

GPU-Accelerated Web Application for Metocean Descriptive Statistics

By

Muhammad Kamal Iqramuddin bin Kamal Arifin

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
Information and Communication Technology

MAY 2015

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
32610
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

GPU-Accelerated Web Application for Metocean Descriptive Statistics

By

Muhammad Kamal Iqramuddin bin Kamal Arifin

A project dissertation submitted to the
Information and Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
INFORMATION AND COMMUNICATION TECHNOLOGY

Approved by,

.....

(Dr Nordin bin Zakaria)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

.....

MUHAMMAD KAMAL IQRAMUDDIN KAMAL ARIFIN

Abstract

In today's real world application, many researchers and developers are using Graphic Processing Unit (GPU) to accelerate non-graphical application. Modern GPUs which are massively parallel general-purpose processors has a big advantages on big data analytics in terms of power efficiency, compute density and scalability. In oil and gas industries, metocean data is being generated, collected and analyzed at an unprecedented scale. Metocean data which is observed measurements of current, wave, sea level and meterological data are regularly collected by major oil and gas companies. This data, is usually collected by specialist companies and distributed to paying parties who will deploy their scientist and engineers to analyze and forecast information based on the information. The analyses of metocean data provide crucial information needed for operation or design work that has health, safety and environment (HSE) and economic consequences. Therefore, this paper proposed GPU-accelerated web applications for metocean descriptive statistics to improve the current CPU based implementation. Metocean descriptive statistics is the analysis of metocean data that helps provide important information required for operation that has health and safety as well as economic consequences. The application will utilize GPU to perform descriptive statistics for metocean data. The implementations of GPU for metocean descriptive statistics is expected to provide a better raw performance, better cost-performance ratios, and better energy performance ratios. The main objective of this project is to develop a GPU-accelerated application for metocean descriptive statistic and a web-based application that links to the GPU-accelerated application, besides demonstrate the capabilities of GPU in performing non-graphical calculation.

This research is important as it will help particular oil and gas companies in making a better, real-time business decision which has health, safety and environment (HSE) and economic consequences. In this research, we propose to have a GPU-accelerated backend for the web application that will be use by metocean user and engineer. The system will utilize GPU to perform a descriptive statistics for metocean data. The GPU-accelerated implementations of descriptive statistics is expected to have a

better raw performance, better cost-performance ratios, and better energy performance ratios.

ACKNOWLEDGEMENT

First and foremost, I would like to express my greatest to the Almighty Allah S.W.T for granting me endurance and persistence to complete this Final Year Project. My deep appreciation and thanks I owe to my parents and family members for their endless support and encouragement.

To my supervisor, Dr. Nordin bin Zakaria, thank you for your guidance and assistance. I am really grateful for his willingness to help me out and spend a lot of time guiding me towards this project completion. Your kindness, constructive feedback and guidance are very much appreciated.

I would also like to express my gratitude and thanks to Mr. Ade Wahyu Ramadhani for his cooperation and advice and to all the parties involved throughout the project completion.

Last but not least, thanks to all my friends and those who have helped me out directly and indirectly in completing this project.

Table of Contents

Abstract	iv
Acknowledgment	vi
List of Figures	ix
List of Tables	x
1.0 Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.2.1 How to improve the existing system of metocean data descriptive statistics system	2
1.3 Objectives	3
1.4 Scope of Study	3
1.5 Relevancy of the project	3
2.0 Literature Review	4
2.1 Introduction	4
2.2 What is GPU?	5
2.3 Graphic Processing Unit (GPU) vs Central Processing Unit (CPU)	5
2.4 Metocean Data	6
2.5 Descriptive Statistics for Metocean Data	8
2.6 The needs of GPU computing in data analysis	9
2.7 Comparative Study	10
3.0 Methodology	12
3.1 Data Gathering	12
3.1.1 Online Research	14
3.1.2 Academic Research	12
3.1.3 Empirical studies, personal experience and keen experience in GPU computing	14
3.2 Development Methodology	15

3.3 Requirement Definition	16
3.4 System Architecture	17
3.5 Integration and System Testing	18
3.6 Hardware and Software Required	19
3.6.1 Required Software	19
3.6.2 Required Hardware	20
3.7 Planned Methodology	20
3.8 Gantt chart	22
3.9 Key Milestone	24
4.0 Result and Discussion	25
4.1. Metocean User Requirement	25
4.2 Activity Diagram	33
4.3 Use Case Diagram of the System	34
4.4 Evaluation of ArrayFire	35
4.5 Code Evaluation	37
4.6 How System will Help the Operations	42
4.7 Performance Comparison of CPU and GPU in processing dataset	42
5.0 Conclusion	43
5.1 Conclusion	43
5.2 Recommendation	44
6.0 References	45

List of Figures

Figure 1: How GPU Acceleration works	4
Figure 2: Comparison of CPU and GPU core	6
Figure 3: Google ANN vs Stanford AI Lab ANN	9
Figure 4: Architectural difference between CPU and GPU	10
Figure 5: Computational Power GPU vs CPU	11
Figure 6: Iterative development model	13
Figure 7: System Architecture	15
Figure 8: Planned Methodology	19
Figure 9: Expected web application interface (List Metocean Grid Point)	23
Figure 10: Expected web application interface (Set Metocean Grid Point)	24
Figure 11: Expected web application interface (Descriptive Statistics)	25
Figure 12: Expected descriptive analysis output	26
Figure 13: Activity Diagram of the System	30
Figure 14: Use Case Diagram of the System	32
Figure 15: Programming interface (Visual Studio 2013 IDE)	33
Figure 16: Expected display interface for output using console	34
Figure 17: Performance Comparison of CPU and GPU	

List of Tables

Table 1: Gantt Chart 20

1.0 Introduction

1.1 Background

A graphics processing unit (GPU) is a specialized electronic circuit typically use to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. It is very efficient at manipulating computer graphics and image processing. Modern GPUs use most of their transistors to do calculations related to 3D computer graphics. Because most of these computations involve matrix and vector operations, more studies on the use of GPU for non-graphical calculation have been carried out by scientists and engineers. Many areas including molecular dynamics, quantum chemistry, numerical analytics, physics, etc. have been using GPU to implement algorithms as regard to it computational power. Their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. Video games and other graphical areas have long benefited from improved GPU performance. GPU can now afford to perform general numerical computing tasks like artificial neural networks.

In oil and gas industries, metocean data are collected and generated at an exceptional scale. Metocean is term use in oil and gas industries to describe the physical environment near a particular location associated with offshore platform. Metocean data includes measurements of current, wave, sea level and other meteorological data. The data are regularly collected on site by major oil and gas (O&G) companies. The analyses of metocean data provide essential information needed for operations or design work that has health and safety and economic consequences.

The processing of big data such as metocean data often require a lot of times and resources. Modern GPUs which are massively parallel general-purpose processors can provide extensive computational power in processing big data analytics. There are a few published works which used GPUs implementation in their system that shows notable performance gains. The utilization of GPU in big data analytics are becoming a trend in today's world and has been applied in many field of works. The application of GPU in descriptive statistic of metocean data is expected to assist companies in oil and gas industries to make a better and real-time business decision.

1.2 Problem Statement

1.2.1 How to improve the existing system of metocean data descriptive statistics system

The current implementation of the system is using CPU backend which can be greatly improved by using a GPU backend. The highly parallel structure of GPU makes it more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. This implementation may improve the performance of the system and assist the oil and gas companies to make a better and real-time business decision.

1.3 Objectives

- 1.3.1 Develop a GPU-accelerated application for descriptive statistics of metocean data.
- 1.3.2 Develop a web-based application that links to the GPU-accelerated application.
- 1.3.3 To demonstrate the capabilities of GPU in performing non-graphical calculation.

1.4 Scope of Study

This project aims at using GPU computing as an alternative to handle the descriptive analysis of metocean data by using ArrayFire, an open source software library for GPU computing in its implementation. The main purpose is to do the descriptive analysis faster and consequently making it suitable for real-time applications. The high parallel architecture of the GPU are highly compatible to process high volume data such as the metocean data in a parallel manner.

The implementation of GPU computing in analysis of metocean data is aimed to improve the performance as compared to CPU implementation.

1.5 Relevancy of the project

The relevancy of the project is this project can help in making better and real time business decision. Besides, the analyses of the metocean data which is an output of the system might have health and safety as well as economic consequences, will be greatly improved by the implementation of this project.

2.0 Literature Review

2.1. Introduction

Many researchers and programmers from different fields has started utilizing GPU (Graphic Processing Unit) to improve performance and maximize efficiency in field of work such as physics, signal and image processing, and visualization techniques (Owens et al. 2007). Applications which was optimized for GPUs are known as GPU-accelerated applications. GPU-accelerated application offers better application performance by offloading compute-intensive portions of the application to the GPU (NVIDIA Corporation, 2015). For example, Oil and Gas industry has also applying GPU to accelerate their work in seismic processing application such as Acceleware, Tsunami RTM, GeoStar Seismic Suite etc.

Traditionally, GPU only handles computation involve in graphics whereas computing in application was handled by CPU. Nowadays, with GPU-accelerated computing, CPU is use together with GPU to accelerate scientific, analytics, engineering, and consumer and enterprise applications. It offers higher performance by offloading compute-intensive part of the application to GPU while remaining of the code still runs on the CPU. From the user point of view, applications improved its performance and run significantly faster (NVIDIA Corporation, 2015). Figure 1 illustrates how GPU acceleration works, which show compute-intensive functions are handle by GPU while the rest of sequential CPU code are handle by CPU.

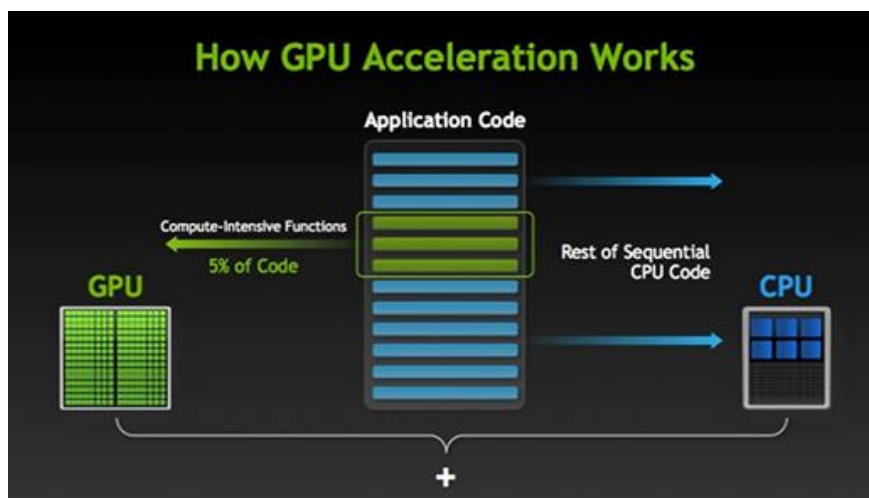


Figure 1: How GPU Acceleration works

2.2. What is GPU?

GPU which stands for Graphical Processing Unit is a specialized microprocessor that serves to accelerate the process of rendering two-dimensional charts or three dimensions. Back in 1970s, it was initially used in arcade system boards to accelerate the drawing of spite graphics for arcade games. (“Graphics Processing Unit”, 2015) In recent years, demanding applications with substantial parallelism increasingly utilize the parallel computing abilities of GPU to achieve higher performance and efficiency. With the use of GPU, application with long execution times which was thought to be impractical in the past feasible (Nickolls & Dally, 2010). GPU, over time has evolved from configurable graphic processor to programmable parallel processor better known as general purpose graphic processing unit (GPGPU) which can perform both graphics and computing applications. Non-graphical calculation that a GPU can perform are large matrix and vector operation, sequence matching, speech recognition, databases, sort and search operation etc.

2.3. Graphic Processing Unit (GPU) vs Central Processing Unit (CPU)

GPU has a parallel architecture consisting thousands of small cores designed for handling multiple task simultaneously while CPU consist of few cores optimized for sequential serial processing. CPU can perform complex operations on a single or few streams of data efficiently, however for handling many streams of data simultaneously GPU is more reliable (NVIDIA Corporation, 2015). To simply understand the difference between GPU and CPU, figure 2 shows the comparison of cores in GPU and CPU. CPU consists of a few cores optimized for sequential serial processing while GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple task simultaneously.

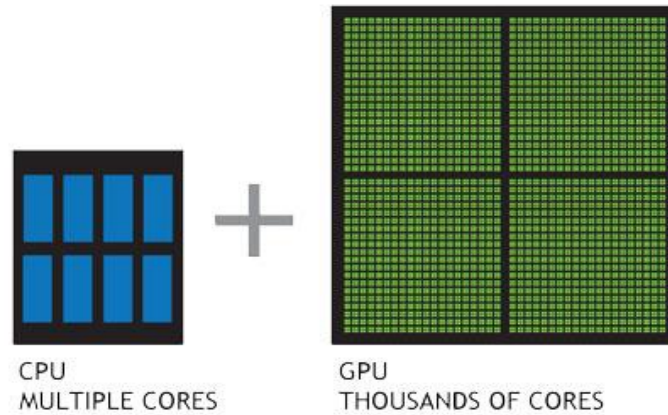


Figure 2: Comparison of CPU and GPU core

2.4. Metocean Data

Metocean is an abbreviation of the words "Meteorology" and "Oceanography". The term metocean data is used in offshore industry to describe physical environment data near an offshore platform.

“Metocean studies involve a lot of statistical analyses in the fields of data validation, determination of normal and extreme metocean conditions and data transformation.” (van Os, Caires & van Gent, 2011)

Metocean data is data collected by observing measurements of current, wave, sea level, temperature and wind. The data are regularly collected in situ by major oil and gas (O&G) companies. Essential information needed for operations or design work that has health and safety as well as economic effect can be provided by metocean analysis. Other than, avoiding high costs for installation and operation, metocean data also help to cut the construction costs with more accurate and less conventional design conditions. As there is no universal wave spectral model which can be applied to all storms in the world, it is a must to support offshore operational planning for the optimal design of offshore installations beside to ensure offshore safety and environmental protection. It is very important and necessary to conduct environmental study during the entire phases of offshore oil and gas exploration and production at the exposed installation area. (Vannak, Liew & Zheng Yew, 2013)

Metoccean data is usually collected by specialist companies. These data is then distributed to paying parties that is interested in obtaining the data. Based on the information obtained, these companies will deploy their scientist and engineers to analyze and forecast new information. Processing and analysis of large data set such as metoccean data, requires long hours on performance workstation and servers, especially when methods such as ARIMA and fuzzy logic is deployed involving large time periods and number of geographical locations.

Presently, there are more than 10, 000 offshore production structures worldwide. With the increase of world energy demand, the number will surely increase as oil and gas companies competing to explore areas to find new sources of hydrocarbon. To avoid an operation of determining new sites to be disrupted by any dangerous occurrence, regional sea state condition surveys are done before the activities. In deep water regions, due to extreme metoccean condition, offshore structures are often exposed to dangerous conditions. Wave forces has reportedly causing damage to offshore drilling platform. In general, the most crucial loads to be measured during the overall design of the offshore structure because the loads are applied to the structure from all directions. A hindcast database which contains important data such as wave heights, wave directions, current amplitude and direction spectral information as well as wind speed over a long period of time supplies the environmental data needed. Hindcast is application or tool used to forecast the phenomenon of previous/pre-years metoccean situations using a computer model based on historical events of wind-wave or atmospheric conditions. Therefore, it can be generated or modeled only after the real events have occurred (Vannak, Liew & Zheng Yew, 2013).

2.5. Descriptive Statistics for Metocean Data

Descriptive statistics is set of brief descriptive coefficients that summarizes a given data set, which can either be a representation of the entire population or a sample. The measures used to depict the information are measures of central tendency and measures of variability or dispersion. Measures of central tendency include the mean, median and mode, while measures of variability include the standard deviation, variance, the minimum and maximum value in the data set, kurtosis and skewness. Descriptive statistics give a valuable summary of the sample when performing empirical and analytical analysis. Although past information is useful in any analysis, the future possibility should also be consider. (“Descriptive Statistic”, 2015).

In designing, installation and running operations in a safe and efficient manner in oil and gas industries, metocean conditions is one of the most important factor. Extreme event related to the environmental loads can have a damaging effect to the offshore structures. However, the estimation of the metocean conditions and its extreme occurrences is complicated since interpretation of the information obtained must be validate to produce reliable environmental loads based on specified return period of the design. This data can be collected in many ways in which, one them is the actual standard measurement system obtained from instrumented buoy. However this data might not be accurate due to missing data or errors, thus it is not sufficient to depend entirely on the measured data alone. Not only that, to reduce errors in measuring extreme values of the environmental loads, measured data have to be observed for long period (Mayeetae, Liew, Abdullah, 2012). Hence, the standard practice is to use hindcast data to derive metocean design criteria if there are insufficient or missing measured data. (Vannak, Liew & Zheng Yew, 2013).

2.6. The needs of GPU computing in data analysis

Metocean data is a large volume of data. A very high computing power are needed in order to analyze the data efficiently. Nowadays, data scientist increasingly using GPUs for big data analytics to improve performance and make better, real-time business decision (NVIDIA Corporation, 2015).

For example, Nvidia partnered with research team at Stanford University has create the world's largest artificial neural network using GPU-accelerated servers. This artificial network is 6.5 times bigger than Google-created artificial neural network back in 2012. Diagram 3 illustrates the graph of parameters against number of servers of both network. The graph shows that Google have 1000 CPU-based servers which totals up to 16, 000 CPU cores that consist of 1.7 billion of parameters while Stanford AI Lab with 16 GPU-accelerated servers and consist of 11.2 billion of parameters. This shows that GPU-accelerated system capabilities is better in comparison to CPU-based system in processing large scale analytics (Parrish, 2013).

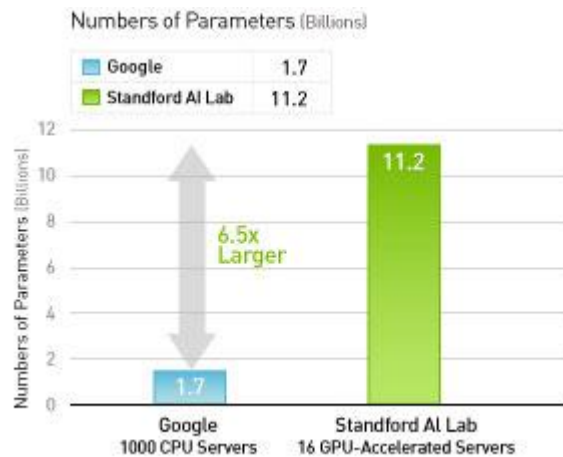


Figure 3: Google ANN vs Stanford AI Lab ANN

Studies by Ren Wu, Bin Zhang, and Meichun Hsu from HP Laboratories prove that their GPU-accelerated system can be 200-400 times faster than MineBench, their baseline benchmark, running on a single CPU core and about 50 - 88 times faster than their highly optimized CPU version running on a single core. Though using a large cluster of machines and specially designed hardware accelerators can offer equivalent performance, very large

cluster and expensive custom designed hardware is needed. The research also shows that GPU based approach gives best cost-performance ratio and offers better energy-performance ratio than cluster.

2.7. Comparative Study

We can compare the performance of CPU and GPU in term of architecture and computational power. CPU can perform complex operations on a single or few streams of data efficiently, however for handling many streams of data simultaneously GPU is more reliable. (NVIDIA Corporation, 2015) Diagram 4 and 5 show the comparison for CPU and GPU.

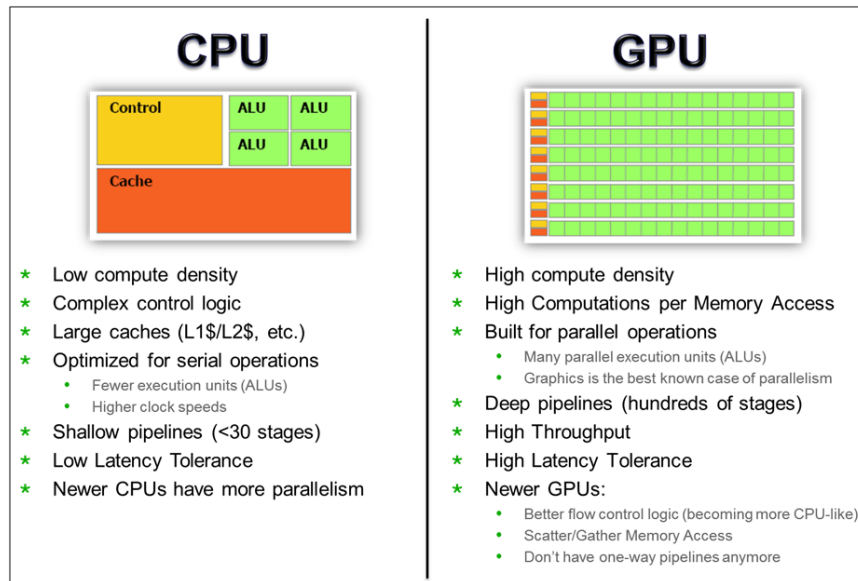


Figure 4: Architectural difference between CPU and GPU

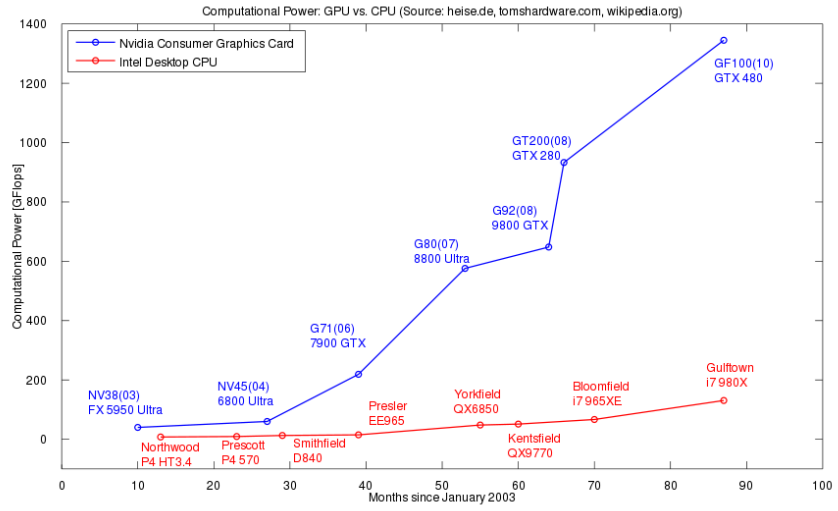


Figure 5: Computational Power GPU vs CPU

3.0 Methodology

This chapter covers all the methodology that will be used throughout the project. After defining the problems and objectives that need to be achieved in Chapter 1, the next step is conducting the literature review on publications, previous works and books relevant to the project. The time allocated for each section is sufficient in order to meet with the project dateline for each chapter. For the studies done by previous authors, most of the information is obtained from books and journals which can be found in the Information Resource Centre as well as websites.

3.1. Data Gathering

Data gathering is the process of collecting relevant data from research papers and information from the Internet for literature review and to be use in the project. These are the method and platform used to collect data and information for the project.

3.1.1. Online Research

Nvidia CUDA Zone (<https://developer.nvidia.com/get-started-parallel-computing>) and AMD OpenCL Zone (<http://developer.amd.com/tools-and-sdks/opencl-zone/>) provide information on GPU computing as well as Cuda and OpenCL. This pages also provide downloads for CUDA driver for Nvidia GPU and AMD OpenCL APP SDK for AMD GPU. ArrayFire Documentation (<http://www.arrayfire.com/docs/index.htm>) is the main source of reference to learn about ArrayFire. This page provide informations such as build instruction, tutorials, list of functions and example. ArrayFire website also provide forum to connect with the ArrayFire community and pick up acceleration tips and innovative best practices. It was recently make open-source and this forum is medium for experts and developers to share knowledge and information. Below are the question asked and answer by the experts from Arrayfire. Attach is the link to the forum discussion on topic of “Read CSV file”. The answer provided is very precise and really helpful for the development of the project.

https://groups.google.com/forum/#!msg/arrayfire-users/Nrwy080ShdM/U2FhA_ng_cMJ

Question:

Hi, I am trying to read in CSV file and then do some statistical studies on the data, but I can't seem to find any example on how to read in CSV file. Can someone help me please? If possible, I want to make the first row as the header. Thank you.

Answer:

Hi, I would suggest you to look at a CSV parser or write your own parser. You will need to parse the data, store it in a host side data structure and then copy it to ArrayFire. There is currently no way to load a file into a device side array directly.

Question:

Hi, I have created my own parser using standard C++ library. How do I store the data in the host and copy to ArrayFire? Thank you.

Answer:

You can use one of the copy constructors in ArrayFire. For example:

// Create a six-element array on the host

```
float hA[] = {0, 1, 2, 3, 4, 5};
```

// Which can be copied into an ArrayFire Array using the pointer copy

// constructor. Here we copy the data into a 2x3 matrix:

```
array A(2, 3, hA);
```

The full list of constructors can be found in the documentation:

http://arrayfire.com/docs/group__construct__mat.htm

3.1.2. Academic Research

Through IEEE websites and Google Scholars, researcher was able to get some research papers and literature review on similar topics that will be assessed. By reading and analyzing these research papers, researcher was able to get deeper understanding on GPU computing and help researcher to recognize issues and gap to be solved.

3.1.3. Empirical studies, personal experience and keen experience in GPU computing

This topic is new to the researcher, so this require researcher to learn the topic and the way to implement this area in real life through many sources such as books, journal, forums etc. By researching and learning from tutorials and other resources on GPU computing, researcher was able to have more understanding on the research area.

3.2. Development Methodology

In this project, iterative development model is chosen to be implemented. This model is combination of both iterative method and incremental build model. The iterative development model is a method of software development where the product is designed, implemented and tested incrementally until the product is finished. This is because during software development, there may be more than one iteration in progress at the same time. For example, initially this research was to be use with OpenCL, then after a few discussion ArrayFire was proposed as the medium instead. Overall software development methodology and software development process determines the relationship between iterations and increments. The main ideology of using this method is developing system through repeated cycles (iterative) and is smaller chunks at time (incremental), which will allow the developers to use the knowledge that has been collected during development of earlier version of the system. At each iteration, design modifications are made and new functional capabilities are added.

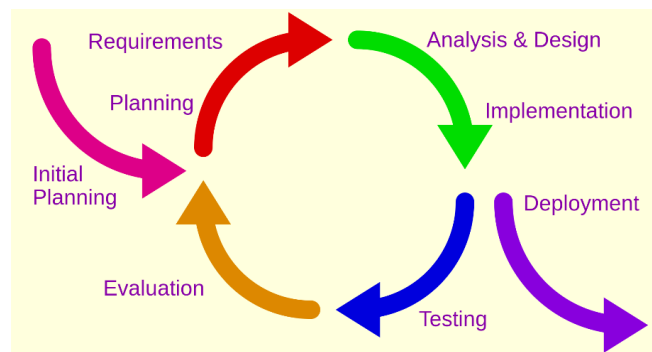


Figure 6: Iterative development model

3.3. Requirement Definition

Requirement definition would require researcher to list out any ideas of project and select a project topic as the Final Year project title. Then, researcher must identify the problem and objective of the project prior to the project title. After few discussions with FYP supervisor, researcher have chosen descriptive statistic for metocean data using GPU platform as the topic. Study on project background ad preliminary research study are crucial element in this phase to gather all the information and study the trend within the scope of GPU computing. Study on project background and preliminary research study would require researcher to find research papers and other information related to the project title. Once all the requirements are collected, the next step will be drafting the literature review.

The goal of this project is to develop a GPU- accelerated web-based application that able to perform descriptive statistic of metocean data. In order to meet the objective, some research and discussion on existing application were conducted to further understand the requirement of the project. The project will be conducted with two different operating system which are Windows 8 and Linux Ubuntu. The software use in this project is Visual Studio 2013 for Windows 8 with ArrayFire libraries. Defining of such necessary requirement could ease the task in later stage.

3.4. System Architecture

During the design phase, system architecture is established to identify and describe the fundamental structure abstraction as well as process flow of the whole system. The system architecture is the conceptual model that defines the structure, behavior and flow of the system. Figure below show a high-level architectural schematic diagram of the system. The primary actors in the figure are: the metocean engineer, researcher, and metocean user. Metocean engineer are engineers and specialist from the Metocean department, the researcher is the researcher that develops the system and metocean user refers to operational users - the end-users who makes use of the metocean data and its analysis in their routine and design operations. Both Metocean engineer and metocean user work with the system via a web frontend, which is web portal. The web portal interfaces to an ArrayFire Backend, a collection of ArrayFire function that perform metocean data descriptive statistic. The researcher are responsible for adding functions to the ArrayFire Backend. The addition has to be manually, involving source code adaptation.

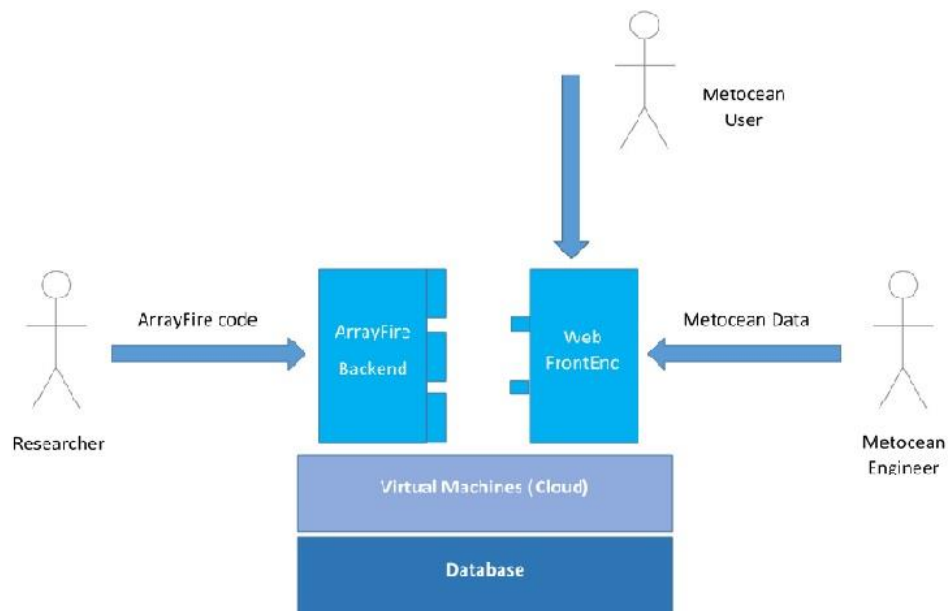


Figure 7: System Architecture

3.5. Integration and System Testing

This project require integration module between hardware, software and database. This project is created separately and the output is used in the next module. The module is separated because this module is created separately in every iteration. This project is incrementally integrate to make it a complete project which is a web-based application using GPU platform.

First, this system need to be integrate between software and hardware which consist of Visual Studio 2013 and AMD Radeon HD 7470 to enable GPU computing using OpenCL. This integration was possible by using AMD OpenCL APP SDK for AMD GPU. OpenCL are used to give an application access to a graphics processing unit for non-graphical computing.

Secondly, integration between the applications programmed using Visual Studio with database in which the metocean data is stored. This integration will allow the application to retrieve the metocean data from the database which then will be used by the application. The retrieved data will be used by the application to perform descriptive statistic of the data.

Third, the integration between GPU backend with the web frontend which is the web portal that act as the interface to the metocean user and metocean engineer. The integration will allow metocean user and metocean engineer to use the function and modules specified at the GPU backend.

Finally, after the integration of all components are done, metocean user and metocean engineer can use the web interface to perform descriptive statistics of the metocean data.

3.6. Hardware and Software Required

3.6.1. Required Software

- ArrayFire libraries
 - ArrayFire is a blazing fast software library for GPU computing. Its easy-to-use API and array-based function set make GPU programming simple. A few lines of code in ArrayFire can replace dozens of lines of raw GPU code, saving you valuable time and lowering development cost.



- ArrayFire utility
 - Sort
 - Mean
 - Variance
 - Standard Deviation
 - Covariance
 - Median
- Visual Studio 2013
 - Visual Studio 2013 is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger.



3.6.2. Required Hardware

- OpenCL or Cuda enabled Graphic Processing Unit. But in this project we are using AMD Radeon HD 7470, which is an OpenCL enabled GPU.



GPU Name	Caicos
Transistor	370 million
Interfaces	PCIe 2.0 x 16
Clock Speed	750 MHz
Memory Clock Speed	900 MHz
Effective Memory Clock Speed	3600 MHz
Memory Bus	64 bit
Memory Size	1024 MB
Memory Type	GDDR5
Memory Bandwidth	28.8 GB/s

OpenCL	1.2
Shading Units	160
Texture Mapping Units	8
Render Output Units	4
Compute Unit	2
Pixel Rate	3.00 GPixel/s
Texture Rate	6.00 GTexel/s
Floating-Point Performance	240.0 GFLOPS

3.7. Planned Methodology

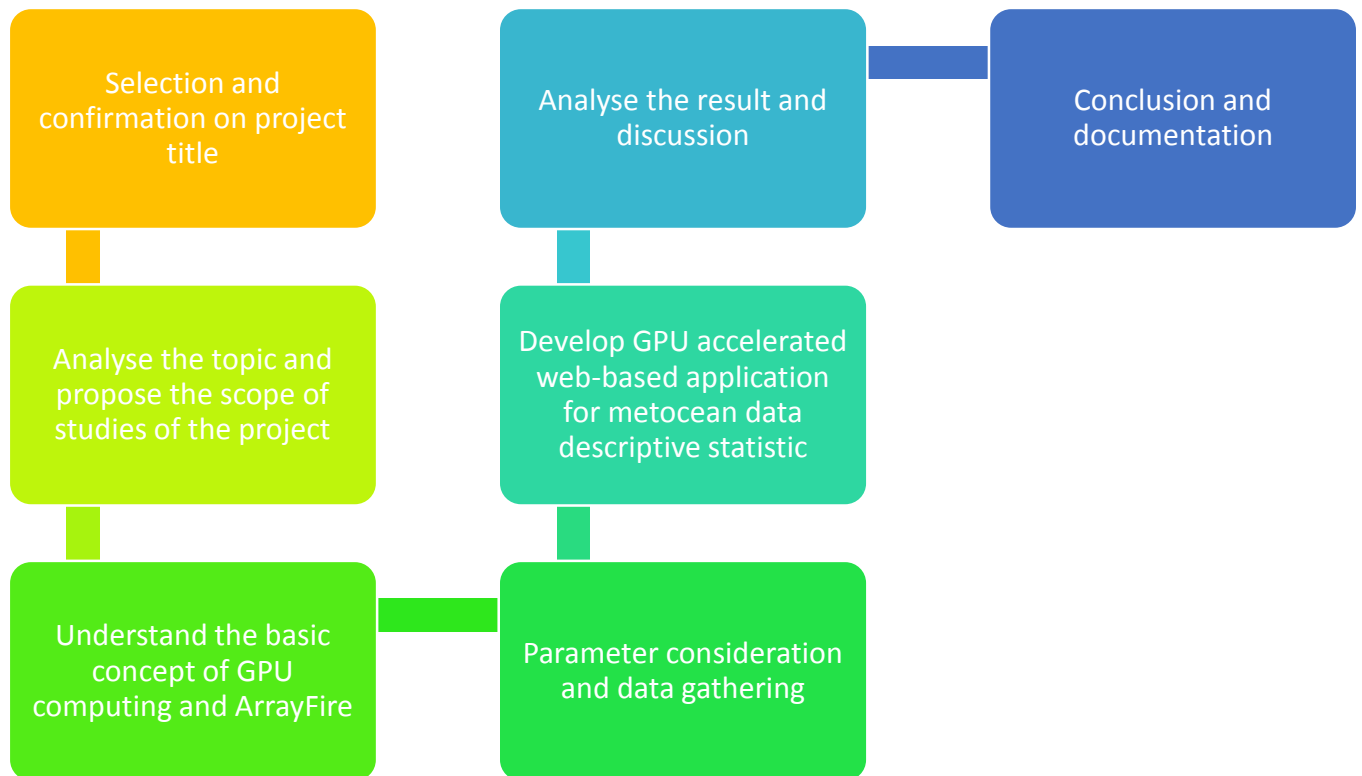


Figure 8: Planned Methodology

3.8. Gantt chart

Activity/Period	Week																											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Planning																												
Discussion with SV																												
Selection of project topic																												
Identify the problem																												
Define objectives of project																												
Study on project background																												
Preliminary research work																												
Literature review																												
Evaluate OpenCL																												
Evaluate ArrayFire																												
Installing ArrayFire																												
Familiarise with ArrayFire Utility																												
Analysis and Design																												
Draft the system flow																												
Design system component																												
Design user interface																												
Coding ArrayFire application																												
Implementation																												
Implement ArrayFire for Metocean Descriptive Statistics																												
Implement Minimum/Maximum function																												
Implement Median function																												
Implement Range, Mean function																												
Implement Standard Deviation, Standard Error Mean and Variance Function																												

3.9. Key Milestone

FYP1

Milestones	Week
Proposal	Week 3
Interim Report	Week 12
Proposal Defence	Week 13

FYP2

Milestones	Week
Progress Report	Week 7
Pre-SEDEX	Week 10
VIVA	Week 13
Final Report	Week 14

4.0 Result and Discussion

4.1. Metocean User Requirement

The figures below show the snapshots of the web application interface that will be use by metocean user and metocean engineer.

The screenshot shows a web browser window with the URL `hpc.utp.edu.my/metocean_v1/cpanel/index.php?page=analysis2`. The page title is "SEAMOME" and the user is logged in as "Metocean Specialist". The main heading is "Metocean Grid Point". Below this, there is a section titled "Point Coordinates In Decimal" with the following form fields:

- Region:
- Latitude:
- Longitude:
- Range (Km):

Below these fields are two buttons: "List Points" and "Reset".

Below the buttons is a table for DMS coordinates:

	Deg	Min	Sec	
Lat	<input type="text" value="20"/>	<input type="text" value="20"/>	<input type="text" value="20"/>	<input type="text" value="N"/>
Long	<input type="text" value="20"/>	<input type="text" value="20"/>	<input type="text" value="20"/>	<input type="text" value="E"/>

Below the table is a button labeled "Convert DMS to Decimal".

Figure 9: Expected web application interface (List Metocean Grid Point)

On this page is the metocean user will define the criteria for the grid points. First, the user will choose the region on which the database is residing. The list of regions area Sea Fine, Sea Coarse, Ma Kasar and Java. Then, the user is required to define the latitude and longitude (latlong) of the location and range from the latlong defined. “List points” button will bring the user to next page, while “Reset” button will set the fields of each item to the preconfigure values.

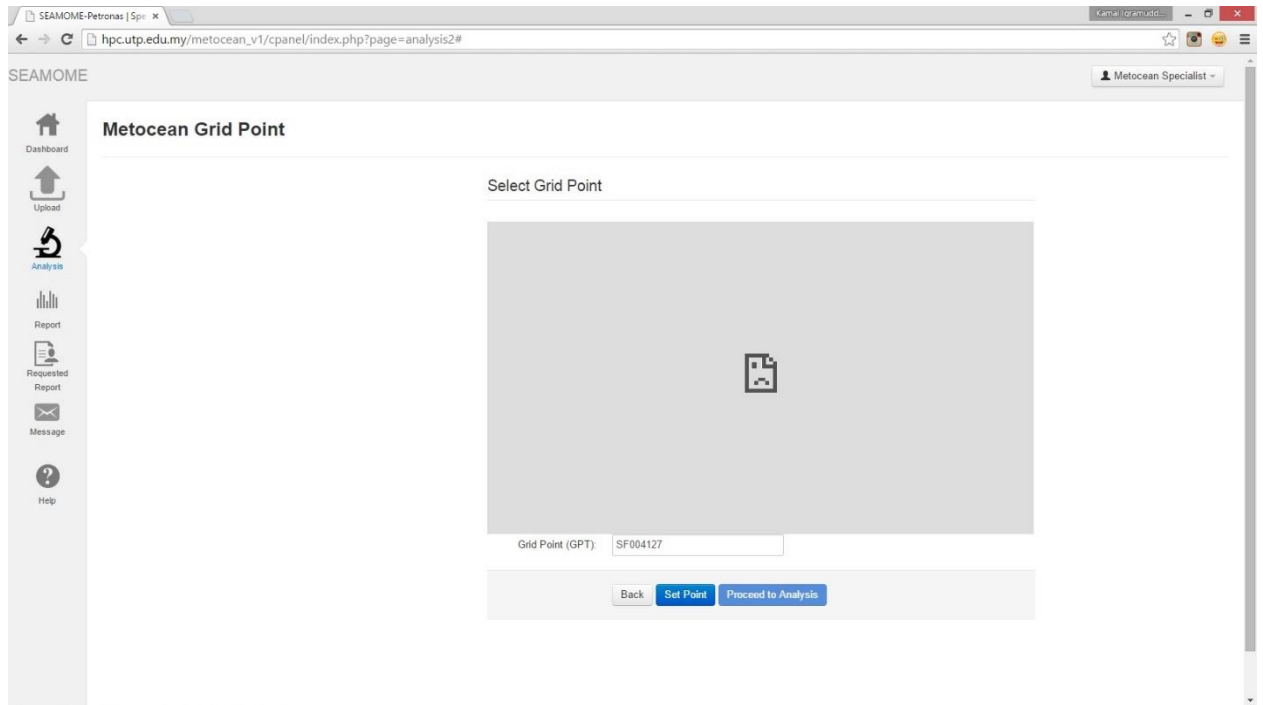


Figure 10: Expected web application interface (Set Metocean Grid Point)

This page will show the list of grid point which fulfill the criteria describe in the previous page. The metocean user will choose a grid point from the list and copy the value of the grid point into the grid point text box. “Set Point” button will read the value in the text box and set the value as grid point. “Proceed to Analysis” will bring the user to Analysis page while “Back” button will bring user to the previous page.

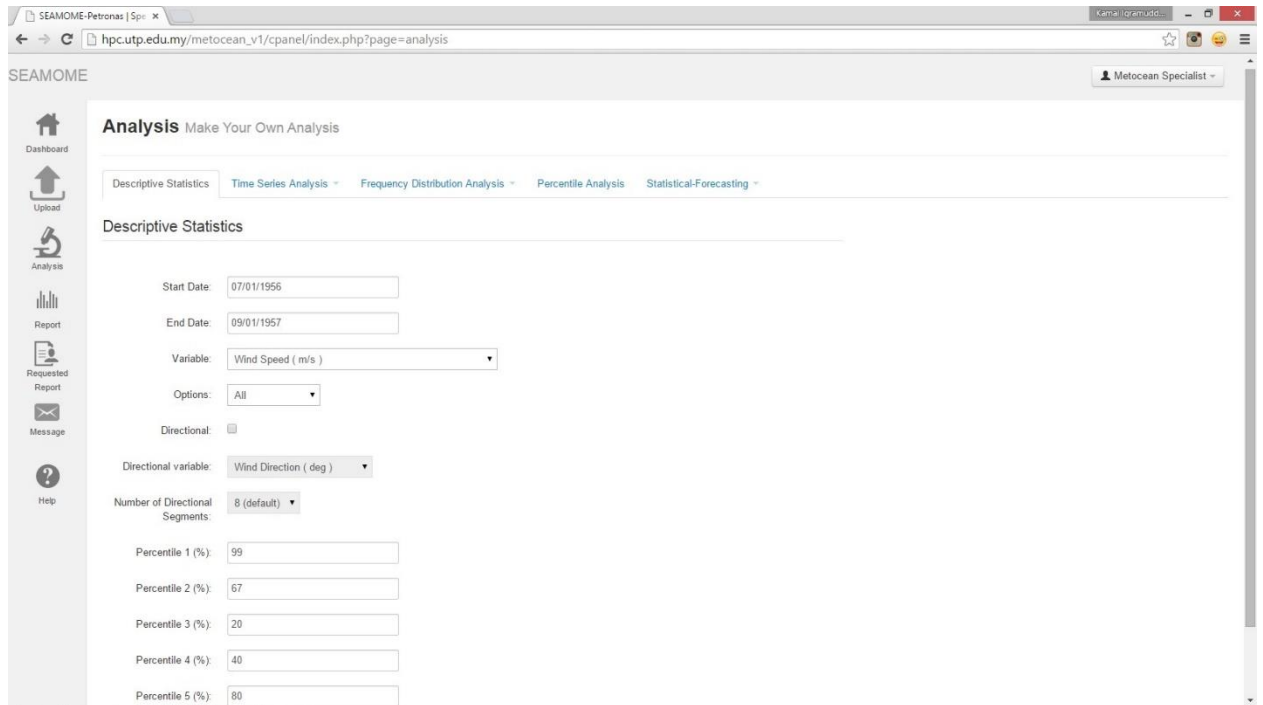


Figure 11: Expected web application interface (Descriptive Statistics)

On the Analysis page, there are five tabs which are Descriptive Statistics tabs, Time Series Analysis tabs, Frequency Distribution Analysis tabs and Statistical-Forecasting tabs. This project focus on the Descriptive Statistics. In this page, user will define the start and end date, variable, options for displaying the result, directional variable (if required), number of directional segment and percentiles. The list of variables are Wind Speed (m/s), Total Variance of Total Spectrum (m^2), Peak Period (sec), Sea Total Variance (m^2), Sea Peak Spectral Period (sec), First Spectral Moment of Total Spectrum (m^2/s), Second Spectral Moment of Total Spectrum (m^2/s^2), Significant Wave Height (m), Angular Spreading Function and In-Line Variance Ratio. Options for displaying results are All, Monthly, Yearly and Yearly Months. The user require to check the “Directional” check box button if the user want to include the Directional Variable in the output. The list of Directional Variables are Wind Direction (deg), Wave Direction (deg), Sea Direction (deg), Swell Direction (deg) and Dominant (deg). The list of number of directional segments are 8, 12 and 16. By default, the value of number of directional segment is 8. The user also require to define the percentiles. “Analyze button” will instruct the application to perform the

analysis as well as generate and output the result, while “Reset” button will set all fields to preconfigure value.

Export ALL data

Frequency of Occurrence - Grid Point: SF004127.txt
 Date Ranges: 1956-07-01 00:00:00 to 1957-09-01 00:00:00
 Vertical variable: Wind Speed (m/s)
 Directional variable: Wind Direction (deg)
 Number of observations for Directional ALL is 10249

Direction	Frequency	Minimum	Maximum	Median	Range	Mean	Standard Deviation	Standard Error Mean	Variance	Kurtosis	Skewness	99%	67%	20%	40%	80%
N	2359	1.14	13.08	5.47	11.94	5.82772	2.32833	0.0479381	5.42112	2.25039	0.329004	10.9626	6.8786	3.69	4.88	8.05
NE	1878	1.36	12.26	6.25	10.9	6.28497	2.04576	0.0472071	4.18514	2.58812	0.00232277	10.7884	7.23	4.48	5.848	8.11
E	144	0.68	7.9	3.395	7.22	3.68951	1.56167	0.130139	2.43881	2.8441	0.639866	7.6755	4.0481	2.306	3.056	5.284
SE	101	0.38	7.29	2.74	6.91	3.13921	1.40528	0.139831	1.97482	4.28464	1.23768	7.14	3.28	2.12	2.52	4.12
S	221	0.96	5.82	2.99	4.86	2.98507	0.944352	0.063524	0.891801	3.01777	0.30004	5.258	3.34	2.24	2.76	3.66
SW	1753	0.76	11.12	4.33	10.36	4.51484	1.73017	0.0413235	2.99348	2.96269	0.466295	8.976	5.2	2.96	3.9	5.98
W	2505	0.98	9.14	4.22	8.16	4.2973	1.42627	0.0284969	2.03424	2.79513	0.286021	8.0576	4.92	2.98	3.85	5.56
NW	1288	0.89	9.31	3.8	8.42	4.00664	1.4558	0.0405643	2.11935	3.19224	0.714824	8.0565	4.55	2.7	3.38	5.2

Figure 12: Expected descriptive analysis output

The output will show the grid point and the descriptive analysis result. The result include frequency, minimum, maximum, median, range, mean, standard deviation, standard error mean, variance, kurtosis and skewness.

- Minimum
 - The smallest value in the data set.
- Maximum
 - The largest value in the data set.
- Median
 - The sample median is in the middle of the data: at least half the observations are less than or equal to it, and at least half are greater than or equal to it. To calculate the median, the data must be sort from smallest to largest values. If N is odd, the sample median is the value in the middle. If N is even, the sample median is the average of the two middle values. For

example, when $N = 5$ and you have data $x_1, x_2, x_3, x_4,$ and x_5 , the median
= x_3 . When $N = 6$ and you have ordered data $x_1, x_2, x_3, x_4, x_5,$ and x_6 :

- $\text{median} = \frac{x_3 + x_4}{2}$
- where x_3 and x_4 are the third and fourth observations.

- Range
 - The range is the difference between the largest and smallest data value in the data set.
 - $R = \text{Maximum} - \text{Minimum}$
- Mean
 - Mean is the sum of all observations divided by the number of (nonmissing) observations.
 - The formula for mean is

$$\bar{x} = \frac{\sum_{i=1}^N x_i}{N}$$

- Notation

Term	Description
x_i	i^{th} observation
N	number of nonmissing observations

- Standard deviation
 - Standard standard deviation provides a measure of the spread of the data.
 - The formula for standard deviation is

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{N - 1}}$$

- Notation

Term	Description
x_i	i^{th} observation
\bar{x}	mean of the observations
N	number of nonmissing observations

- Standard error mean
 - The standard error of the mean is calculated as the standard deviation divided by the square root of the sample size.
 - The formula for standard error mean is

$$SE \text{ Mean} = \frac{s}{\sqrt{N}}$$
 - Notation

Term	Description
------	-------------

s	standard deviation of the sample
-----	----------------------------------

N	number of nonmissing observations
-----	-----------------------------------

- Variance
 - Variance is the measure of how far the data are spread from the mean.
 - Formula

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{N - 1} = \frac{\sum x_i^2 - (N \times \bar{x})^2 / N}{N - 1}$$

- Notation

Term	Description
------	-------------

x_i	i^{th} observation
-------	-----------------------------

\bar{x}	mean of the observations
-----------	--------------------------

N	number of nonmissing observations
-----	-----------------------------------

- Kurtosis
 - Kurtosis is the measure of how different a distribution is from the normal distribution. A positive value usually indicates that the distribution has a sharper peak than the normal distribution. A negative value indicates that the distribution has a flatter peak than the normal distribution.

- Formula

$$b_2 = \frac{N(N+1)}{(N-1)(N-2)(N-3)} \sum \left[\frac{x_i - \bar{x}}{s} \right]^4 - \frac{3(N-1)^2}{(N-2)(N-3)}$$

- Notation

Term	Description
x_i	i^{th} observation
\bar{x}	mean of the observations
N	number of nonmissing observations
s	standard deviation of the sample

- Skewness

- Skewness is a measure of asymmetry. A negative value indicates skewness to the left, and a positive value indicates skewness to the right. A zero value does not necessarily indicate symmetry.

- Formula

$$b_1 = \frac{N}{(N-1)(N-2)} \sum \left[\frac{x_i - \bar{x}}{s} \right]^3$$

- Notation

Term	Description
x_i	i^{th} observation
\bar{x}	mean of the observations
N	number of nonmissing observations
s	standard deviation of the sample

4.2. Activity Diagram

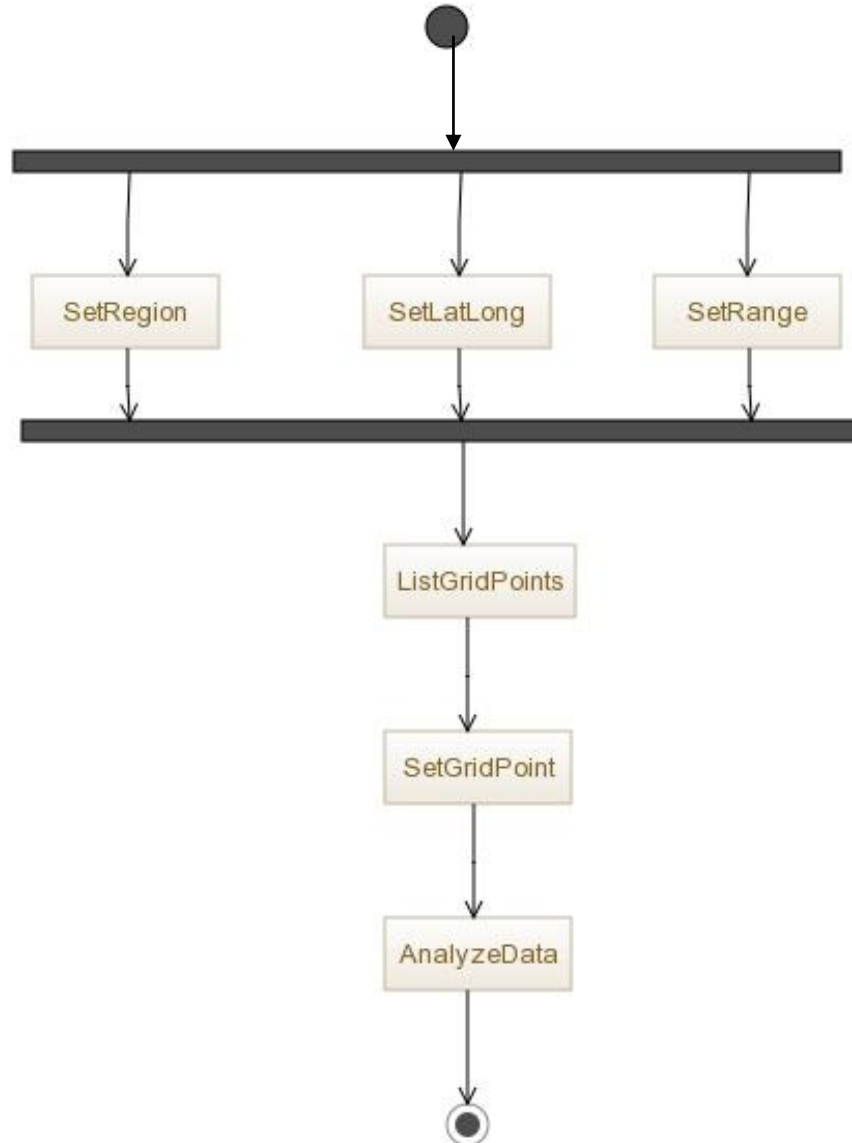


Figure 13: Activity Diagram of the System

First, user need to define Region, Latitude and Longitude (LatLong) and the range from the LatLong. Then, the system will generate a list of grid points based on the criteria defined by user. User will select a grid point from the list and set the grid point. Then, the system will analyze the data based on the user input.

4.3. Use Case Diagram of the System

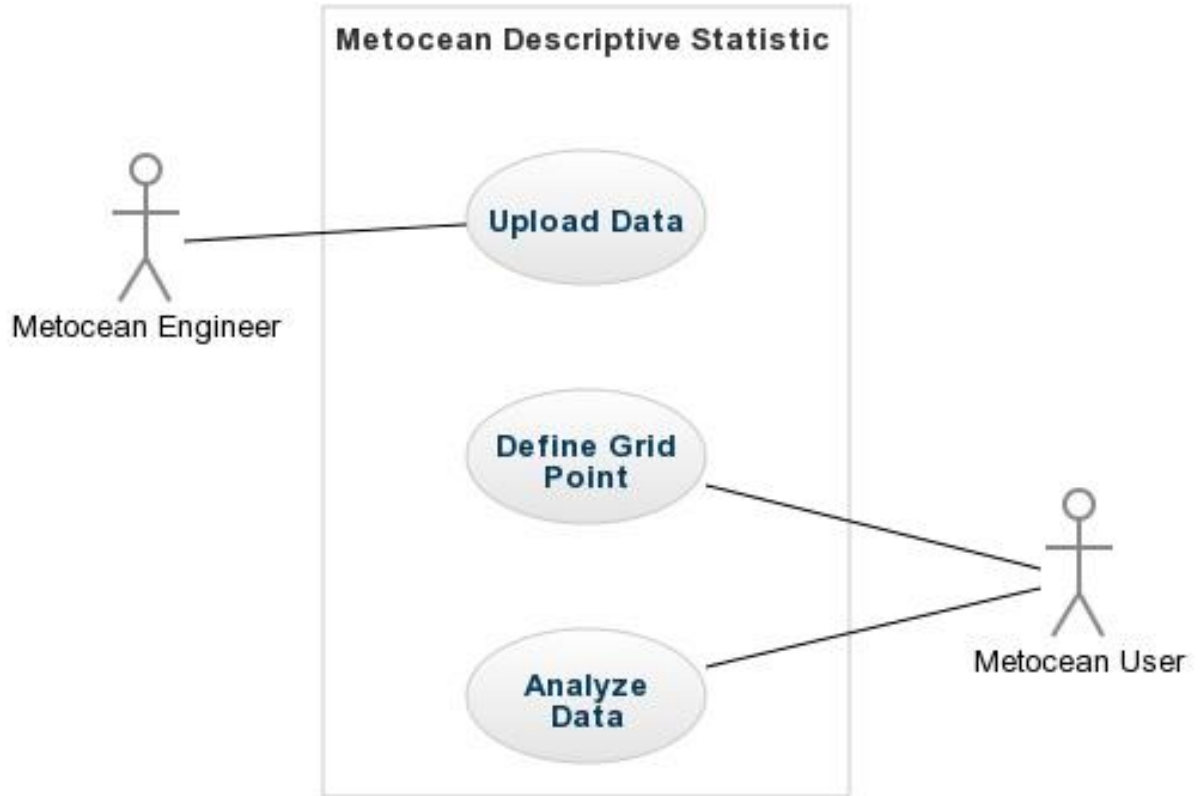
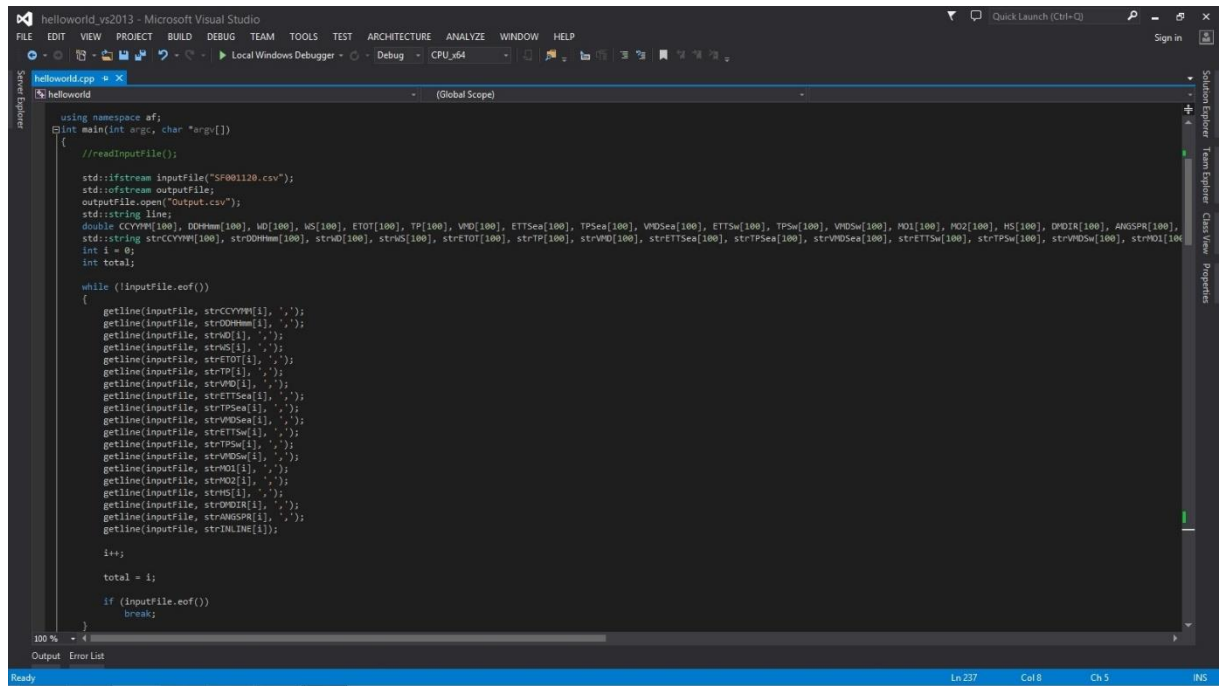


Figure 14: Use Case Diagram of the System

Metocean engineer responsibility is to upload the metocean data into the system database. Metocean user will define the grid point required and analyze the data.

4.4. Evaluation of ArrayFire

The programming interface in which the researcher will write the ArrayFire code for the system is Visual Studio 2013 IDE. Figure below show the snapshots of the programming interface.



```
using namespace af;
int main(int argc, char *argv[])
{
    //readInputFile();

    ifstream inputFile("SF001120.csv");
    ofstream outputFile;
    outputFile.open("Output.csv");
    string line;
    double CCYMM[100], DDHMM[100], WD[100], NS[100], ETOT[100], TP[100], WDI[100], ETTSea[100], TPSea[100], WDSw[100], ETTsw[100], TPSw[100], WDSw[100], MOI[100], MOIsw[100], HS[100], DPDIR[100], ANSPR[100],
    strCCYMM[100], strDDHMM[100], strWD[100], strNS[100], strETOT[100], strTP[100], strWDI[100], strETTSea[100], strTPSea[100], strWDSw[100], strETTsw[100], strTPSw[100], strWDSw[100], strMOI[100],
    int i = 0;
    int total;

    while (!inputFile.eof())
    {
        getline(inputFile, strCCYMM[i], ',');
        getline(inputFile, strDDHMM[i], ',');
        getline(inputFile, strWD[i], ',');
        getline(inputFile, strNS[i], ',');
        getline(inputFile, strETOT[i], ',');
        getline(inputFile, strTP[i], ',');
        getline(inputFile, strWDI[i], ',');
        getline(inputFile, strETTSea[i], ',');
        getline(inputFile, strTPSea[i], ',');
        getline(inputFile, strWDSw[i], ',');
        getline(inputFile, strETTsw[i], ',');
        getline(inputFile, strTPSw[i], ',');
        getline(inputFile, strWDSw[i], ',');
        getline(inputFile, strMOI[i], ',');
        getline(inputFile, strMOIsw[i], ',');
        getline(inputFile, strHS[i], ',');
        getline(inputFile, strDPDIR[i], ',');
        getline(inputFile, strANSPR[i], ',');
        getline(inputFile, strINLINE[i]);

        i++;

        total = i;

        if (inputFile.eof())
            break;
    }
}
```

Figure 15: Programming interface (Visual Studio 2013 IDE)

4.5. Code evaluation

```
#include <arrayfire.h>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <time.h>

void readInputFile(void);

time_t start = time(0);

const int numPatterns = 400000;
double CCYMM[numPatterns], DDHMM[numPatterns], WD[numPatterns], WS[numPatterns], ETOT[numPatterns],
TP[numPatterns], VMD[numPatterns], ETTSea[numPatterns], TPSea[numPatterns], VMDSea[numPatterns],
ETTSw[numPatterns], TPSw[numPatterns], VMDSw[numPatterns], MO1[numPatterns], MO2[numPatterns],
HS[numPatterns], DMDIR[numPatterns], ANGSPR[numPatterns], INLINE[numPatterns];

using namespace af;
int main(int argc, char *argv[])
{
    readInputFile(); // read data from file

    array A = array(numPatterns, 1, CCYMM); //creation of array for CCYMM
    array B = array(numPatterns, 1, DDHMM); //creation of array for DDHMM
    array C = array(numPatterns, 1, WD); //creation of array for WD
    array D = array(numPatterns, 1, WS); //creation of array for WS
    array E = array(numPatterns, 1, ETOT); //creation of array for ETOT
    array F = array(numPatterns, 1, TP); //creation of array for TP
    array G = array(numPatterns, 1, VMD); //creation of array for VMD
    array H = array(numPatterns, 1, ETTSea); //creation of array for ETTSea
    array I = array(numPatterns, 1, TPSea); //creation of array for TPSea
    array J = array(numPatterns, 1, VMDSea); //creation of array for VMDSea
    array K = array(numPatterns, 1, ETTSw); //creation of array for ETTSw
    array L = array(numPatterns, 1, TPSw); //creation of array for TPSw
    array M = array(numPatterns, 1, VMDSw); //creation of array for VMDSw
    array N = array(numPatterns, 1, MO1); //creation of array for MO1
    array O = array(numPatterns, 1, MO2); //creation of array for MO2
    array P = array(numPatterns, 1, HS); //creation of array for HS
    array Q = array(numPatterns, 1, DMDIR); //creation of array for DMDIR
    array R = array(numPatterns, 1, ANGSPR); //creation of array for ANGSPR
    array S = array(numPatterns, 1, INLINE); //creation of array for INLINE

    af_print(mean(A)); //calculate and print mean for CCYMM
    af_print(mean(B)); //calculate and print mean for DDHMM
    af_print(mean(C)); //calculate and print mean for WD
    af_print(mean(D)); //calculate and print mean for WS
    af_print(mean(E)); //calculate and print mean for ETOT
    af_print(mean(F)); //calculate and print mean for TP
    af_print(mean(G)); //calculate and print mean for VMD
    af_print(mean(H)); //calculate and print mean for ETTSea
    af_print(mean(I)); //calculate and print mean for TPSea
    af_print(mean(J)); //calculate and print mean for VMDSea
    af_print(mean(K)); //calculate and print mean for ETTSw
    af_print(mean(L)); //calculate and print mean for TPSw
    af_print(mean(M)); //calculate and print mean for VMDSw
    af_print(mean(N)); //calculate and print mean for MO1
    af_print(mean(O)); //calculate and print mean for MO2
```

```

af_print(mean(P)); //calculate and print mean for HS
af_print(mean(Q)); //calculate and print mean for DMDIR
af_print(mean(R)); //calculate and print mean for ANGSPR
af_print(mean(S)); //calculate and print mean for INLINE

std::cout << std::endl;

af_print(median(A)); //calculate and print median for CCYMMM
af_print(median(B)); //calculate and print median for DDHHmm
af_print(median(C)); //calculate and print median for WD
af_print(median(D)); //calculate and print median for WS
af_print(median(E)); //calculate and print median for ETOT
af_print(median(F)); //calculate and print median for TP
af_print(median(G)); //calculate and print median for VMD
af_print(median(H)); //calculate and print median for ETTSea
af_print(median(I)); //calculate and print median for TPSea
af_print(median(J)); //calculate and print median for VMDSea
af_print(median(K)); //calculate and print median for ETTsw
af_print(median(L)); //calculate and print median for TPSw
af_print(median(M)); //calculate and print median for VMDSw
af_print(median(N)); //calculate and print median for M01
af_print(median(O)); //calculate and print median for M02
af_print(median(P)); //calculate and print median for HS
af_print(median(Q)); //calculate and print median for DMDIR
af_print(median(R)); //calculate and print median for ANGSPR
af_print(median(S)); //calculate and print median for INLINE

std::cout << std::endl;

af_print(stdev(A)); //calculate and print standard deviation for CCYMMM
af_print(stdev(B)); //calculate and print standard deviation for DDHHmm
af_print(stdev(C)); //calculate and print standard deviation for WD
af_print(stdev(D)); //calculate and print standard deviation for WS
af_print(stdev(E)); //calculate and print standard deviation for ETOT
af_print(stdev(F)); //calculate and print standard deviation for TP
af_print(stdev(G)); //calculate and print standard deviation for VMD
af_print(stdev(H)); //calculate and print standard deviation for ETTSea
af_print(stdev(I)); //calculate and print standard deviation for TPSea
af_print(stdev(J)); //calculate and print standard deviation for VMDSea
af_print(stdev(K)); //calculate and print standard deviation for ETTsw
af_print(stdev(L)); //calculate and print standard deviation for TPSw
af_print(stdev(M)); //calculate and print standard deviation for VMDSw
af_print(stdev(N)); //calculate and print standard deviation for M01
af_print(stdev(O)); //calculate and print standard deviation for M02
af_print(stdev(P)); //calculate and print standard deviation for HS
af_print(stdev(Q)); //calculate and print standard deviation for DMDIR
af_print(stdev(R)); //calculate and print standard deviation for ANGSPR
af_print(stdev(S)); //calculate and print standard deviation for INLINE

std::cout << std::endl;

af_print(var(A)); //calculate and print variance for CCYMMM
af_print(var(B)); //calculate and print variance for DDHHmm
af_print(var(C)); //calculate and print variance for WD
af_print(var(D)); //calculate and print variance for WS
af_print(var(E)); //calculate and print variance for ETOT
af_print(var(F)); //calculate and print variance for TP
af_print(var(G)); //calculate and print variance for VMD
af_print(var(H)); //calculate and print variance for ETTSea
af_print(var(I)); //calculate and print variance for TPSea
af_print(var(J)); //calculate and print variance for VMDSea

```



```

af_print(var(K)); //calculate and print variance for ETTSw
af_print(var(L)); //calculate and print variance for TPSw
af_print(var(M)); //calculate and print variance for VMDSw
af_print(var(N)); //calculate and print variance for M01
af_print(var(O)); //calculate and print variance for M02
af_print(var(P)); //calculate and print variance for HS
af_print(var(Q)); //calculate and print variance for DMDIR
af_print(var(R)); //calculate and print variance for ANGSPR
af_print(var(S)); //calculate and print variance for INLINE

```

```

double seconds_since_start = difftime(time(0), start);
std::cout << seconds_since_start << " seconds" << std::endl; //calculate processing time

```

```

system("pause");
return 0;
}

```

```

void readInputFile(void)
{
    std::ifstream inputFile("SF001120(1).csv"); //open input file
    std::ofstream outputFile;
    outputFile.open("Output.csv");
    std::string line;

    int y = 0;
    while (std::getline(inputFile, line))
    {
        std::stringstream lineStream(line);
        std::string cell;
        int x = 0;
        while (std::getline(lineStream, cell, ','))
        {
            if (x == 0)
            {
                CCYMMM[y] = atof(cell.c_str());
            }
            else if (x == 1)
            {
                DDHHmm[y] = atof(cell.c_str());
            }
            else if (x == 2)
            {
                WD[y] = atof(cell.c_str());
            }
            else if (x == 3)
            {
                WS[y] = atof(cell.c_str());
            }
            else if (x == 4)
            {
                ETOT[y] = atof(cell.c_str());
            }
            else if (x == 5)
            {
                TP[y] = atof(cell.c_str());
            }
            else if (x == 6)
            {
                VMD[y] = atof(cell.c_str());
            }
            else if (x == 7)

```

```

        {
            ETTSea[y] = atof(cell.c_str());
        }
        else if (x == 8)
        {
            TPSea[y] = atof(cell.c_str());
        }
        else if (x == 9)
        {
            VMDSea[y] = atof(cell.c_str());
        }
        else if (x == 10)
        {
            ETTSw[y] = atof(cell.c_str());
        }
        else if (x == 11)
        {
            TPSw[y] = atof(cell.c_str());
        }
        else if (x == 12)
        {
            VMDSw[y] = atof(cell.c_str());
        }
        else if (x == 13)
        {
            MO1[y] = atof(cell.c_str());
        }
        else if (x == 14)
        {
            MO2[y] = atof(cell.c_str());
        }
        else if (x == 15)
        {
            HS[y] = atof(cell.c_str());
        }
        else if (x == 16)
        {
            DMDIR[y] = atof(cell.c_str());
        }
        else if (x == 17)
        {
            ANGSPR[y] = atof(cell.c_str());
        }
        else if (x == 18)
        {
            INLINE[y] = atof(cell.c_str());
        }
        }
        x++;
    }
    std::cout << CCYMM[y] << ", " << DDHHmm[y] << ", " << WD[y] << ", " << WS[y] << ", "
    << ETOT[y] << ", " << TP[y] << ", " << VMD[y] << ", " << ETTSea[y] << ", " << TPSea[y] << ", " << VMDSea[y] << ", " <<
    ETTSw[y] << ", " << TPSw[y] << ", " << VMDSw[y] << ", " << MO1[y] << ", " << MO2[y] << ", " << HS[y] << ", " <<
    DMDIR[y] << ", " << ANGSPR[y] << ", " << INLINE[y] << std::endl;
    outputFile << CCYMM[y] << ", " << DDHHmm[y] << ", " << WD[y] << ", " << WS[y] << ",
    " << ETOT[y] << ", " << TP[y] << ", " << VMD[y] << ", " << ETTSea[y] << ", " << TPSea[y] << ", " << VMDSea[y] << ", " <<
    ETTSw[y] << ", " << TPSw[y] << ", " << VMDSw[y] << ", " << MO1[y] << ", " << MO2[y] << ", " << HS[y] << ", " <<
    DMDIR[y] << ", " << ANGSPR[y] << ", " << INLINE[y] << std::endl;
    y++;
    if (y == numPatterns){ break; }
}
std::cout << "\nNumber of line = " << y << std::endl;

```

```
std::cout << "\nDone Execute" << std::endl;  
outputFile.close();  
}
```

4.6. How System will Help the Operations

This system will help in improving the performance of the current metocean data descriptive statistic system. The GPU-accelerated web application is expected to have higher performance and can process larger data set than existing system.

4.7. Performance Comparison of CPU and GPU in Processing Dataset

Figure below shows the tabulation of the number of dataset versus the processing time:

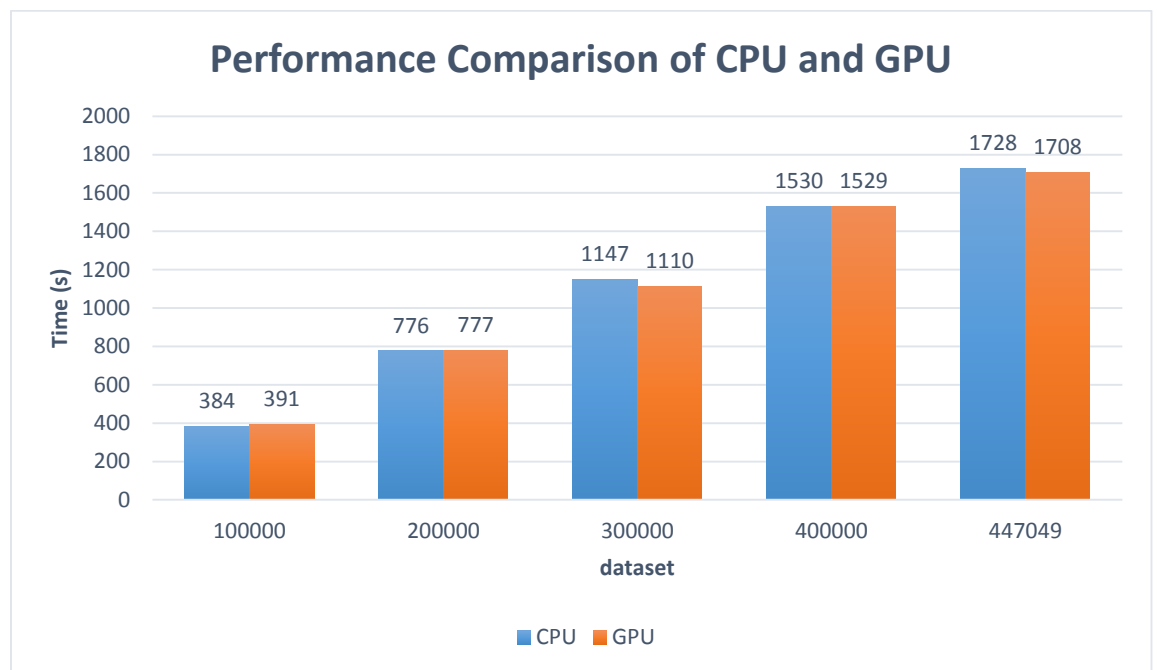


Figure 17: Performance Comparison of CPU and GPU

Based on the result, we can conclude that with the increase amount of dataset the GPU shows a better performance in term of processing time compared to CPU. Increased performance in processing time and less time required to process large number of datasets shows that a GPU-accelerated is more effective in processing a large amount of data.

5.0 Conclusion and recommendation

5.1. Conclusion

This research consist of two phases which are FYP 1 and FYP 2. The first phase was concerning on the setting up of the environment for ArrayFire. This phase focusing on installing ArrayFire libraries on Windows and Linux. The part also include learning about ArrayFire utility which consist of sort, mean, variance, standard deviation, covariance, median and correlation coefficient functions from the documentation in ArrayFire webpage. Researcher also has learn from example that was included in the webpage to understand more about ArrayFire. Other than that, researcher has learn to develop a C++ application which utilizes ArrayFire library.

In the second part, the researcher has develop the application where the knowledge accumulated in FYP 1 is utilized. By using GPU in the implementation, it was expected to improve the performance of the system in comparison to existing CPU based system. GPUs which are massively parallel general-purpose processors are more reliable in terms of power efficiency, compute density and scalability on big data analytics, thus, the application of GPU in descriptive statistic of metocean data is expected to assist companies in oil and gas industries to make a better and real-time business decision.

During this phases, some of the challenges that the researcher has faced is difficulty in installing the ArrayFire libraries. This is because the researcher are not use to installing libraries from source which make it a challenge to the researcher. Besides that, the researcher also facing a problem in understanding the code for GPU computing as this area is a new area of knowledge to the researcher.

5.2. Recommendation

In the future, the researcher would like to recommend to have a basic knowledge in GPU computing specifically OpenCL and CUDA. This is because this research are mainly on GPU computing and by having enough knowledge on the area will help the research and development to go smoothly. It is also recommended to add more parameters in comparing the performance between GPU and CPU besides processing time.

References

1. Van Os, J., Caires, S., Van Gent, M.R.A. (2011) Guidelines for Metocean Data Analysis. Proc. 21th Int. Offshore and Polar Eng. Conference, Maui, Hawaii, USA, June 2011.
2. NVIDIA Corporation (2015). What is GPU Accelerated Computing? [ONLINE] Available at: <http://www.nvidia.com/object/what-is-gpu-computing.html>. [Last Accessed 21 March 2015].
3. Owens, J. D., Houston, M., Luebke, D., Green, S., John E. S., and Phillips J. C., (2008). GPU Computing. Proceedings of the IEEE. 96 (5), pp.879-899
4. Suchard, M. (n.d.). Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures. Journal of Computational and Graphical Statistics, 419-438.
5. R. Wu, B. Zhang, and M. Hsu, "Clustering billions of data points using GPUs," in UCHPC-MAW'09: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop, Ischia, Italy, 2009, pp. 1-6.
6. NVIDIA Corporation (2015). Data Science. [ONLINE] Available at: <http://www.nvidia.com/object/data-science-analytics-database.html>. [Last Accessed 21 March 2015].
7. Vannak, D., Liew, M.S., Yew, G.Z., (2013) Time Domain and Frequency Domain Analyses of Measured Metocean Data for Malaysian Waters Guidelines for Metocean Data Analysis. World Academy of Science, Engineering and Technology International Journal of Environmental, Ecological, Geological and Geophysical Engineering, 7(8), 299-304.
8. Z. Mayeetae, M. S. Liew and M. N. Abdullah, "Validating Hindcast Metocean Parameter with Measured Environmental Loads of Malaysian Water," APSEC-ICCER, Surabaya, 2012.
9. R Mokhtari, M Stumm. IPDPS, (2014) BigKernel—High Performance CPU-GPU Communication Pipelining for Big Data-style Applications.

10. Kevin Parrish (2013). Nvidia GPUs Power World's Largest Virtual Brain. [ONLINE] Available at: http://www.tomsitpro.com/articles/stanford-nvidia-gpu-cuda-machine_learning,1-1104.html. [Last Accessed 21 March 2015].