awakeExp GSOC 2019 Proposal

Database and Python Analysis Package for AWAKE (Advanced Proton Driven Plasma Wakefield Acceleration Experiment)

Balasubramanyam Evani
balasubramanyam.evani@gmail.com

https://github.com/BalasubramanyamEvani

Mentors: Spencer J. Gessner, Marlene Turner

Abstract

The AWAKE experiment at CERN had generated a total of 13TB data, stored in the form of HDF (Hierarchical Data Format) files. The current proposal is focused on utilizing the capabilities of Python, for interacting with these HDF files in order to develop a database of the datasets along with a package that performs core data analysis on it. Jupyter notebooks which are JSON documents, presents an extremely interactive environment that integrates code and its output. These notebooks are easy to understand and will be used in order to demonstrate the basic features of the library/package created.

1 Objective

The objective of this project is to address the following key points:

- Creating a database out of the generated HDF5 files
- Provide search functionalities, namely
  - String searches to identify the available datasets
  - Boolean searches to identify subsets of data matching some defined condition
- Create tools for analyzing the AWAKE data
  - Extracting data from images
  - Correlating and plotting data across events
- Create Documentation of Interface Created

2 Motivation

CERN is the place where internet was born and is also the place where numerous particle accelerator experiments have been performed. Now, the fact that CERN has proposed a project to create a database and do analysis on its collected data from AWAKE made me feel excited. The amount of data that has been generated is tremendous and as an aspiring data scientist this presents me with the opportunity to work on numerous algorithms in order to best make sense out it, also its a great opportunity to develop such a package that would help scientists ease their work of querying data and easily perform appropriate analysis on it.

3 Approach

Before going forward with the performed experiments and final approaches, I would like to point out that the respective programs has been written in python3.5 and run on a personal laptop with intel core i5 processor (8^{th} generation) and 8GB of RAM, OS is Ubuntu 18.04 LTS. Experiments are performed on the 10 HDF files provided to us for experimentation purposes.

There are three API's available in python in order to interact with the HDF5 files [1] viz. **h5py, pyTables and pandas**. Pandas is exclusively a data analysis library rather than just an interface and so for the application of creating a database, I compared h5py [2] and pyTables [3] on the basis of supported data types, speed of traversing all nodes of all 10 *.h5* files. The speed mentioned is the best out of 5 executions. (please see appendix 1a for source)

Table 1: h5py and pyTables comparison

Name	Type	Description	dtypes	Speed
h5py	API for HDF5	python interface to hdf5	int, float, complex, compound, string, array, enumeration, boolean	3 s
pyTables	Database	python API to organize and manipulate numeric objects	boolean, int, uint, float, complex, string, time, enum, all NUMPY dtypes	8 s

pyTables performed a little slower if compared to h5py, although by not much, looking at the read data, exceptions were caught during the h5py reading suggesting some dtypes were not supported where as result from pytables showed that all dtypes datasets were read confirming the point all dtypes of Numpy as supported in pytables. Hence, I decided to use pytables for future experimentation since it also felt more pythonic and presented an easier interface.

The second decision was to choose between SQL or NoSQL or if at all use any database. Giving a succinct description, NoSQL are harder to administrate but they come with the advantage that the schema can be made flexible where as SQL type DB are sometimes faster than NoSQL if NoSQL is not configured correctly, but it has a fixed schema. Now the objective at hand needs a DB such that it is **created once** and data to be stored is **Name of the dataset, Source File from where it was taken, Size, Shape, Content and dtype** of it, looking at it, SQL DB should be able to do the job well in this case as it is easier to administer and has a constant schema. I selected **SQLite DB** and **MySQL DB** as two databases for testing purposes since these two are most used in applications. I stored the data (group and datasets) in these two DB's with the following structure (I didn't include actual data during testing) and created individual tables for each *.h5* file read:

Table 2: Table columns

Name	RecordType	DatasetSize	DatasetShape	dtype

The second way of storing could be by the use of **External Links** in HDF files. I got to know of this through my question on [4]. An external link is essentially a link to another node (group or dataset) in another HDF file. Hence, by storing external links to other files, we can access datasets from those externally linked files as well. So, I created a new *.h5* file and stored the links of all 10 *.h5* files in it.

The third way could be to store the information in **CSV files**, CSV stands for Comma Separated Value Files. Pandas [5] provides fast interface for reading CSV files and querying data from it. So, I included this as one of the methods of comparison too. Created 10 CSV files for 10 *.h5* files.

The following is the string search performance analysis for the mentioned three types of storage, results are min and max of 10 executions.

Table 3: Performance Analysis

QueryString	DB	Time
"BCTF"	SQLite	6 - 7 ms
"BCTF"	MySQL	12 - 15 ms
"BCTF"	CSV	86 - 100 ms
"BCTF"	ExternalLinks	2.8 s

All three resulted in the same results, but SQLite performs much better than its counterparts. Rather than storing data in multiple tables, I also tried storing it in a single table to see whether it has any affect on the performance, as such I didn't find any significant difference in performance. Based on these experiments I have used SQLite DB approaches for the database creation task.

3.1 Database Creation

Libraries used:

- pyTables
- sqlite3
- io
- numpy
- pickle

3.1.1 Approach 1: Store All data on SQLite DB

In this approach, I propose to use *pyTables* for traversing all non-empty datasets (size is 0) in the HDF files and use *sqlite3* for querying data. *sqlite3* provides methods for registering adapter and converting [6] so that custom dtypes can be stored in the DB. I propose to use this and store all the datasets onto the SQLite DB

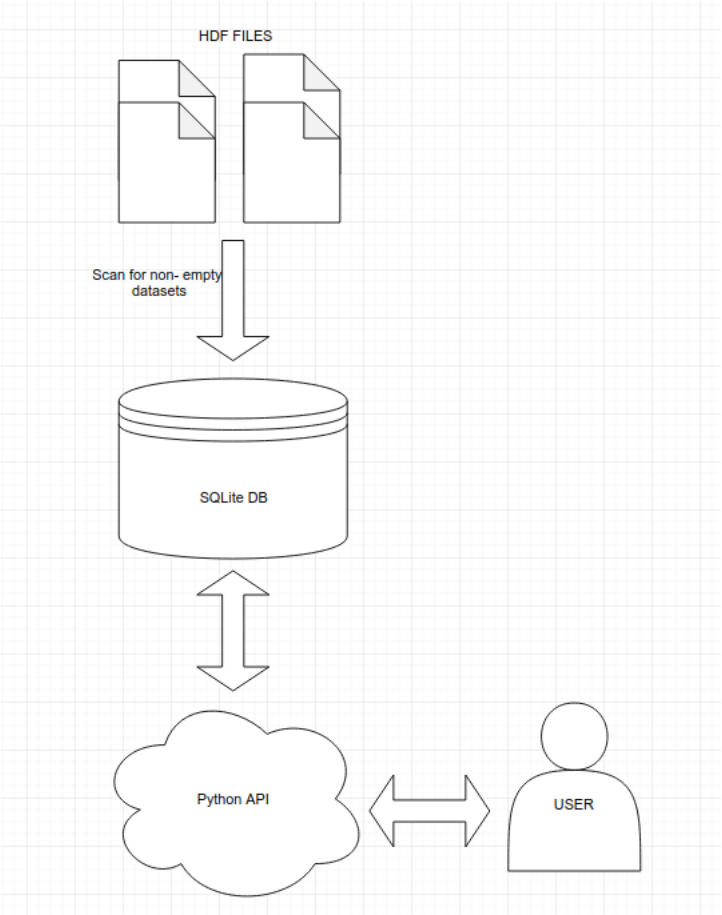

Figure 1: Representation of Approach 1

The columns of table would be the following:

Table 4: Table columns for Approach 1

Name (TEXT)	Size (TEXT)	Shape (TEXT)	Dtype (TEXT)	singVal (TEXT)	NumpyData (Array)

- Name: Dataset Name also contains source h5 files appended to it
- Size: Dataset Size
- Shape: Dataset Size
- Dtype: Dataset datatype

- singVal: Singular value datasets will be stored in this column, stored as string since it takes up less space than if stored as float
- NumpyData: Array items will be stored here, when called/queried it will return data in numpy format only

3.1.2 Approach 2: Store in SQLite DB and HDF file

In this proposed approach, I propose to store singular value datasets, meta-data such as name, size, shape, dtype of datasets in SQLite DB and create another *.h5* file where all array datasets will be stored, the path to these in the created *.h5* will be stored in the SQLite DB under the Data column.

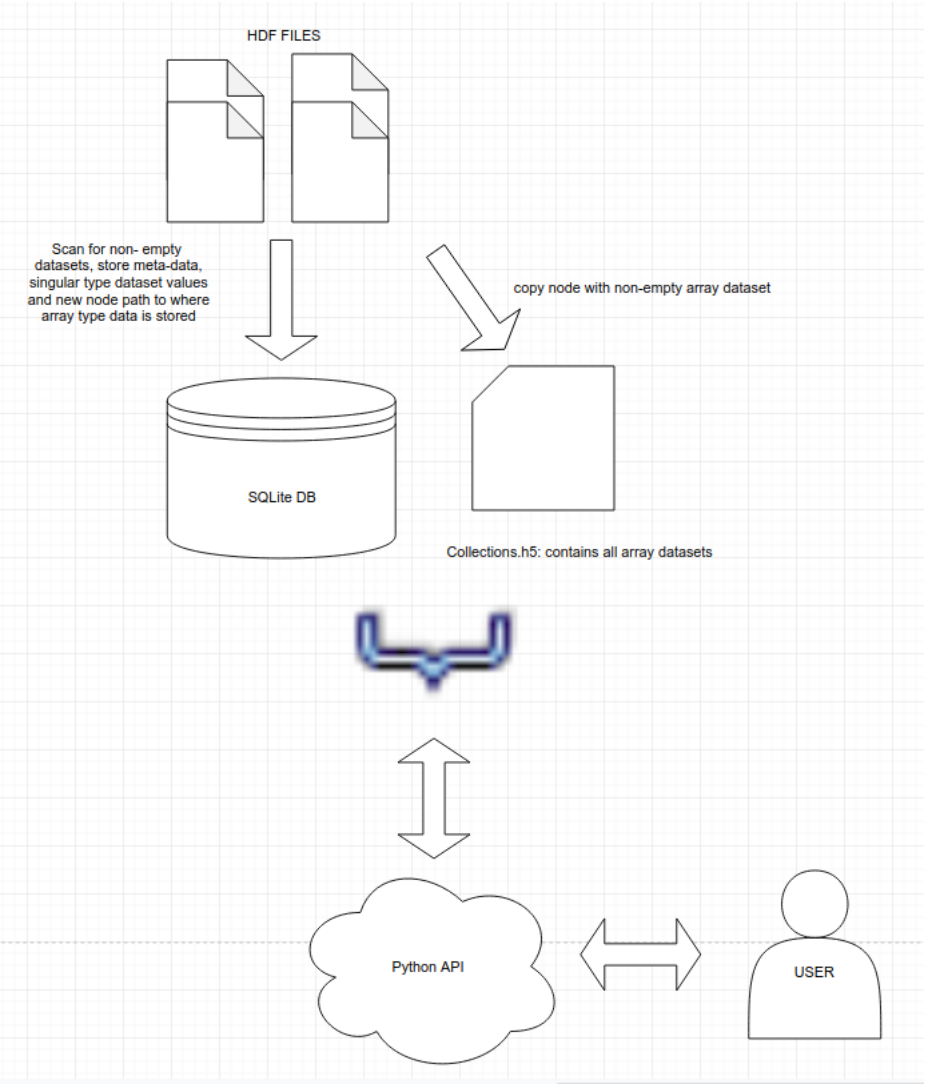

Figure 2: Representation of Approach 2

The columns of table would be the following:

Table 4: Table columns for Approach 2				
Name (TEXT)	Size (TEXT)	Shape (TEXT)	Dtype (TEXT)	Data (TEXT)

- Name: Dataset Name also contains source h5 files appended to it
- Size: Dataset Size

- Shape: Dataset Size
- Dtype: Dataset datatype
- Data: Stores path to Array items in created *.h5* file and values of datasets with size 1

3.1.3 Approach 3: Store in SQLite DB and Pickle files

In this proposed approach, I propose to store singluar value datasets, meta-data such as name, size, shape, dtype of datasets in SQLite DB and use object serialization-deserialization module *pickle*,(Note *cpickle* of **python2** is equivalent to the *pickle* of **python3**), each array dataset will be stored in *.pkl* file. JSON is another serialization and deserialization package but it is harder to create json dumps for numpy arrays, although it can be done, pickle offers much better performance for reading data.

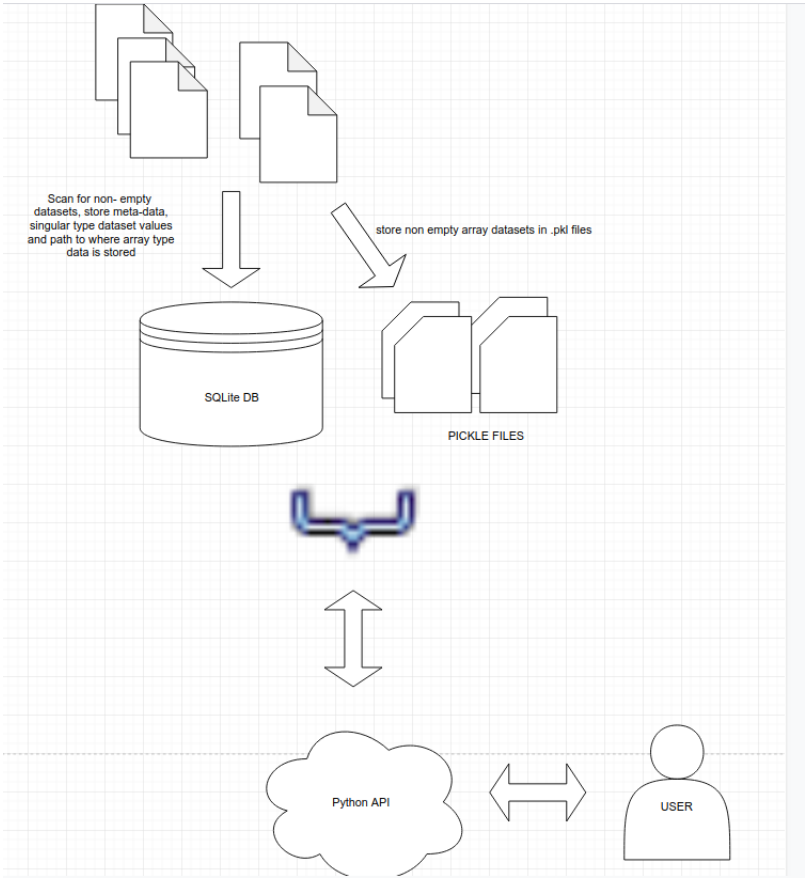

Figure 3: Representation of Approach 3

The columns of table would be the following:

Table 5: Table columns for Approach 3				
Name (TEXT)	Size (TEXT)	Shape (TEXT)	Dtype (TEXT)	Data (TEXT)

- Name: Dataset Name, also contains source h5 files appended to it
- Size: Dataset Size
- Shape: Dataset Size
- Dtype: Dataset datatype

- Data: Path to Array items will be stored here, when called/queried it will return path of created *.pkl* file and stores values of datasets having size 1

Apart from databases, I decided to explore Elasticsearch which is an open source search engine built on top of apache Lucecne, it stores data as JSON document. Elasticsearch requires java, I worked using elasticsearch python client [7]. Elasticsearch main components include `node`, `cluster`, `index`, `doc_type`. It also supports sharding, sharding is done automatically by the engine. An index can be thought of as a database with doc type being the table. Node is subset of cluster. Whenever we start the elasticsearch service, we start a node instance of a cluster. The following presents proposed approach 4.

3.2 Approach 4: Elasticsearch and Pickle files

Similar to approach 3, the array type items are store in form of pickles and the path is stored in the search engine along with values of datasets with size 1 to perform boolean queries. The Elasticsearch mapping is constructed as follows:

```
        mapping(
                index="cdb_testdemo",
                doc_type="datasets",
                body={
                    "properties": {
                        "Name": {"type":"text"},
                        "Size": {"type": "text"},
                        "Shape": {"type": "text"},
                        "Dtype":{"type":"text"},
                        "Data":{"type":"text"}
                    }
                }
            )
```

In this approach, all the files are first scanned and then put in a pandas data frame before being transferred as bulk to elasticsearch engine.

3.2.1 Comparison Analysis

The following table presents the time it took for querying results using the four proposed methods. (please refer to appendix B and C for detailed code)

Table 6: Query Time Analysis (Results are best out of 5)

Query	Type	Database	Time
		Approach 1	18 ms
"BCTF"	StringSearch	Approach 2	5 ms
		Approach 3	5 ms
		Approach 4	9 ms
		Approach 1	6 ms
		Approach 2	7 ms
"/AwakeEventData/TT41.BCTF.412340/Acquisition/totalCurrentPreferred > 0.068"	BooleanSearch	Approach 3	7 ms
		Approach 4	4 ms
		Approach 1	16 ms
15419621089350000000_167_838.h5:/AwakeEventData/TSG40.AWAKE-LASER-CORRELATOR/FileRead/dataImage	getArray Data	Approach 2	3 ms
		Approach 3	1 ms
		Approach 4	1 ms

Figure 4: Query Comparison Results

Table 7: Pros and Cons

Approach	Pros	Cons
1	no dependence on HDF or pickle files, easier interface	database size would be large, querying time the slowest out of three
2	faster access to array data, SQLite database size not large since all array type datasets on separate file	size of HDF created : 569 MB, dependence on external HDF file, interface would be little bit complex but not much
3	fastest access to array data, database size not large	size of all pickle files sum to 562.1 MB, dependence on pickle files, interface bit complex but not much
4	fastest access to array data, better boolean query time	interface written is little bit hard to write

Regarding SQLite DB, the database is extremely portable and since the intended database is to be created once, SQLite is extremely suitable for this. The only problem seems to be the security, SQLite doesn't provide any security features like MySQL DB for data access. Since, SQLite can be placed in filesystem, the directory where it is kept can be protected or we can use pysqlcipher [8] to encrypt the database itself. Also to note I mentioned storing data into pickle files, pickle is not safe if the source of data is not secure, I considered storing in this format since I thought the HDF files were obtained from secure source.
Elasticsearch does not present portability but has much better security features than SQLite DB, it works similar to http requests and if indexed properly may lead to better performance.

In the end I presented possible four methods of creating a database that I believe are easier to manage, build and has good performance capabilities. My aim will will be to incorporate one of the mentioned methods as per the organization needs.

3.3 Analysis Package

3.3.1 Image Data

Processing of Images involves tasks such as filtering, curve fitting, enhancement and many more. Python has a plethora of packages for analyzing this; few of them include; scipy [9], scikit-image processing [10] and OpenCV package [11]. `matplotlib` and `seaborn` are some plotting modules that can be used for creating visually appealing graphs.

3.3.2 scipy

scipy provides some amazing methods for filtering images such as the scipy.signal package consists of methods relating to noise filtering in images, one such example is the medfilter (implements median filter with defined window size), medfilter is good for salt and pepper noise. scipy.optimize provide method `curve_fit`, this can be used in order to fit any defined function/curve such as poisson or gaussian curve to the data.

In the following images, I showed a little example on how it can be applied to awake data, I used the medfilt and `scipy.optimize.curve_fit` in order to filter a numpy image data and fit a defined gaussian curve to it. Since, `curve_fit` uses an initial guess I used another method from scipy.optimize named `differential_evolution` which uses genetic algorithm (GA, a meta-heuristic optimization technique), so the initial guess to the `curve_fit` is given by result obtained from GA. (please refer to appendix D for source)

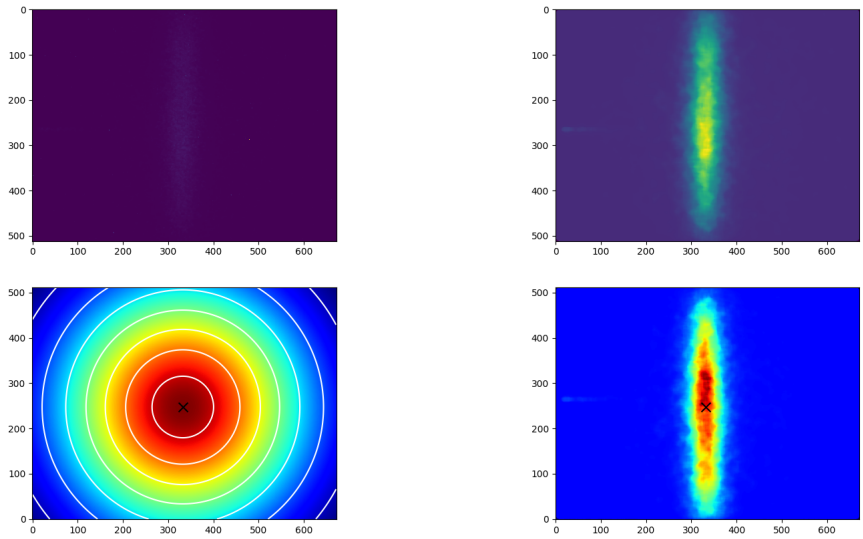

Figure 5: top left: original, top right: median filtered, bottom left: predicted gaussian, bottom right: peak point in filtered image

3.3.3 scikit-image

scikit-image also provide a number of methods for filtering, morphological transformations, edge and line detection and many more. The cross correlation function of scikit-image, can be applied with successive set of images in order to plot relative shift between them.

3.3.4 OpenCV

OpenCV python wrapper is also one the most used packages and comes bundled with many deep learning methods for image processing tasks.

3.4 Correlation among Data

The awake data contained tremendous amount of data, as it is with 2D points, what better to visualize them initially than a scatter plot. Additionally, correlations can also be found out within the points, whether it is projects a positive correlation, negative correlation or any correlation at all. Correlation refers to the covariance or dependence on variables, through this we can perform a regression analysis in order to predict the next outcome.

There are many types of regression, some are `Linear Regression`, `Multiple linear regression`, `Ridge and Lasso` etc. Regression analysis presents a powerful statistical tool for curve fitting, seeing the dependence among dependent or independent variables. `pandas` with `numpy` and `matplotlib` presents a great way of presenting correlations in form of graphs and matrices.

Data can also be used to perform `clustering`, in order to find the number of clusters among the data points. Clustering refers to the ordering of points such that same type of points belong to one class. A number of clustering algorithms can be tested some are:

- K-Means Clustering
- DBSCAN (Density based clustering)
- BIRCH Clustering

4 Brief Timeline

- Pre-GSOC period (till May 27)
- Coding Phase 1 (27 May 2019 - 24 June 2019)
- Phase 1 submissions (24 June 2019 - 28 June 2019)
- Coding Phase 2 (till 22 July 2019)
- Phase 2 submissions (22 July 2019 - 26 July 2019)
- Coding Phase 3 (till 19 August 2019)
- Final Submissions and Evaluation Period (19 August 2019 - 26 August 2019)

4.1 Pre-GSOC period

During this period I will setup the necessary environments on my laptop, install necessary libraries. In addition to this I will refine my milestone deliverables with the help of my mentors, make necessary changes and discuss more about functions that can be added and get familiar with the SWAN service and the AWAKE Experiment.

4.2 Coding Phase (till 1st June 2019)

I will start working on developing a skeleton work=flow for the project and basic schema design for the database.

4.3 Coding Phase (1 June - 8 June 2019)

I will not be available much of the time, since this is my final year of engineering I need to present my final year project to the college. The viva voce will be conducted during this period. Sorry for this still i will definitely try to do as much as possible during this week.

4.4 Coding Phase (8 June 2019 - 24 June 2019)

During this period I will develop the agreed on database design and implement an interface for querying data from it. I will work on creating test cases where the interface might fail and rectify it. Document the scripts written and create Jupyter Notebooks for future users. Finally look for more bugs, improving the flow of querying if possible and wrap it for phase 1 submission.

4.4.1 Phase 1 Deliverables

- Well documented API for connecting with the database and querying data from it
- The API would be able to implement StringSearches, BooleanSearches and get datasets from database
- Example Notebooks showing some examples

4.5 Coding Phase (till 22 July 2019)

Start full blown work on analysis package for awakeExp, initially I plan to tackle this in two parts. The first part would be completed during this coding phase and the other in the next coding phase. The first phase would consist of analyzing image datasets. with the help of discussed techniques in |section 3.2.1|. I will be implementing functions for cross correlations between images, filtering methods for noise remove, thresholding, curve fitting and if time still remains explore some machine learning approaches in order to perform image segmentation.

4.5.1 Phase 2 Deliverables

- Well documented Analysis Package for Image analysis
- Create Example Notebooks showing some examples

4.6 Coding Phase 3 (till 19 August 2019)

This phase will be the concluding part of the analysis package which will include the development of analysis tools that will find correlations in the series data, perform regression analysis on it and predict the success and failure of an experiment, some algorithms have been discussed in the `section 3.3` other than those I will try to implement other machine learning algorithms (SVM, Neural Networks) for its possible incorporation. The second would be to perform clustering analysis on the data as it will help in finding a structure on unlabelled data. Lastly, go over the entire package created find possible bugs in it and fix, well document it and provide example Notebooks for users.

4.6.1 Phase 3 Deliverables

- Creation of full analysis package along with example Notebooks
- Write appropriate setup scripts
- Submit the whole package for evaluation along with it I would be writing a detailed report on this project.

4.7 Note

Along with the mentioned tentative work to be done, I will also be involved in studying about the AWAKE experiment and physics behind, this will further enhance my understanding on the subject and will enable me to develop appropriate analysis tools that will be useful for users.

I will be mailing mentors on a weekly basis updating the status of the work. Open Source is all about communication, hence I will be in continuous contact with the mentors. I will prepare a more refined checklist once everything is finalized during the pre-GSOC period and share it via flock or any other agreed on portal, so that mentors can easily track my work and add new tasks for me to complete.

5 Personal Information

I am final year engineering student in Manipal University Jaipur, Rajasthan, India pursuing Electronics and Communication Engineering. This is my first time participating in GSOC and haven't applied to any other organization other than this. I may be new to the open source community, but I intend to learn a lot from this platform but most importantly deliver/develop a robust interface for usage, I have taken up courses on Electromagnetic theory, Microwave Engineering during my undergraduate studies so I believe I would be able to understand the physics behind the experiment. Since my final exams will be over by 1^{st} week of June, and nothing to do during summers, I can easily give 35-40 hrs for the project each week. My local time zone throughout summer will be Indian Standard Time (IST), GMT+05:30.

5.1 Experience

5.1.1 Research Intern at Indian Institute of Technology, Kanpur, U.P, India (May - July 2018)

Developed Multi-Robot Multi-Sensor Dataset Repository for Short and Long Term SLAM (Simultaneous Localization and Mapping).

`Used` : ROS Kinetic, Python, C++

5.1.2 Research Intern at Indian Statistical Institute, Kolkata, West Bengal, India (January 2019 - Present)

I have been conducting my final year research project at Indian Statistical Institute, Kolkata on Multi-Label Classification using Neural Networks.

`Using` : MATLAB, Python

Apart from these internships, I am also involved in a project involving image enhancement using meta-heuristics at National Institute of Technology, Jaipur, Rajasthan, India.

6 Appendix

6.1 A

```
1
2  #!/usr/bin/env python3
3
4  ##########################################################################
5  #
6  # Created : 25th March 2019
7  # Author: balasuburamanyam[dot]evani[at]gmail[dot]com
8  # AWAKE GSOC 2019 Experiments
9  # traverse.py
10 ##########################################################################
11
12 import h5py
13 import numpy as np
14 import tables as tb
15
16 class task_h5py(object):
17
18     def __init__(self,file_name):
19
20         self.mat = []
21         self.file_name = file_name
22
23     def recurr(self,h,root=''):
24         for key in h.keys():
25
26             item = h[key]
27             path = root+'/'+str(key)
```

```
            name = self.file_name.replace("../hdf_files/","")+":"+path
            if isinstance(item, h5py.Dataset): # checks whether instance is dataset
                try:
                    data_type = item.dtype
                except Exception as error:
                    data_type = str(error)
                if item.size is not 0:
                    self.mat.append([name,'dataset',str(item.size),str(item.shape),str(data_type)])

            elif isinstance(item, h5py.Group): # checks whether instance is group
                self.mat.append([name,'group','','',''])
                self.recurr(item, path)

    def traverse_and_save(self):

        print('---- traversing the provided file ----')

        with h5py.File(self.file_name, 'r') as f:
            self.recurr(f)

        f.close()

class task_pytables(object):

    def __init__(self,file_name):

        self.mat = []
        self.file_name = file_name

    def traverse_and_save(self):

        print('---- traversing the provided file ----')
        f = tb.open_file(self.file_name, mode='r')
        for node in f:
            shape = size = dtype = ''
            RecordType = "group"
            name = self.file_name.replace("../hdf_files/","")+":"+str(node).split(" ")[0]
            if isinstance(node, tb.group.Group) == False:

                data = f.get_node(node).read()
                size = data.size
                if size is 0:
                    continue
                shape = node.shape
                dtype = node.dtype
                RecordType = "dataset"

            self.mat.append([name, RecordType, str(size), str(shape), str(dtype)])
        f.close()

###############
#
# Main file calls traverse.py
#
###############
import os
from traverse import *
import time

if __name__=="__main__":

  tick1 = time.time()
  source = "../hdf_files"
  for file in os.listdir(source):
    test_2  = task_pytables(source+"/"+file)
    test_2.traverse_and_save()
  tock1 = time.time()

  tick2 = time.time()
  source = "../hdf_files"
  for file in os.listdir(source):
    sort_records = False
```

```
    test_2  = task_h5py(source+"/"+file)
    test_2.traverse_and_save()
  tock2 = time.time()

  print("for pytables: ",int(round((tock1-tick1))),"s")
  print("for h5py: ",int(round((tock2-tick2))),"s")
```

6.2 B

```

#!/usr/bin/env python3

#########################################################################
#
# Created : 25th March 2019
# Author: balasuburamanyam[dot]evani[at]gmail[dot]com
# AWAKE GSOC 2019 Experiments
#
#########################################################################

'''
Helper Library for performing experiments
'''

'''
Loading essential libraries
'''

import numpy as np
import tables as tb
import sqlite3
import io
import pickle
import os

#########################################################

class cern_database_type1(object) :

    """
    cern_database_type1 contains methods to append data from h5 files in the following format
    metadata such as : dataset size, shape, name, dtype and data having singluar value stored in
    SQLite DB as well as
    NumpyArray, the numpy arrays are stored in the custom format of type numpy, such that querying
    of data returns in numpy ndArray
    too

    """

    def __init__(self, sqlite_cur) :

        """
        i/p: cursor to SQLite DB

        """

        self.sqlite_cur = sqlite_cur

    def __append__(self, h5_file) :

        """
        Appends dataset to SQLite table as read from h5 file
        """

        print("storing data from file:",h5_file,"to SQLite database")

        flag = len(self.sqlite_cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND
    name='CERN_DATASETS1'").fetchall()) > 0;

        if flag is False :
          self.sqlite_cur.execute("CREATE TABLE CERN_DATASETS1 (Name TEXT, Size TEXT, Shape TEXT,
    Dtype TEXT, singVal TEXT, ArrayData array);")
```

```
60
61          f = tb.open_file(h5_file, mode='r')
62          source_file = h5_file.replace("hdf_files/","")
63
64          for node in f :
65
66              if isinstance(node, tb.group.Group) is False :
67
68                  name = str(node).split(" ")[0]
69                  data = f.get_node(node).read()
70                  size = data.size
71                  shape = str(data.shape)
72                  dtype = str(data.dtype)
73
74                  if size is 0 :
75                      continue
76
77                  elif size is 1 :
78
79            ## if bytes need to be stored in string, uncomment this
80
81                      # if isinstance(data.tolist(), bytes):
82                      #   value = str(data.tolist(), 'utf-8')
83                      # else:
84                      #   value = str(data).lstrip('[').rstrip(']')
85
86                      value = str(data).lstrip('[').rstrip(']')
87                      data = None
88
89                  else :
90
91                      value = None
92
93                  frame = [source_file+":"+name, size, shape, dtype, value, data]
94                  self.sqlite_cur.execute("INSERT INTO CERN_DATASETS1 VALUES (?, ?, ?, ?, ?, ?);",
    frame)
95
96          f.close()
97
98
99      def __conditionSwitcher__(self, condition) :
100
101         """
102         parsing conditions for __booleanSearch__
103         """
104         my_dict = {
105             '-gt' : '>',
106             '-lt' : '<',
107             '-lteq' : '<=',
108             '-gteq' : '>=',
109             '-eq' : '='
110         }
111
112         return  my_dict.get(condition, "No selection")
113
114     def __stringSearch__(self, search_string, strict=False, selection="*") :
115
116         """
117         String search method
118         search_string: type(str); string pattern to search
119         strict: boolean ; whether to exactly match or partially
120         selection: type(str); default -> * (selects all columns) or provide string with column
    names to return
121         pass string of columns like this "Name, Dtype"
122
123         returns list of matching rows
124         """
125
126         if strict is False :
127             sql = "SELECT {S} FROM CERN_DATASETS1 WHERE Name LIKE '%{ss}%';".format(S=selection, ss
    =search_string)
128         elif strict is True :
```

```
129             sql = "SELECT {S} FROM CERN_DATASETS1 WHERE Name LIKE '{ss}';".format(S=selection, ss=
    search_string)
130 
131         self.sqlite_cur.execute(sql)
132         results = self.sqlite_cur.fetchall()
133         return [list(i) for i in results]
134 
135     def __boolSearch__(self, search_string, condition, val, strict_match=False, selection="*") :
136 
137         """
138         Method implements booleanSearch for SingluarValue Data
139         search_string: type(str); dataset name
140         strict: boolean; False -> partial, True -> full match; default-> False
141         selection: default-> All columns selected, pass string of columns like this "Name, Dtype"
142         condition: please __conditionSwitcher__ for supported conditions
143 
144         returns list of matching rows
145         """
146         try :
147             val = float(val)
148         except ValueError:
149             print("val should be a number")
150             pass
151 
152         condition = self.__conditionSwitcher__(condition)
153         if condition is "No selection":
154             raise ValueError("wrong condition selection")
155 
156         if strict_match is False :
157             sql = "SELECT {S} FROM CERN_DATASETS1 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name
    LIKE '%{ss}%' AND CAST(singVal as REAL) {c} {v};".format(S=selection, ss=search_string, c=
    condition, v=val)
158         elif strict_match is True :
159             sql = "SELECT {S} FROM CERN_DATASETS1 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name
    LIKE '{ss}' AND CAST(singVal as REAL) {c} {v};".format(S=selection, ss=search_string, c=
    condition, v=val)
160 
161         self.sqlite_cur.execute(sql)
162         results = self.sqlite_cur.fetchall()
163         return [list(i) for i in results]
164 
165     def __getArrayData__(self, dataset, condition="-all", selection="*") :
166 
167         """
168         Fetches ndArray datasets
169         conditions: -all -> return all values
170                     -max -> return max value
171                     -min -> return min value
172         selection: * -> select all columns
173         dataset: name of dataset; strict matching is performed
174         """
175 
176         if condition not in ["-max", "-min", "-all"] :
177             raise ValueError("wrong condition selection")
178 
179         sql = "SELECT {S} FROM CERN_DATASETS1 WHERE Size > 1 AND Name LIKE '{d}' ;".format(S=
    selection, d=dataset)
180         self.sqlite_cur.execute(sql)
181 
182         result = self.sqlite_cur.fetchall()
183 
184         if condition == "-max" :
185             return np.max(result[0][5])
186         elif condition == "-min":
187             return np.min(result[0][5])
188         else:
189             return result[0][5]
190 
191 ############################################################
192 
193 ############################################################
194 
```

```
class cern_database_type2(object) :

    """
    cern_database_type2 contains methods to append data from h5 files in the following format
    metadata such as : dataset size, shape, name, dtype and data having singluar value stored in
    SQLite DB
    Numpy Array data stored in a separate h5 file

    """

    def __init__(self, sqlite_cur) :

        """
        i/p: cursor to SQLite DB

        """

        self.sqlite_cur = sqlite_cur

    def __append__(self, h5_file) :

        """

        Writes numpy array dataset/ copies dataset into new file "datasets.h5" in the same folder
        In the new file: the structure is constructed in the following form

        /original_fileName/ds(index); where index is incremented, this file name is stored in the
    SQLite DB column

        """

        ds_h5f = tb.open_file('datasets.h5','a',title='Datasets')

        print('storing data from file:',h5_file,'to SQLite database')

        flag = len(self.sqlite_cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND
    name='CERN_DATASETS2'").fetchall()) > 0;

        if flag is False:
            self.sqlite_cur.execute("CREATE TABLE CERN_DATASETS2 (Name TEXT, Size TEXT, Shape TEXT,
     Dtype TEXT, Data TEXT);")

        f = tb.open_file(h5_file, mode='r')
        source_file = h5_file.replace("hdf_files/","")

        group = ds_h5f.create_group('/',source_file[:-3])
        index = 1

        for node in f :

            if isinstance(node, tb.group.Group) is False :

                ds_name = str(node).split(" ")[0]
                data = f.get_node(node).read()
                size = data.size
                shape = str(data.shape)
                dtype = str(data.dtype)

                if size is 0 :
                    continue

                else :

                    if size is 1:

            ## if bytes need to be stored in string, uncomment this
            # if isinstance(data.tolist(), bytes):
            #   value = str(data.tolist(), 'utf-8')
            # else:
            #   value = str(data).lstrip('[').rstrip(']')

                        value = str(data).lstrip('[').rstrip(']')
```

```
263
264                     else :
265
266                         value = str(group).split(" ")[0]+"/ds"+str(index)
267                         ds_h5f.copy_node(node,newparent=group,newname="ds"+str(index),overwrite=
    True)
268                         index = index + 1
269
270                 frame = [source_file+":"+ds_name, size, shape, dtype, value]
271                 self.sqlite_cur.execute("INSERT INTO CERN_DATASETS2 VALUES (?, ?, ?, ?, ?);",frame)
272
273         f.close()
274         ds_h5f.close()
275
276     def __stringSearch__(self, search_string, strict=False, selection="*") :
277
278         """
279         String search method
280         search_string: type(str); string pattern to search
281         strict: boolean ; whether to exactly match or partially
282         selection: type(str); default -> * (selects all columns) or provide string with column
    names to return
283         pass string of columns like this "Name, Dtype"
284
285         returns list of matching rows
286         """
287
288         if strict is False :
289             sql = "SELECT {C} FROM CERN_DATASETS2 WHERE Name LIKE '%{S}%'".format(C=selection, S=
    search_string)
290         elif strict is True :
291             sql = "SELECT {C} FROM CERN_DATASETS2 WHERE Name LIKE '{S}'".format(C=selection, S=
    search_string)
292
293         self.sqlite_cur.execute(sql)
294         results = self.sqlite_cur.fetchall()
295         return [list(i) for i in results]
296
297     def __conditionSwitcher__(self, condition) :
298
299         """
300         parsing conditions for __booleanSearch__
301         """
302
303         my_dict = {
304             '-gt' : '>',
305             '-lt' : '<',
306             '-lteq' : '<=',
307             '-gteq' : '>=',
308             '-eq' : '=',
309         }
310
311         return  my_dict.get(condition, "No selection")
312
313     def __boolSearch__(self, search_string, condition, val, strict=False, selection="*") :
314
315         """
316         Method implements booleanSearch for SingluarValue Data
317         search_string: type(str); dataset name
318         strict: boolean; False -> partial, True -> full match; default-> False
319         selection: default-> All columns selected, pass string of columns like this "Name, Dtype"
320         condition: please __conditionSwitcher__ for supported conditions
321
322         returns list of matching rows
323         """
324
325         try :
326             val = float(val)
327         except ValueError:
328             print("val should be a number")
329             pass
330
```

```
331         condition = self.__conditionSwitcher__(condition)
332 
333         if condition is "No selection": ## selecting only data
334             raise ValueError("wrong condition selection")
335 
336         if strict is False :
337             sql = "SELECT {S} FROM CERN_DATASETS2 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name
    LIKE '%{ss}%' and CAST(Data as REAL) {c} {v};".format(S=selection, ss=search_string, c=
    condition, v=val)
338         elif strict is True :
339             sql = "SELECT {S} FROM CERN_DATASETS2 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name
    LIKE '{ss}' and CAST(Data as REAL) {c} {v};".format(S=selection, ss=search_string, c=condition,
     v=val)
340 
341         self.sqlite_cur.execute(sql)
342         results = self.sqlite_cur.fetchall()
343         return [list(i) for i in results]
344 
345     def __getArrayData__(self, dataset, condition="-all", selection="*") :
346 
347         """
348         Method to get dataset which are of type numpy array
349         returns max,min or full dataset
350         """
351 
352         if condition not in ["-max", "-min", "-all"] :
353             raise ValueError("wrong condition selection")
354 
355         source_file = "./datasets.h5"
356 
357         f = tb.open_file(source_file,mode='r')
358 
359         try:
360           data = f.get_node(dataset).read()
361           f.close()
362         except Exception as err:
363           print(err)
364 
365         if condition == "-max" :
366             return np.max(data)
367         elif condition == "-min":
368             return np.min(data)
369         else:
370             return data
371 
372 ##########################################################
373 
374 ##########################################################
375 
376 class cern_database_type3(object) :
377 
378     """
379     cern_database_type3
380     metadata such as : dataset size, shape, name, dtype and data having singluar value stored in
    SQLite DB
381     Numpy Array data stored in a separate pickle files, localtion of pickle file stored in SQLite
    DB
382 
383     """
384 
385     def __init__(self, sqlite_cur) :
386 
387         """
388         i/p: cursor to SQLite DB
389 
390         """
391 
392         self.sqlite_cur = sqlite_cur
393 
394     def __append__(self, h5_file) :
395 
396         """
```

```
        generate pickle files for ndArray and store location of it in DB

        """

        print('storing data from file:',h5_file,'to SQLite database')

        flag = len(self.sqlite_cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='CERN_DATASETS3'").fetchall()) > 0;

        if flag is False :
            self.sqlite_cur.execute("CREATE TABLE CERN_DATASETS3 (Name TEXT, Size TEXT, Shape TEXT, Dtype TEXT, Data TEXT);")

        f = tb.open_file(h5_file, mode='r')
        source_file = h5_file.replace("hdf_files/","")
        index = 1

        for node in f :

            if isinstance(node, tb.group.Group) is False:

                ds_name = str(node).split(" ")[0]
                data = f.get_node(node).read()
                size = data.size
                shape = str(data.shape)
                dtype = str(data.dtype)

                if size is 0 :
                    continue

                else :

                    if size is 1 :

                        ## if bytes need to be stored in string, uncomment this

                      # if isinstance(data.tolist(), bytes):
                      #   value = str(data.tolist(), 'utf-8')
                      # else:
                      #   value = str(data).lstrip('[').rstrip(']')

                        value = str(data).lstrip('[').rstrip(']')

                    else :

                        path = './pickle_files/'+source_file[:-3]
                        if not os.path.exists(path) :
                          os.makedirs(path)

                        value = path+'/ds'+str(index)+'.pkl'
                        pkfile = open(value,'wb')
                        pickle.dump(data, pkfile)
                        pkfile.close()

                        index = index + 1


                frame = [source_file+":"+ds_name, size, shape, dtype, value]
                self.sqlite_cur.execute("INSERT INTO CERN_DATASETS3 VALUES (?, ?, ?, ?, ?);",frame)

        f.close()

    def __stringSearch__(self, search_string, strict=False, selection="*") :

        """
        String search method
        search_string: type(str); string pattern to search
        strict: boolean ; whether to exactly match or partially
        selection: type(str); default -> * (selects all columns) or provide string with column names to return
        pass string of columns like this "Name, Dtype"

```

```
        returns list of matching rows
        """

        if strict is False :
            sql = "SELECT {C} FROM CERN_DATASETS3 WHERE Name LIKE '%{S}%'".format(C=selection, S=search_string)
        elif strict is True :
            sql = "SELECT {C} FROM CERN_DATASETS3 WHERE Name LIKE '{S}'".format(C=selection, S=search_string)

        self.sqlite_cur.execute(sql)
        results = self.sqlite_cur.fetchall()
        return [list(i) for i in results]

    def __conditionSwitcher__(self, condition) :

        """
        parsing conditions for __booleanSearch__
        """

        my_dict = {
            '-gt' : '>',
            '-lt' : '<',
            '-lteq' : '<=',
            '-gteq' : '>=',
            '-eq' : '='
        }

        return  my_dict.get(condition, "No selection")

    def __boolSearch__(self, search_string, condition, val, strict=False, selection="*") :

        """
        Method implements booleanSearch for SingluarValue Data
        search_string: type(str); dataset name
        strict: boolean; False -> partial, True -> full match; default-> False
        selection: default-> All columns selected, pass string of columns like this "Name, Dtype"
        condition: please __conditionSwitcher__ for supported conditions

        returns list of matching rows
        """

        try :
            val = float(val)
        except ValueError :
            print("val should be a number")
            pass

        condition = self.__conditionSwitcher__(condition)

        if condition is "No selection":
            raise ValueError("wrong condition selection")

        if strict is False :
            sql = "SELECT {S} FROM CERN_DATASETS3 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name LIKE '%{ss}%' and CAST(Data as REAL) {c} {v};".format(S=selection, ss=search_string, c=condition, v=val)
        elif strict is True :
            sql = "SELECT {S} FROM CERN_DATASETS3 WHERE Size = 1 AND Dtype NOT LIKE '|S%' AND Name LIKE '{ss}' and CAST(Data as REAL) {c} {v};".format(S=selection, ss=search_string, c=condition, v=val)

        self.sqlite_cur.execute(sql)
        results = self.sqlite_cur.fetchall()
        return [list(i) for i in results]

    def __getArrayData__(self, dataset, condition="-all", selection="*") :

        """
        Method to get dataset from pickle which are of type numpy array
        returns max,min or full dataset
        """
```

```
532
533         if condition not in ["-max", "-min", "-all"] :
534             raise ValueError("wrong condition selection")
535
536         data = pickle.load(open(dataset, 'rb'))
537
538         if condition == "-max" :
539             return np.max(data)
540         elif condition == "-min":
541             return np.min(data)
542         else:
543             return data
544
545 ##########################################################
546
547 ##########################################################
548
549 class preprocess(object) :
550
551     """
552     Adopted from question asked in https://stackoverflow.com/questions/18621513/python-insert-numpy
    -array-into-sqlite3-database/31312102#31312102
553     and read docs
554     """
555
556     def __init__(self, db_name) :
557         self.db_name = db_name
558
559     def __adapt_array__(self, array) :
560         out = io.BytesIO()
561         np.save(out, array)
562         out.seek(0)
563         return sqlite3.Binary(out.read())
564
565     def __convert_array__(self,s) :
566         out = io.BytesIO(s)
567         out.seek(0)
568         return np.load(out)
569
570     def __connect__(self, t) :
571         if t is not 1:
572             return sqlite3.connect(self.db_name)
573
574         sqlite3.register_adapter(np.ndarray, self.__adapt_array__)
575         sqlite3.register_converter("array", self.__convert_array__)
576         db = sqlite3.connect(self.db_name, detect_types=sqlite3.PARSE_DECLTYPES)
577         return db
578
579
580 ##########################################################################
581 #
582 # Created : 29th March 2019
583 # Author: balasuburamanyam[dot]evani[at]gmail[dot]com
584 # AWAKE GSOC 2019 Experiments
585 # Can be run on jupyter also
586 #
587 ##########################################################################
588 #!/usr/bin/env python3
589
590 '''
591 Main file calls library.py
592
593 type1 databse -> Only SQLite
594 type2 databse -> SQLite with HDF
595 type3 databse -> SQLite with pickle
596
597 '''
598
599 from library import *
600 import os
601 import time
602 import warnings          ## I added this because for some reason when i read data from pytables,
```

```
    it throws a lot of warnings,
warnings.filterwarnings('ignore') ## doesn't affect the reading of the h5 file though, searched online only explanation was maybe the
                ## original file was not created by pytables, i don't know about this.

db_name = '/home/balasb/Documents/CERN/test2/sqlite_db/testing.db'
t = 1
db = preprocess(db_name).__connect__(1) ## t {type: 1,2,3}
cdb = cern_database_type1(db.cursor()) ## change this to cern_database_type{1,2,3}

## when building the database

# source = "hdf_files"

# tick = time.time()
# for file in os.listdir(source):
#   h5f = source+"/"+file
#   cdb.__append__(h5f)
#   db.commit()
# tock = time.time()

# print("elapsed time for creating SQLite database: ", int(round(tock-tick)),"s")

t1_s = time.time()
res1 = cdb.__stringSearch__('BCTF',selection='*')
t1_e = time.time()

for i in range(len(res1)) :
  print(res1[i])

print("elapsed time for string query: ", round((t1_e-t1_s) * 1000),"ms")
print("--------------------------")

t2_s = time.time()
res2 = cdb.__boolSearch__('/AwakeEventData/TT41.BCTF.412340/Acquisition/totalCurrentPreferred','-gt',0.068)
t2_e = time.time()

for i in range(len(res2)) :
  print(res2[i])

print("elapsed time for boolean query: ", round((t2_e-t2_s) * 1000),"ms")
print("-----------------------------")

# ## if using cern_database_type1
t3_s = time.time()
res3 = cdb.__getArrayData__('1541962108935000000_167_838.h5:/AwakeEventData/TSG40.AWAKE-LASER-CORRELATOR/FileRead/dataImage',condition="-all")
t3_e = time.time()

print(res3)

print("elapsed time for getArray query: ", round((t3_e-t3_s) * 1000),"ms")

# if using cern_database_type2
# need to provide the path to dataset node, when you do a stringSearch query, we can get this path
# print("--------------------------------")

# t3_s = time.time()
# res3 = cdb.__getArrayData__('/1541962108935000000_167_838/ds176',condition="-all")
# t3_e = time.time()

# print(res3)
# print("elapsed time for getArray query: ", round((t3_e-t3_s) * 1000),"ms")


## if using cern_database_type3
## need to provide the path to dataset .pkl, when you do a stringSearch query, we can get this path

# print("-------------------------------")

# t3_s = time.time()
```

```
# res3 = cdb.__getArrayData__('./pickle_files/1541962108935000000_167_838/ds176.pkl',condition="-all")
# t3_e = time.time()

# print(res3)

# print("elapsed time for getArray query: ", round((t3_e-t3_s)*1000),"ms")
```

6.3 C

```
#!/usr/bin/env python3

#########################################################################
#
# Created : 3 April 2019
# Author: balasuburamanyam[dot]evani[at]gmail[dot]com
# AWAKE GSOC 2019 Experiments
#
#########################################################################

'''
Helper Library_2 for performing experiments
'''

'''
Loading essential libraries
'''

import tables as tb
import pickle
import pandas as pd
import os
import json
import time

from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk

#########################################################

class cern_database_type4(object) :

    def __init__(self) :

        self.es = Elasticsearch(timeout=100)

    def __create__(self):

        self.frame = []

        try :

            self.es.indices.create(index='cdb_testdemo')

        except :

            print("deleting previous index")
            self.es.indices.delete(index='cdb_testdemo',ignore=[400, 404])
            self.es.indices.create(index='cdb_testdemo')

            self.es.indices.put_mapping(
                  index="cdb_testdemo",
                  doc_type="datasets",
                  body={
                      "properties": {
                          "Name": {"type":"text"},
                          "Size": {"type": "text"},
                          "Shape": {"type": "text"},
                          "Dtype":{"type":"text"},
                          "Data":{"type":"text"}
                      }
                  }
```

```
            )

        #print(self.es.indices.get_mapping(index='cdb_testdemo5',doc_type='datasets'))

    def __append__(self, h5_file) :

        """
        Appends dataset to elasticsearch index as read from h5 file
        """

        print('collecting data from file:',h5_file)

        f = tb.open_file(h5_file, mode='r')
        source_file = h5_file.replace("hdf_files/","")
        index = 1

        for node in f :

            if isinstance(node, tb.group.Group) is False :

                name = str(node).split(" ")[0]
                data = f.get_node(node).read()
                size = data.size
                shape = str(data.shape)
                dtype = str(data.dtype)

                if size is 0 :
                    continue

                elif size is 1 :

                    # if bytes needs to be stored in string
                    #     if isinstance(data.tolist(), bytes) :
                    #         value = str(data.tolist(),'utf-8')
                    #     else :
                    #         value = str(data).lstrip('[').rstrip(']')

                    value = str(data).lstrip('[').rstrip(']')

                else :

                    path = './pickle_files2/'+source_file[:-3]+"/"
                    if not os.path.isdir(path) :
                        os.makedirs(path)

                    value = path+'ds'+str(index)+'.pkl'

                    pkfile = open(value,'wb')
                    pickle.dump(data, pkfile)

                    index = index + 1

                self.frame.append([source_file+":"+name, str(size), shape, dtype, value])

        f.close()

    def __shift__(self, source) :

        for file in os.listdir(source):

            h5f = source+"/"+file
            self.__append__(h5f)

        cols = ['Name','Size','Shape','Dtype','Data']
        df = pd.DataFrame(self.frame, columns=cols)
        df.to_csv("document.csv")

        print("storing to elasticsearch")

        documents = df.to_dict(orient='records')
        bulk(self.es, documents, index='cdb_testdemo', doc_type='datasets', raise_on_error=True)

```

```
135
136     def __stringSearch__(self, search_string) :
137
138         """
139         String search method
140         """
141
142         body = json.dumps({'query':{'match_phrase':{'Name': search_string}}})
143
144         r = self.es.search(index="cdb_testdemo", body=body, size=1000)
145         my_dict = []
146
147         for hits in r['hits']['hits']:
148             my_dict.append(hits['_source'])
149
150         return my_dict
151
152     def __boolSearch__(self, search_string, condition, val) :
153
154         """
155         Method implements booleanSearch for SingluarValue Data
156         """
157         try :
158             val = float(val)
159         except ValueError:
160             print("val should be a number")
161             pass
162
163         if condition not in ['gt','lt','gte','lte'] :
164             raise ValueError("not accepted condition")
165
166         body = json.dumps({'query':{'bool':{'must':{'match_phrase':{'Name': search_string}},"filter
    ":{"range":{"Data":{condition:val}}}}}})
167         r = self.es.search(index="cdb_testdemo", body=body, size=10)
168
169         my_dict = []
170
171         try:
172             for hits in r['hits']['hits']:
173                 my_dict.append(hits['_source'])
174         except:
175             print("no results found")
176
177         return my_dict
178
179
180     def __getArrayData__(self, dataset, condition="-all", selection="*") :
181
182             """
183             Method to get dataset from pickle which are of type numpy array
184             returns max,min or full dataset
185             """
186
187             if condition not in ["-max", "-min", "-all"] :
188                 raise ValueError("wrong condition selection")
189
190             data = pickle.load(open(dataset, 'rb'))
191
192             if condition == "-max" :
193                 return np.max(data)
194             elif condition == "-min":
195                 return np.min(data)
196             else:
197                 return data
198
199 ##############################################################
200
201
202 #!/usr/bin/env python3
203
204 ######################################################################
205 #
```

```
# Created : 4 April March 2019
# Author: balasuburamanyam[dot]evani[at]gmail[dot]com
# AWAKE GSOC 2019 Experiments
# Can be run on jupyter also
#
#########################################################################

'''
Main file calls library2.py

'''

from library2 import *
import os
import time
import warnings
warnings.filterwarnings('ignore')


cdb = cern_database_type4() ## change this to cern_database_type{1,2,3}

# # when building the database
# source = "hdf_files"

# tick = time.time()
# cdb.__create__()
# cdb.__shift__(source)
# tock = time.time()

# print("FINALLY DONE !")
# print("elapsed time for Shifting to Elasticsearch Search Engine: ", int(round(tock-tick)),"s")

t1_s = time.time()
res1 = cdb.__stringSearch__("BCTF") ## current implementation sensitive to what is entered
t1_e = time.time()

for i in range(len(res1)) :
  print(res1[i])

print("elapsed time for string query: ", round((t1_e-t1_s) * 1000),"ms")
print("-------------------------")

t2_s = time.time()
res2 = cdb.__boolSearch__('/AwakeEventData/TT41.BCTF.412340/Acquisition/totalCurrentPreferred','gt'
    ,0.068) ## case sensitive
t2_e = time.time()

for i in range(len(res2)) :
  print(res2[i])

print("elapsed time for boolean query: ", round((t2_e-t2_s) * 1000),"ms")
print("-------------------------------")

# ## need to provide the path to dataset .pkl, when you do a stringSearch query, we can get this
    path

print("---------------------------------")

t3_s = time.time()
res3 = cdb.__getArrayData__('./pickle_files2/1541962108935000000_167_838/ds176.pkl',condition="-all
    ")
t3_e = time.time()

print(res3)

print("elapsed time for getArray query: ", round((t3_e-t3_s)*1000),"ms")
```

6.4 D

```
#!/usr/bin/env python3

#########################################################################
```

```
#
# Created : 30th March 2019
# Author: balasuburamanyam[dot]evani[at]gmail[dot]com
# AWAKE GSOC 2019 Experiments
# CURVE FITTING EXPERIMENT
#
###############################################################################

import tables as tb
import numpy as np
import scipy.optimize as opt
from matplotlib import pyplot as plt
from scipy.signal import medfilt
import warnings

def func(xy, x0, y0, sigma, amp) :

    x, y = xy[0], xy[1]

    A = 1/(2*sigma**2 + 1e-6)
    res = amp*np.exp(-A*((x - x0)**2 + (y - y0)**2))
    return res

def rms(parameterTuple) :

  rms = np.sqrt(np.sum((y - func(x, *parameterTuple).ravel())**2))
  return rms

def generate_initial_guess(bounds_x0, bounds_y0, bounds_sigma, bounds_H) :

  parameterBounds = [bounds_x0, bounds_y0, bounds_sigma, bounds_H]
  result = opt.differential_evolution(rms, parameterBounds, seed = 10)
  return result.x

f = tb.open_file('./hdf_files/1541962108935000000_167_838.h5',mode='r')
data = f.get_node('/AwakeEventData/XMPP-STREAK/StreakImage/streakImageData').read()
w = f.get_node('/AwakeEventData/XMPP-STREAK/StreakImage/streakImageWidth').read()[0]
h = f.get_node('/AwakeEventData/XMPP-STREAK/StreakImage/streakImageHeight').read()[0]
f.close()

image = np.reshape(data,(h,w))
new_image = medfilt(image,15)

xx = np.arange(0, w, 1)
yy = np.arange(0, h, 1)
X,Y = np.meshgrid(xx, yy)

x = np.vstack((X.ravel(), Y.ravel()))
y = new_image.ravel()

initial_guess = generate_initial_guess([0, new_image.shape[0]], [0, new_image.shape[1]], [0, np.max
    (new_image)], [0, np.max(new_image)*2])

params, pconv = opt.curve_fit(func, x, y, p0 = initial_guess, method='lm', maxfev=1000)
predictions = func(x, *params)
RMS = np.sqrt(np.mean((new_image.ravel() - predictions)**2))

print("Predicted params : ", params)
print("Residual : ", RMS)

print(new_image.shape)
print(image.shape)
print(np.reshape(predictions,(h,w)).shape)

fig, ax = plt.subplots(2,2)
ax[0,0].imshow(image)
ax[0,1].imshow(new_image)
ax[1,0].imshow(np.reshape(predictions,(h,w)), cmap=plt.cm.jet ,interpolation='nearest', origin='
    lower')
ax[1,0].scatter(params[0], params[1], c = 'black', marker = 'x', s=100)
ax[1,0].contour(X, Y, np.reshape(predictions,(h,w)), 8, colors='w')
ax[1,1].imshow(new_image, cmap=plt.cm.jet ,interpolation='nearest', origin='lower')
```

```
ax[1,1].scatter(params[0], params[1], c = 'black', marker = 'x', s=100)
plt.show()
```

References

[1] ""Summary of Software Using HDF5 by Type"," [Online; accessed: 1 April 2019]. [Online]. Available: https://support.hdfgroup.org/products/hdf5_tools/SWSummarybyType.htm

[2] ""h5py user guide"," [Online; accessed: 1 April 2019]. [Online]. Available: http://docs.h5py.org/en/stable/quick.html

[3] ""pytables user guide"," [Online; accessed: 1 April 2019]. [Online]. Available: https://www.pytables.org/usersguide

[4] ""stakcoverflow question"," [Online; Asked: 24 March 2019]. [Online]. Available: https://stackoverflow.com/questions/55320372/how-to-construct-a-database-of-hdf5-files

[5] ""pandas"," [Online; accessed: 1 April 2019]. [Online]. Available: https://pandas.pydata.org/

[6] ""sqlite3"," [Online; Accessed: 2 April 2019]. [Online]. Available: https://docs.python.org/2/library/sqlite3.html

[7] ""elasticsearch python"," [Online; Accessed: 4 April 2019]. [Online]. Available: https://elasticsearch-py.readthedocs.io/en/master/

[8] ""sqlcipher"," [Online; Accessed: 2 April 2019]. [Online]. Available: https://github.com/sqlcipher/sqlcipher

[9] ""scipy"," [Online; Accessed: 2 April 2019]. [Online]. Available: https://www.scipy.org/

[10] ""scikit-image"," [Online; Accessed: 2 April 2019]. [Online]. Available: https://scikit-image.org/

[11] ""OpenCV"," [Online; Accessed: 2 April 2019]. [Online]. Available: https://github.com/opencv/opencv