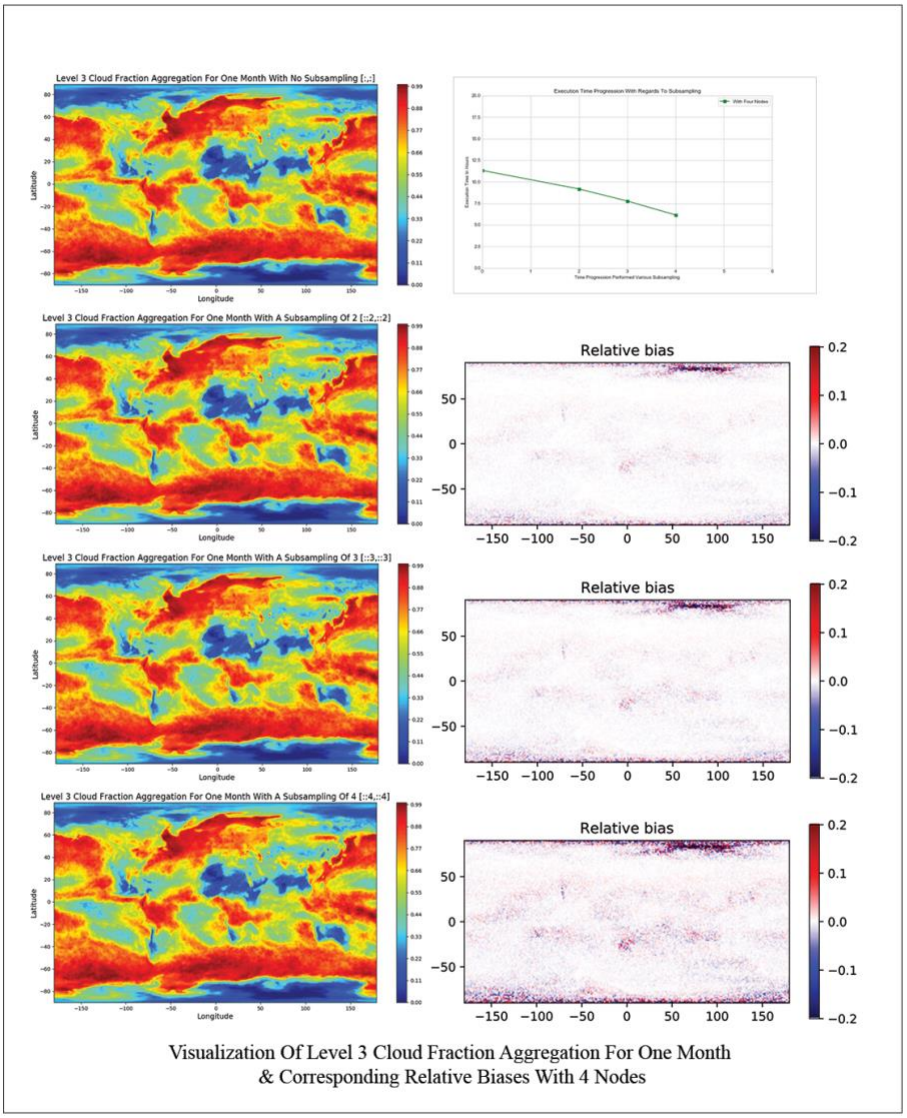


Visualization Of Level 3 Cloud Fraction Aggregation For One Month
& Corresponding Relative Biases With 3 Nodes

Figure 4.3: Visualization Of Level 3 Cloud Fraction Aggregation With 3 Nodes

Env.	Code File	SLURM File	Subsampling	# Of Nodes	Data Loss (%)	Execution Time
Taki	nosub4n_onemonth.py	nosub4n_onemonth.slurm	0	4	NA	57450.86 Se (15.95 Hrs)
Taki	twosub4n_onemonth.py	twosub4n_onemonth.slurm	2	4	3.5370	17425.20 Se (4.83 Hrs)
Taki	threesub4n_onemonth.py	threesub4n_onemonth.slurm	3	4	5.4871	9894.87 Se (2.74 Hrs)
Taki	foursub4n_onemonth.py	foursub4n_onemonth.slurm	4	4	2.2154	9304.60 Se (2.58 Hrs)

Figure 4.7: System Execution Time And Data Loss Evaluation Table For Taki HPC With 4 Nodes

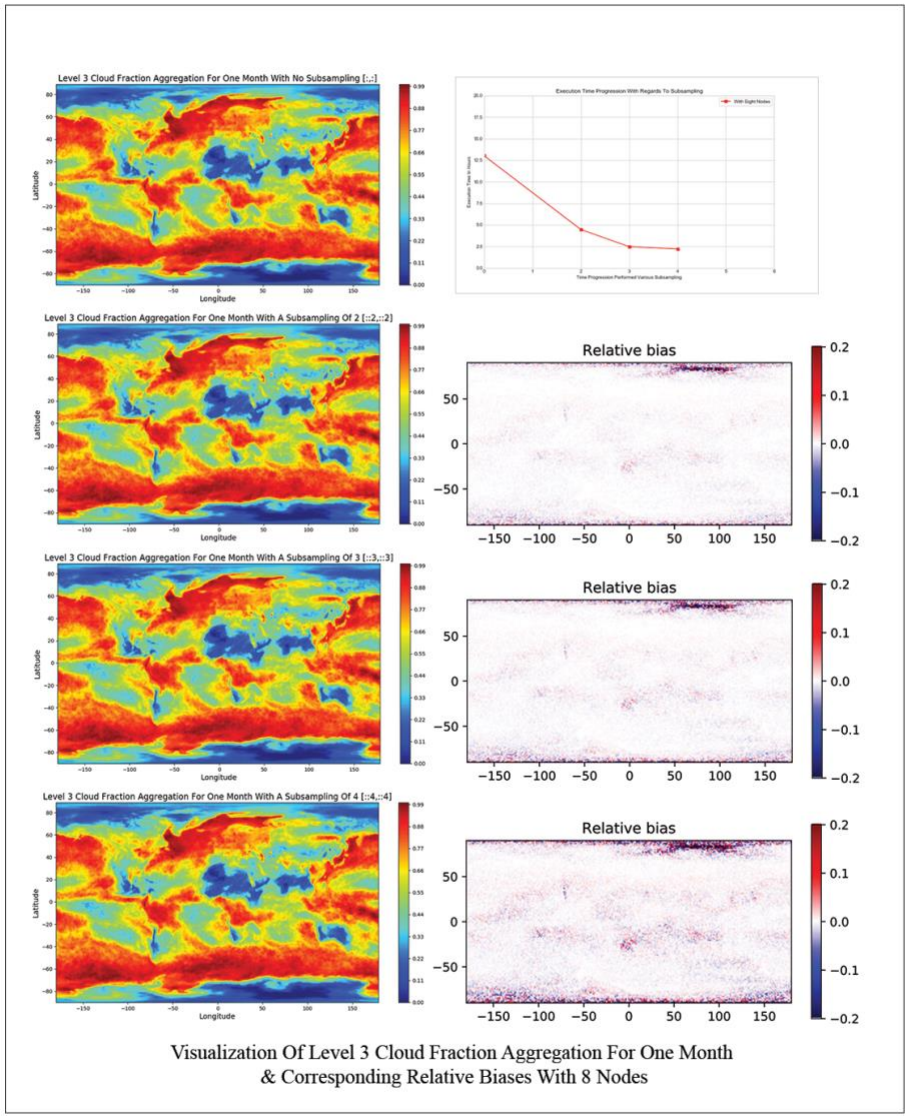


Visualization Of Level 3 Cloud Fraction Aggregation For One Month
& Corresponding Relative Biases With 4 Nodes

Figure 4.4: Visualization Of Level 3 Cloud Fraction Aggregation With 4 Nodes

Env.	Code File	SLURM File	Subsampling	# Of Nodes	Data Loss (%)	Execution Time
Taki	nosub8n_onemonth.py	nosub8n_onemonth.slurm	0	8	NA	46836.01 Sec (13.01 Hrs)
Taki	twosub8n_onemonth.py	twosub8n_onemonth.slurm	2	8	3.5370	16064.33 Sec (4.46 Hrs)
Taki	threesub8n_onemonth.py	threesub8n_onemonth.slurm	3	8	5.4871	8892.29 Sec (2.47 Hrs)
Taki	foursub8n_onemonth.py	foursub8n_onemonth.slurm	4	8	2.2154	7960.32 Sec (2.21 Hrs)

Figure 4.8: System Execution Time And Data Loss Evaluation Table With 8 Nodes



Visualization Of Level 3 Cloud Fraction Aggregation For One Month
& Corresponding Relative Biases With 8 Nodes

Figure 4.5: Visualization Of Level 3 Cloud Fraction Aggregation With 8 Nodes

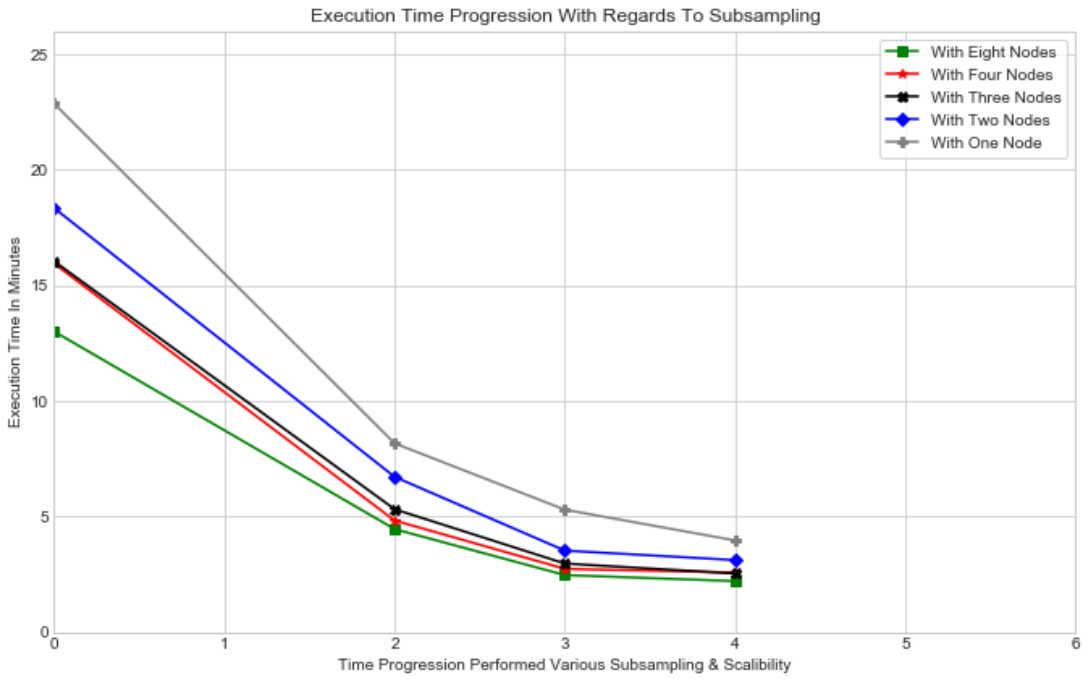


Figure _. _ Execution Time Progression With Regards To Various Subsampling

As we can see after the evaluation, a subsampling of 4 with the dataset for a month renders a 2.2154 % of data loss percentage which can be negligible in certain use cases. In comparison with no subsampling, the computation takes substantial longer time span. The subsampling with 4 takes shorter time span which is about 83.01% decrease in total time with the example of ‘eightnode’ use case.

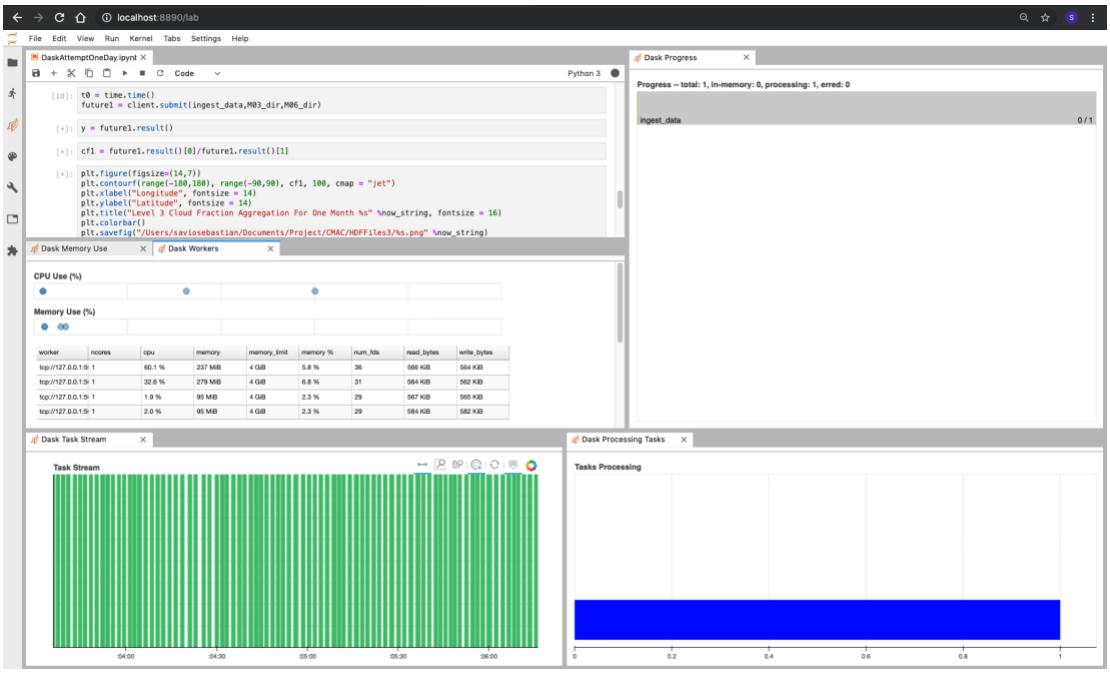


Figure _. _ Execution Time Progression With Regards To Various Subsampling

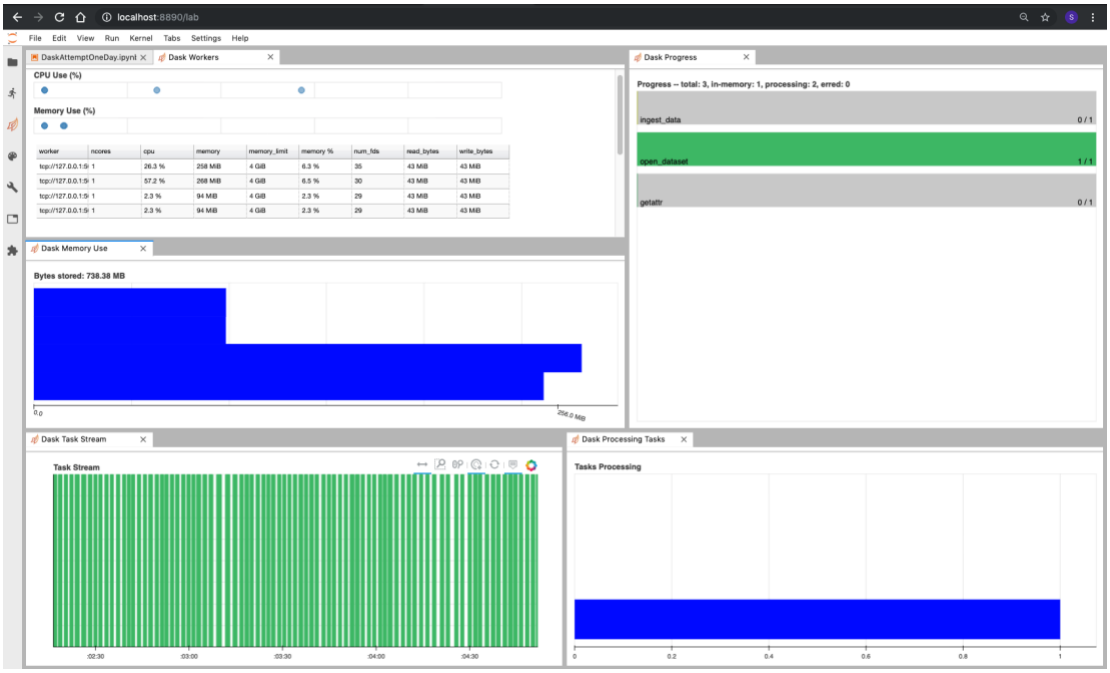
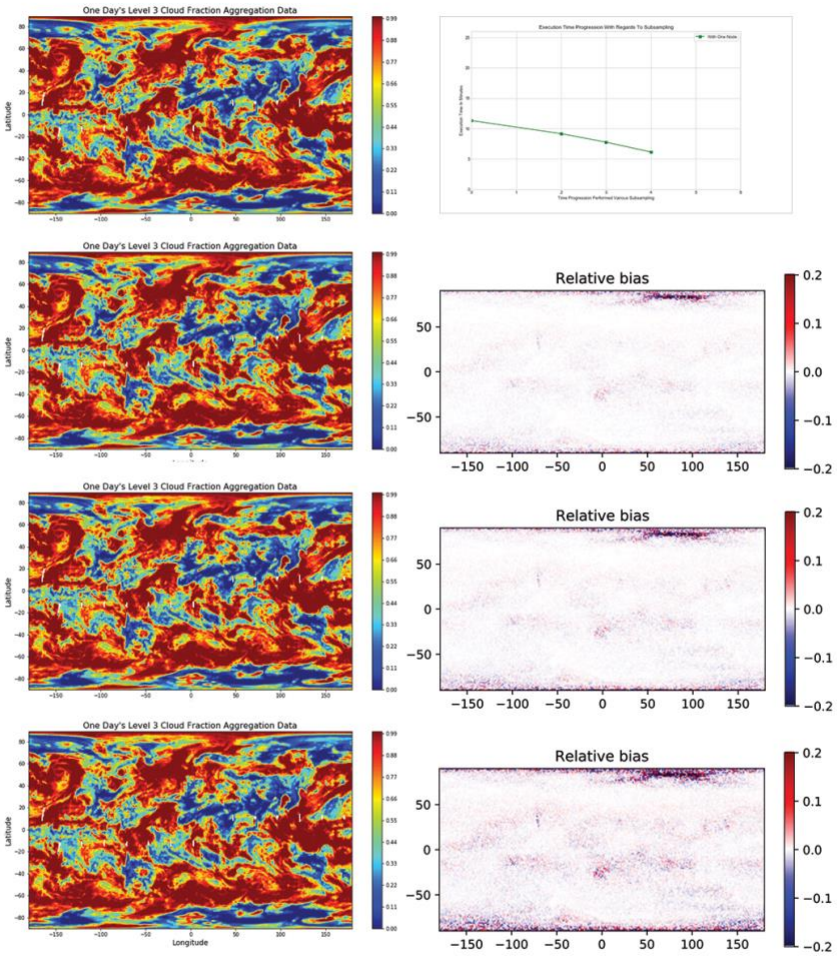


Figure _. _ Execution Time Progression With Regards To Various Subsampling

4.2.1.2 Evaluation Through Binder Cloud Server:

Env.	Code File	Subsampling	# Of Nodes	Data Loss (Percentage)	Execution Time
Binder	nosub1n_oneday.py	0	1	NA	679.201 Sec (11.32 Ms)
Binder	twosub1n_onday.py	2	1	4.8974	550.23 Sec (9.17 Ms)
Binder	threesub1n_oneday.py	3	1	5.6091	466.29 Sec (7.77 Ms)
Binder	foursub1n_oneday.py	4	1	3.3231	369.126 Sec (6.15 Ms)

Figure 4.12: System Execution Time And Data Loss Evaluation Table For Binder Cloud With 1 Node



Visualization Of Level 3 Cloud Fraction Aggregation For One Day
& Corresponding Relative Biases With 1 Node

Chapter 5

Conclusion & Future Work

5.1 Conclusion:

With the intent of the research study, we determine the substantial systematic error or data loss between the original result without subsampling. However, the conventional approach is unable to handle a large amount of dataset in a shorter period. To handle the above pediment, we have to use the various subsampling techniques and combined it with Dask ecosystem which provides parallelism through clustering to determine the result in a shorter time span.

In this study, we discussed the matter of the of concocting a large amount of data by using the high-performance computing clusters environment which distributes the workload across various worker to parallelize the execution. We equated the time taken for each execution and data loss observed in each dataset to identify the visible change between processed dataset. We are testing with the January 2008 month dataset from the NASA portal to identify our study's principal theories.

We assessed our systems by varying the resources of the cluster and the amount of subsampling performed; with this, we equated the total time taken for the entire system to compute the results. Overall, we observed that as the number of subsampling increases, the overall system execution for the result decreases. The results observed in both the environments of high-performance computing and Binder

virtual environment is similar. We also equated the results with variable resources and number of workers in the cluster to see the observed difference in system execution. The error observed to remain the same. However, there was an abundant reduction in overall time extent, particularly about 81% decrease in the overall period in the eight nodes use case.

The Dask SLUR Cluster and Job Queue in the Dask Framework helped us to manage and process the data in real-time parallelly. Internally the system creates the designated workers and shares the workload across the cluster for more accelerated execution. Processing time is the overall time taken by the system to complete all the work process assigned to each worker together. We equated the processing time taken by each use case with a variable number of nodes and scenarios and logged it in the results for evaluation.

5.2 Future Work

With this research, we used the subsampling techniques on January month dataset only. That included 8928 files for one-month processing and 288 files for one-day processing. The system currently processes the entirety of 8928 files for cloud property and equates the mean cloud fraction for the entire month. We plan to use the approach of processing every single day in the entire month for cloud property. It will result in 31 different cloud property files instead of the individual file for the mean cloud property. In the future, we also intend to execute the operation with the above various approaches, calculate its impact, efficiency, and perform a comparison based on the results.

In the current implementation, we use XArray as a data structure and Dask framework to parallelize the execution, whereas in the future we plan to use Dask as the data structure and as a tool to parallelize the workload to get better native support for the system. We also plan to use

different parallelizing tools like Apache Spark, Hadoop to determine the accuracy and efficiency even further.

Bibliography

[1]Group, N. D. (99, May 19). Hierarchical Data Format. Retrieved July 10, 19, from https://support.hdfgroup.org/ftp/HDF/prev-Documentation/HDF4.1r3/Users_Guide/UG41r3_html/Intro.fm1.html

[2]Stevens, J. P. (16, April 06). Why you need metadata for Big Data. Retrieved July 10, 19, from https://www.datasciencecentral.com/profiles/blogs/why-you-need-metadata-for-big-data

[3]Sven-Arne, Monahan, H., Milad, E., M., Fyhn, Marianne, & Anders. (2018, March 23). Experimental Directory Structure (Exdir): An Alternative to HDF5 Without Introducing a New File Format. Retrieved from https://www.frontiersin.org/articles/10.3389/fninf.2018.00016/full

[4]Why should I care about the HDF5 1.10.2 (2018, August 14). Retrieved from https://www.hdfgroup.org/2018/04/why-should-i-care-about-the-hdf5-1-10-2-release/[5]HDF-EOS5 Data Model, File Format and Library | Earthdata. (2018, June 5). Retrieved from https://earthdata.nasa.gov/esdis/eso/standards-and-references/hdf-eos5

[6]Indexing and Selecting Data¶. (n.d.). Retrieved from https://pandas-docs.github.io/pandas-docs-travis/user_guide/indexing.html

[7]Hoye, S. (2018, July 24). XArray Development Roadmap. Retrieved from https://xarray.pydata.org/en/stable/roadmap.html

[8]Merritt, P., & Bi, H. (n.d.). Big Earth Data: A comprehensive analysis of visualization analytics issues. Retrieved from https://www.tandfonline.com/doi/full/10.1080/20964471.2019.1576260

[9] Purdy, R. (n.d.). High Performance Computing Facility. Retrieved from https://hpcf.umbc.edu/system-description-taki/

[10]Nagarajan, M. P., Dr. (2019, July 12). Data Processing Levels. Retrieved from https://oceancolor.gsfc.nasa.gov/products/

[11] Purdy, R. (2018, July 06). High Performance Computing Facility. Retrieved from https://hpcf.umbc.edu/system-description-taki/#heading_toc_j_0

[12]HDF-EOS5 Data Model, File Format and Library | Earth data. (2017, April 25). Retrieved from https://earthdata.nasa.gov/esdis/eso/standards-and-references/hdf-eos5

[13]ZHANG, Z. (n.d.). MODIS Cloud Optical Properties: User Guide for the Collection 6/6.1 Level-2 MOD06/MYD06 Product and Associated Level-3 Datasets. Retrieved from https://modis-atmosphere.gsfc.nasa.gov/sites/default/files/ModAtmo/MODISCloudOpticalPropertyUserGuideFinal_v1.1.pdf

[14]Frazier, S., & Mancheron, B. (2015, October 12). Moderate Resolution Imaging Spectroradiometer Specification. Retrieved from https://modis.gsfc.nasa.gov/about/specifications.php

[15]USER SUPPORT TEAM, M. (n.d.). MODIS/Aqua Geolocation Fields 5-Min L1A Swath 1km Specification. Retrieved from https://gcmd.gsfc.nasa.gov/search/Metadata.do&entry=MYD03_6.1NRT#metadata