An Experimental Study on Relationship between Performance and Energy Consumption of Serial and Parallel Text Searching Algorithm.

By

Siti Zulaikha Isahak

A project dissertation submitted to the

Information & Communication Technology Programme

Universiti Teknologi PETRONAS

The requirements for the

Bachelor of Technology (Hons)

(Business Information System)

JAN 2013

UniversitiTeknologi PETRONAS
Bandar Seri Iskandar,
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

An Experimental Study on Relationship between

Performance and Energy Consumption

of Serial and Parallel Text Searching Algorithm.by

By

Siti Zulaikha Isahak

A project dissertation submitted to the

Information & Communication Technology Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION & COMMUNICATION TECHNOLOGY)

Approved by,

____

(Ms Nazleeni Haron)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

APRIL 2013

ABSTRACT

The world data is growing vigorously intersecting of large ordered sets and it is a common problem in the evaluation of data queries to a search engine. Thus, text retrieval systems have become a popular way in providing support for text databases. However this becomes a major question among us like how much energy is consumed? How to reduce execution time in searching large amount of data?

In this paper, text searching algorithm is using to study the relationship between performance of computer and amount of energy produced in serial and parallel text searching algorithm. The amount of energy produced should be reduced along with the execution time to increase performance in data searching. Based on data recorded from the series of experiments, Serial Text Searching Algorithm is saving energy and reducing power usage. However, their performance is reducing as a smaller processor speed is using. In contrast to Parallel Text Searching Algorithm, there are larger amount of energy consumed from this experiment. However, it is approved that the performance of parallel experiment is far better than a single node performance.

LIST OF FIGURES

ABBREVIATIONS AND NOMENCLATURES

HPC	High Performance Computing
CPU	Central Processing Unit

TABLE OF CONTENT

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Energy consumption in computing activity

The world is facing an environmental crisis. Global warming threatens to alter weather patterns, with enormous human and economic consequences. Most government, independent society including organization is working towards reducing the carbon emission. There are many initiatives provide by them to reduce global warming effect. One of very important ways to protect the environment is by reducing electricity usage.

According to World Factbook of the CIA, it is stated that China is the largest nation that consumed power electricity, 4,940,000,000 Watt, followed by United States, 3,886,400,000 Watt in between 2011 and 2012 years. Malaysia consumption is more than 120,000,000 Watt as recorded in 2011. There are many factors contributing to these amounts of electricity consumption like cooling and heating appliance, television, refrigerator, washing machine, lighting and computers for household category. For big organization, they have to pay higher electricity cost for every month.

With the world moving very fast, most organization is fully depended on internet to transmit, receive and search data. The energy consumption today is expected to double by 2020 due to worldwide usage of computers, data storage and communications network. Carbonfund.org claimed that deliver and dispose of junk mail produces more greenhouse gas emissions than 2.8 million cars on the road.

Below is a normal usage of computers:

Device	**Power (W)**
Desktop Computer & 17” CRT monitor	150 - 340
Desktop Computer & Monitor (in sleep mode)	1 - 20
17” CRT monitor	90
17” LCD monitor	40
Laptop computer	45

So what is the relationship between this algorithm and global warming? Main concept of this algorithm is acting as a filter that read input and produces the output accordingly. According to Külekci M. O, data is available electronically and requires efficient searching techniques to extract the desired data.

There are several Search Engines that commonly been used by users like Google, Yahoo and Bing. As recorded by Google, their average searches per day are reaching 4, 717, 000, 000. For every single search button you hit on Google, about 0.3 watts produced to our surrounding. So, the total energy produced by Google search per day is 1, 415, 100, 000 watts. This huge amount of energy can use to light up 100 watt bulb for almost 1801 days continuously.

Above data show that, there are billions of energy produced per day just from our simple search activity in search engine. Therefore, this project is purposely to conduct several experiments to study the trade-off between performance of text searching algorithm in parallel and serial with the amount of energy consumed.

Parallel Computing

During the earlier generation, computers were built with a single central processing unit (CPU) and performed serial fixed-point arithmetic using a program counter, branch instructions, and a processor registers which used to store intermediate results. The CPU is consists of memory access and input/ output mechanisms act as the interface to the human operator. This design is based on Hungarian mathematician John von Neumann architecture. The design of Von Neumann architecture is simpler which an instruction fetches and a data operation cannot occur at the same time because they share a similar bus.

Later the evolution of computers continue by implementing High level languages (HLLs) such as Fortran, Algol and Cobol and inventing of pipelining and cache memory to increase speed between CPU and main memory. Currently, parallel computing is on rising by implementing various architectures, using shared or distributed memory or optional vector hardware. According to Hwang.K, starting from 1975 until 1990, the technology of parallel processing gradually become mature and entered the production mainstream. The example of parallel computing is Illiac IV, MasPar and Cray Y-MP. (Hwang, K., 2005)

According to Quentin F. Stout, the use of two or more processors (cores, computers) in combination to solve a single problem is called parallel computing. In parallel computing, a problem is broken into discrete parts where each part is further broken down to series of instructions. The instructions from each part will be executed simultaneously on different CPUs.

In most condition, software codes have been written to suit the environment of serial computing. The codes aimed to be run on single CPU. Then the code will be broken into several parts and executes one by one. Only one instruction will be processed at one time. (Barney B. 2012). In order to run the text processing systems in parallel computing method, the written codes must be changed a little bit. The written codes should be work by breaking apart into discrete parts where they can be processed simultaneously by several CPUs. This method is to allow several instructions been executed simultaneously on different CPUs.

UTP has abundance of available computer in the lab and has utilized the usage of available computers resource in lab to form parallel grid. The idle time on computers in lab will be used and form a distributed high performance computer. The main reasons UTP set their own campus grid are to save time and money. Those computers can be used to compute and solve larger problems which are impossible or taking longer time to be solved if using a single computer. For example in this project is to search a data by implementing text searching algorithm. One processor requires longer time to compute huge data in contrast to several processors.

This is good enough for an organization like UTP to have parallel computing source. However, it is a question to wonder on how much the energy consumption by campus grid in contrast with serial computing. The energy consumption is becoming important limiting factor while there are many people out there working out to reduce energy to save money and in the same time increases the performance of computing process. (Freeh V. W., Lowenthal D. K. and et. al., 2007).

Text Searching Algorithms

To conduct the experiment, some large data is required to be processed on both serial and parallel computing. Thus, Text Searching Algorithm is implemented. The texts are main thing in "word processing" systems, which provide facilities for the manipulation of texts. Several existing studies have applied text classification techniques in handling the problem with ambiguous requirements. (Polpinij, J., 2009). The systems require advance processor to filter objects which are quite large. There are several examples of these applications are text filtering, text searching, information extraction, information statistic, content rewriting, automatic generation of text and others.

Text Searching Algorithm is getting famous nowadays since the hostilely rising of textual information electronically. It is reported by WorldWideWebSize.com that until 29 January, 2013 there are at least 12.86 billion pages in Indexed Web. This number is believed to grow year by year. Basically Text Searching Algorithm is to find within a text t a match for a pattern p, where text, pattern, and match can be interpreted in different ways. (Obaja A., 2012). This technique is usually involved text document that contains very large data. This concept acts as filters that read input and produce the output simultaneously. There are several ways can be implemented in Text Searching:

- Simple text searching
- Rabin-Karp algorithm
- Knuth-Morris-Pratt algorithm
- Boyer-Moore(-Horspool) algorithm
- Approximate matching
- Regular expression

Throughout this project, a simple text searching is implemented to search for a students' ID in a text file.

1.2 Problem Statement

Power consumption is becoming a critical issue for high performance computing because of some limitation factors including cost, space, reliability, and maintenance. (Son S. W., Malkowski K. and et. al., 2006). For larger dataset they may take longer time to run on single machine. For example, the longer time taken to process data on serial searching algorithm, the more energy will be consumed by the machine. In contrast, it is known that parallel text searching algorithm can reduce response time in processing data. However, the total amount of energy consumed by numbers of nodes in parallel machine is yet to be measured.

This experiment is conducted on both machines, serial and parallel to study the relationship between performance and energy consumption by Text Searching Algorithm. It is imperative to know the optimal node size for trade-off between performance and energy consumption. By knowing the optimal node size, it can help in maximizing energy saved without increasing execution time. This can increase performance on high performance computer.

1.3 Objectives of study

The purpose of this project is to investigate on the performance of the campus grid set up:

- To evaluate the trade-off between energy and performance of Text Searching Algorithm.

Scope of Study

To fulfill the research project requirement, there are three main areas that need to be deliberated:

- Text Searching Algorithms (TSA).
- Parallel computing architecture.
- Distributed memory (MPI Library)
- Electrical power consumption.

CHAPTER 2
LITERATURE REVIEW and/ or THEORY

The World Wide Web is growing rapidly and forming into a massive collection of information. This data is available electronically and requires efficient searching techniques to extract the desired data. Because of this, searching for keyword on text files becomes a very common task on everyday activity. (Külekci M. O., 2007). According to Google, there are more than one billion questions asked by prople around the globe. 15% of the searches in everyday are new to them. (Matt Cutts, 2012).

Parallel computing is undergoing evolution and attempts to solve many complex and interrelated natural events or industrial process happening at the same time. For example are galaxy formation, research in genetics, rush hour traffic, auto assembly, data mining and medical imaging and diagnosis. Currently, fast and inexpensive computers are now essential for almost all human activity and have been critical factor in increasing economic productivity, enabling new defense systems, and advancing the frontiers of science. As demand for increased technology performance shows no signs of slowing, thus researchers need to find ways to sustain increasing performance. (Murphy B. 2011).

Since the capabilities of single-core processors cannot satisfy the requirements of the computer applications. Multicore processors are more and more popular to be used to provide computational power. (Liu Y. and Gao F., 2010). Individual computer processors have limits to their performance. Only through parallel programming developers can scientific and practice high performance scale their workloads. (Demko, A. B., 2011). However, a programmer must be skilled in many areas especially in low level language like C or getting familiarize with Linux as the operation system, for example.

2.1 Parallelizing code

Through scalability, parallel program can be executed faster than those constrained to single processors. (Demko, A. B., 2011). As a result user can process more data and generate more accurate results. This process made impossible problem become possible to be solved.

In parallel computing, there are two major camps, homogenous and heterogeneous. Homogenous systems are widely used to this day including hardware like the traditional multicore processor which contains a number of similar or symmetrical cores. Conclude that homogenous does not contain any processing elements that differ from others within the same system. In contrast, heterogeneous systems contain processing elements that differ from others within the same system. They are not composed in the same hardware instead of contain elements that may be more suited to specialized tasks. (Solomon, S. 2012).

2.2 Flynn's Classical Taxonomy

Flynn's classification is based on multiplicity of instruction streams and data streams observed by the CPU during program execution. All the computers classified by Flynn are not parallel computers, but to grasp the concept of parallel computers, it is necessary to understand all types of Flynn's classification.(Vedyadhara, 2010).

Outside of definition based on processing elements composition, Flynn with his taxonomy, splits up parallel computing systems into three categories Single Instruction, Multiple Data (SIMD), Multiple Instruction, Single Data (MISD) and Multiple Instruction, Multiple Data (MIMD). (Barney, B., 2012). Single Instruction, Single Data (SISD) is not considered as parallel computing. It is a serial computer processor because only one instruction stream is being processed by the CPU during any one clock cycle.

In MIMD each processing element executes independently of one another. It allows for each processing element to execute different instructions on different data from one another. SIMD, on the other hand, involves each processing element executing in lock-step with one another. In the other meaning, every pro processing element executes the same instruction at the same time, but executes this instruction on different data. MIMD exploits task-level parallelism (achieving parallelism by executing multiple tasks at one time), while SIMD exploits data-level parallelism (achieving parallelism by taking advantage of repetitive tasks applied to different pieces of data). (Solomon, S., 2012).

2.3 Parallel Computer Memory Architectures

There are three common type of memory architecture implement in parallel computing. The first one is Shared memory. Generally, shared memory has ability for all processors to access all memory as global address space. There will be multiple processors operate independently but share the similar memory resources. (Barney, B., 1995). The major advantage for this memory is easy for user to use memory efficiently and data sharing will be faster. The limitation of this memory is can cause severe bottlenecks due to limited bandwidth.

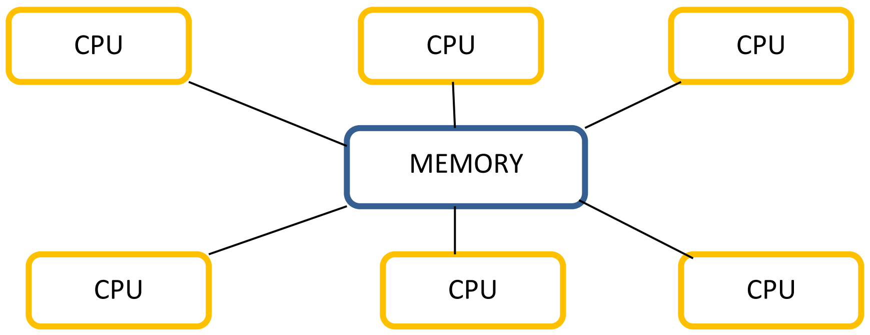


Figure 1. Shared memory

The second type of memory is Distributed memory. In distributed memory, each processor has its own local memory. Memories between processors do not rely on each other. Data is shared across a communication network using message passing. The advantage of this type of memory is memory size depends on number of processors, size of memory and volume of bandwidth. Besides, processors can access their own data without any interference.

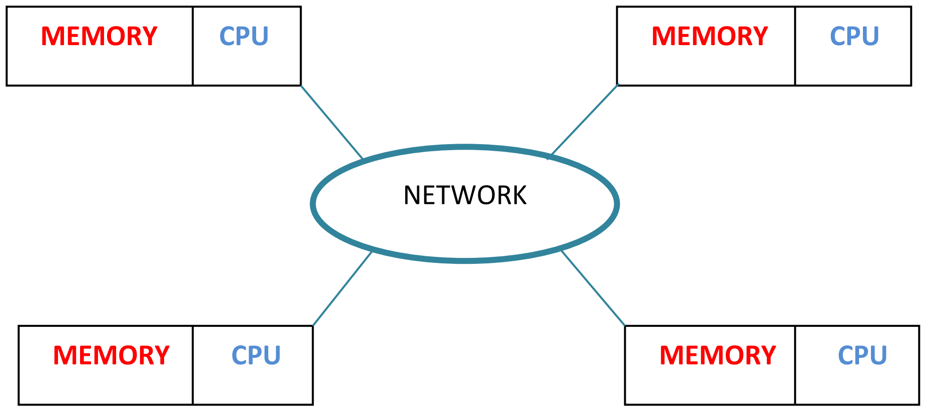


Figure 2. Distributed memory

The third type of memory is Hybrid Distributed-Shared memory. This type of memory is combination of Shared and Distributed memory.

<u>2.4 Processor Communication</u>

On the other hand, programmer must decide the best way to interconnect and organize the various task processors in the system. In the delegation (boss-worker, master-slave) model Figure 1 (a), one process is the boss, while one or more processes are the workers. The workers do nothing but pull work from the boss and return results. The boss is responsible for distributing work, controlling overall application flow and terminating the workers. The boss does not do any computational work itself and simply sits and waiting for requests and results from the workers. (Solomon, S. 2012).

In the peer-to-peer model Figure 1 (b), all the processes are more or less equal. All of them are doing the same work. This implementation is the closest to a pure SIMD or MIMD implementation. In the pipeline model Figure 1 (c), each process or thread handles one step in a multi-step process. This process is dividing by task and an example of MIMD implementation. Pipelining computations can provide an efficient solution to cross-node dependencies in loops. As a result, a number of parallelizing compilers and systems use pipelining to exploit parallelism. (Balasubramanian, K. and Lowenthal, D. K., 2002). Finally, the models can be combined into a composition as in Figure 1 (d). This is often the result of combining algorithms, either sequentially or in a nested way. (Solomon, S. 2012).

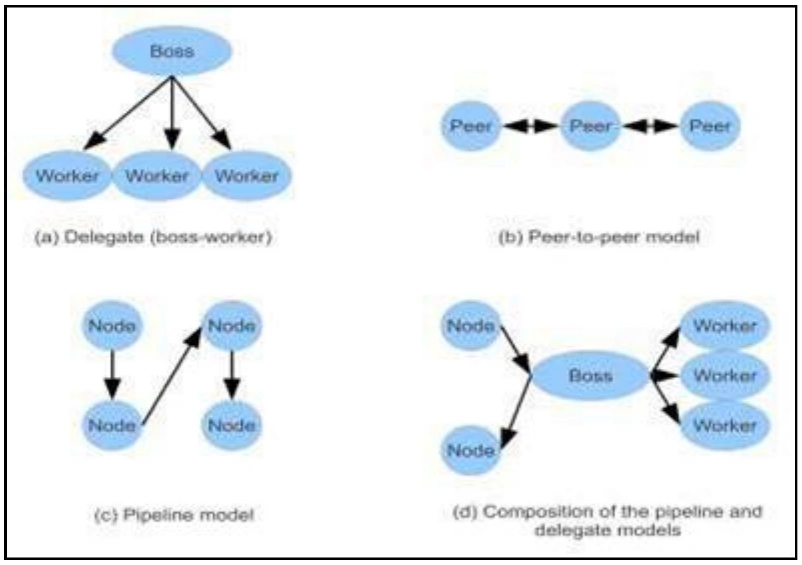


Figure 3.A sample of organizational models. Arrows indicate data flow.

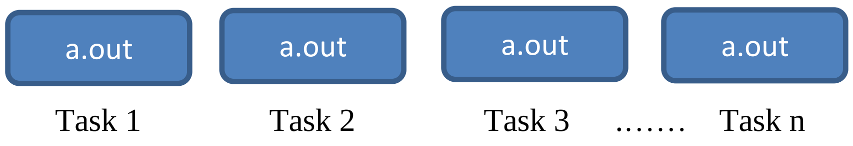
By far the most important reason for developing data parallel applications is the potential for scalable performance. Even in the face of the longer development times, a performance improvement of one or two orders of magnitude on current parallel machines may well be worth the effort. (Nyland L. S., Prins J. F. et al., 2000).

2.5 Parallel Programming Models

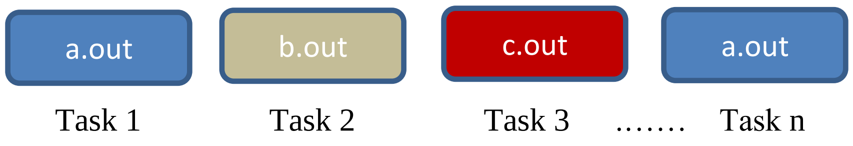
According to Barney, B., there are several parallel programming models that are commonly used.

- Shared Memory Model (without threads)
  - Tasks are sharing common address space. Locks or semaphores can be used to control access to the shared memory.
- Threads Model.
  - Type of shred memory programming.
  - Each thread local data but also shares the entire resource of program in the main program, a.out. The a.out performs some serial works and generates threads to be scheduled and run by operating system concurrently.
  - It is best described as a subroutine within the main program.
  - Thread can be implemented in POSIX Threads or OpenMP library.
- Distributed Memory/ Message Passing Model
  - Tasks in this model exchange through communications by sending and receiving messages.
  - This model is using MPI library.
- Data Parallel Model
  - Tasks will perform depending on data set. The data set usually in a form of structure like array or cube.
  - This model can implement shared or distributed memory. For shared memory, all tasks can get access through global memory. While in distributed memory, data is split up and resides as “chunks” in respective local memory.

- SPMD and MPMD
  - **SPMD**: Single Program Multiple Data.
  - All tasks execute the same instruction (a.out) simultaneously on different data.
  - This program can be threads, message passing data parallel or hybrid depending on nature of problem that need to be solved.



  - **MPMD**: Multiple Program Multiple Data.
  - Tasks may execute different programs (a.out, b.out, and c.out) simultaneously on different data.
  - MPMD can extend problem into functional decomposition better than into domain decomposition.



2.6 Parallel Programming Tools

There are several programming languages available to compose parallel programming application. Some programmers are prefer to use C++ and compiled with original SGI compiler. (Amato, N. M. and Dale, L.K., 1999). The other languages are Fortran, and C. Parallel programming tools that facilitate communication and coordination among master and other nodes. For distributed memory application we are focusing on MPI.

MPI is stand for Message Passing Interface (MPI). It is a library has an extensive collection of operations for exchanging messages and collecting information. MPI is compatible with all major operating systems and can be used with C, C++ and FORTRAN. MPI is commonly used for distribution memory system because they do not share the memory. (De, M., De, S. and et. al, 2008). Unlike MPI, Open Multi-Processing (OpenMP) which is a combination with MPI can be applied in shared memory machine. (Kasabov, A. and Kerkwijk, J. V., 2011). The Open MP has provides a compiler which can automatically parallelizing numbers of loops.

According to Barney, B., OpenMP is a compiler directive based. The OpenMP for Fortran API was released earlier in 1997 compare to OpenMP C/ C++ was released in late 1998. (Barney, B., 2012). OpenMP under C/C++ permits code to be augmented with OpenMP directives. Plus, OpenMP is compatible with other platform including Unix and Windows NT.

CHAPTER 3
RESEARCH METHODOLOGY
and PROJECT WORK

Research Methodology

Throughout this project, my research methods have been focus on studying several techniques in parallel computing and identify the relation of performance and energy consumption:

1. Develop a Serial Text Searching Algorithm
2. Conduct experiment and repeat by using different number of processors.
3. Explore on architecture and task distribution in distributed memory of parallel computing.
4. Design a programming architecture of Text Searching Algorithm.
5. Conduct Parallel Text Searching Algorithm on different number of nodes to measure the level of energy consumed against the performance.
6. Analyzed data.

Project Work

Implementing agile method suits this project implementation. This is due to the basic concepts of agile, adapt to change. Choosing agile method allows the logical program to be amended easily. The code is designed to cope and adapt to new ideas from the beginning to allow changes to be made easily. Implementing agile, changes can be made if needed without getting the entire program rewritten.

Simply, agile development offers a lightweight framework for helping the project to focus in achieving the main purpose of this project, the efficiency of parallelizing text processing. These can reduce the potential of overall risk associated during software development. Agile development promotes continuous planning and feedback to ensure the project meets the ultimate goal. In particular, through agile development, it helps to accelerate the delivery of systems value.

There are various methods present in agile testing like Scrum, Extreme
Programming and Adaptive Software Development. Extreme Programming (XP) allows the development team to build software quickly and build software properly. The development and testing happen in the same time. In XP the project requirements are describe in one story.

In XP, focus is more on customer/client stories which represent the project requirements for each system release. The XP approach involved many small releases of system functionality and features. At first, the code is written to test the system functionality and features before actually writing the specific code. The involvement of the customer from the inception of the project through the Customer/Client acceptance testing before production release of code ensures strong buy in the Customer/client. (Sauter, V. L., 2012). In this project, the role of client can be UTP itself.

There are many ways to describe Agile XP method. For this project, the method is summarized in 6 phases:

<u>Planning</u>
- Define problem
- Listing requirements
- Gather information on security related

<u>Analysis</u>
- Prioritize the main story or main function
- Define iteration span (time)
- Resource planning for development

Design

- Divide the tasks
- Test scenario preparation for each task

Execution

- Coding
- Unit testing
- Execution of serial Text Searching Algorithm program
- Iterate Text Searching Algorithm program for serial computing
- Iterate Text Searching Algorithm program for parallel computing

Wrapping

- Regression testing
- Demos and reviews
- Develop new requirements based on the need
- Process improvements based on end of iteration review comments from lecturers

Closure

- Upgrading software
- Status report

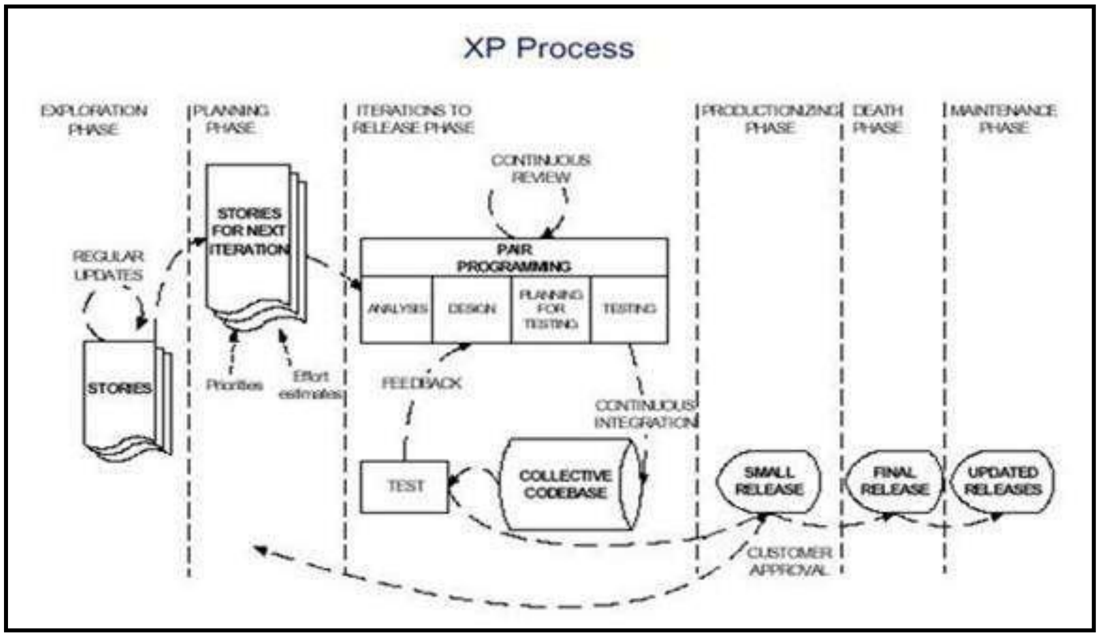
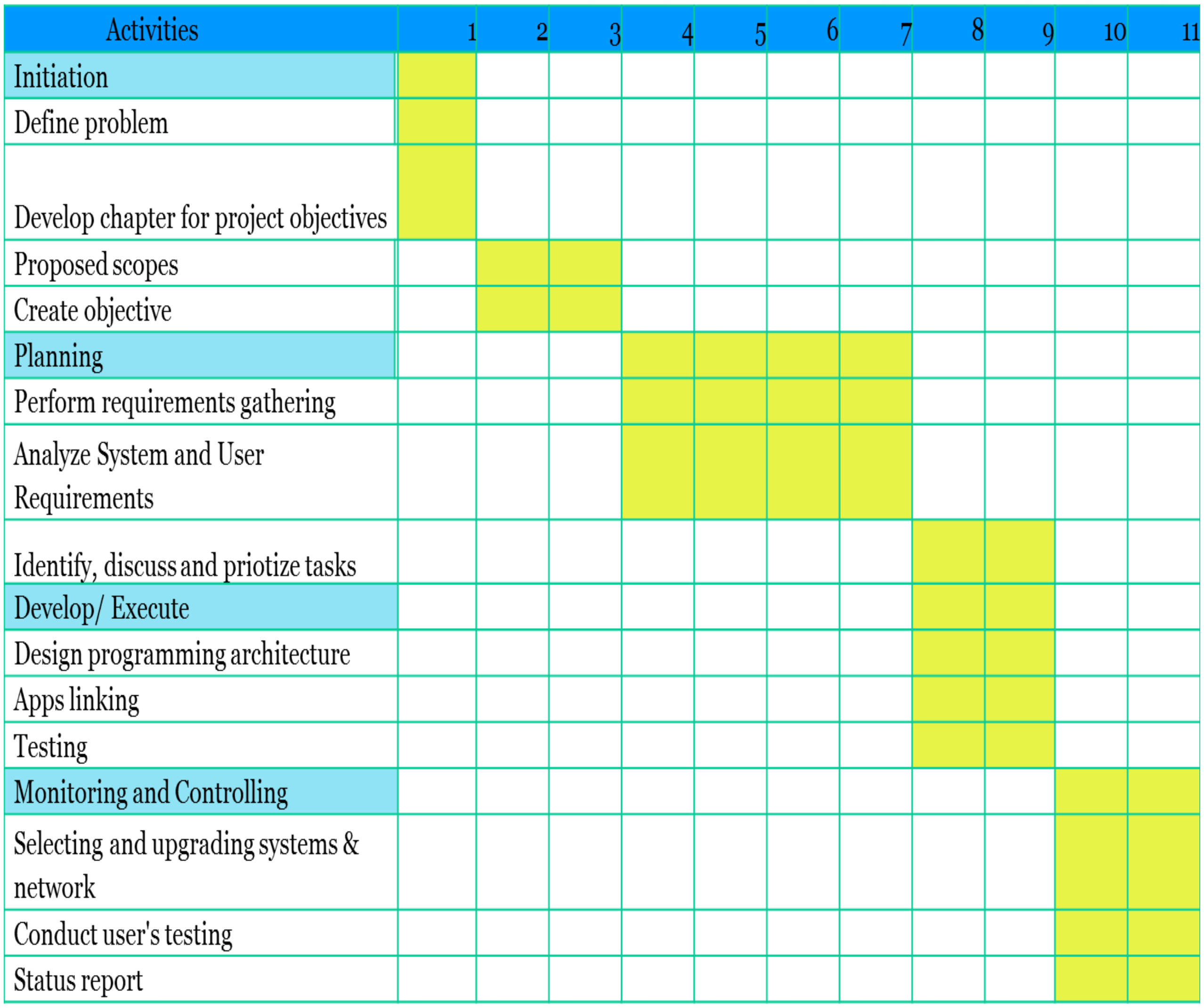


Figure 4. Practices of XP in the system development life cycle

GANTT CHART

Our defined Gantt chart is as below:



Activities	1	2	3	4	5	6	7	8	9	10	11
Initiation	🟨										
Define problem	🟨										
Develop chapter for project objectives	🟨										
Proposed scopes		🟨	🟨								
Create objective		🟨	🟨								
Planning				🟨	🟨	🟨	🟨				
Perform requirements gathering				🟨	🟨	🟨	🟨				
Analyze System and User Requirements				🟨	🟨	🟨	🟨				
Identify, discuss and priotize tasks								🟨	🟨		
Develop/ Execute								🟨	🟨		
Design programming architecture								🟨	🟨		
Apps linking								🟨	🟨		
Testing								🟨	🟨		
Monitoring and Controlling										🟨	🟨
Selecting and upgrading systems & network										🟨	🟨
Conduct user's testing										🟨	🟨
Status report										🟨	🟨

2.7 Performance Measures

Below are some measurements to indicate the performance of Serial and Parallel text Searching Algorithm and power consumed.

- Number of processor / Frequency (*GHz*)
- Number of nodes
- Electric Power (*Watt*)
- Execution time (*s*)
- Voltage (*V*)

Tools

- C Programming Language
- Visual Studio 2008
- MPI library
- Personal computer (with processor of 1.6 GHz, 2.0 GHz, 2.7 GHz)
- Computers in lab
- Power Meter

CHAPTER 4
RESULT AND DISCUSSION

<u>Sequence Algorithms</u>

For this project, Sequence search algorithm is implemented to find an item in an unsorted sequence. Under Sequence search topic, there are numbers of algorithm to perform this task. For example, linear search, selection algorithm, binary search algorithm, jump search, predictive search and Fibonacci search techniques. Different algorithms have different level of efficiency in processing text.

For now, I have implemented a linear search technique which is used to find whether a given Student name or Id number is presented in an array. If the data is found in text file, it will display the corresponding number of line and Id number. The function will do testing by looping on each element in array by comparing with the desired data.

For the parallel programming objective, this code will be rewrite again to run in parallel machine. The comparison of the processing time will be recorded to find which has greater efficiency performance.

Flowchart for Serial Text Searching Algorithm

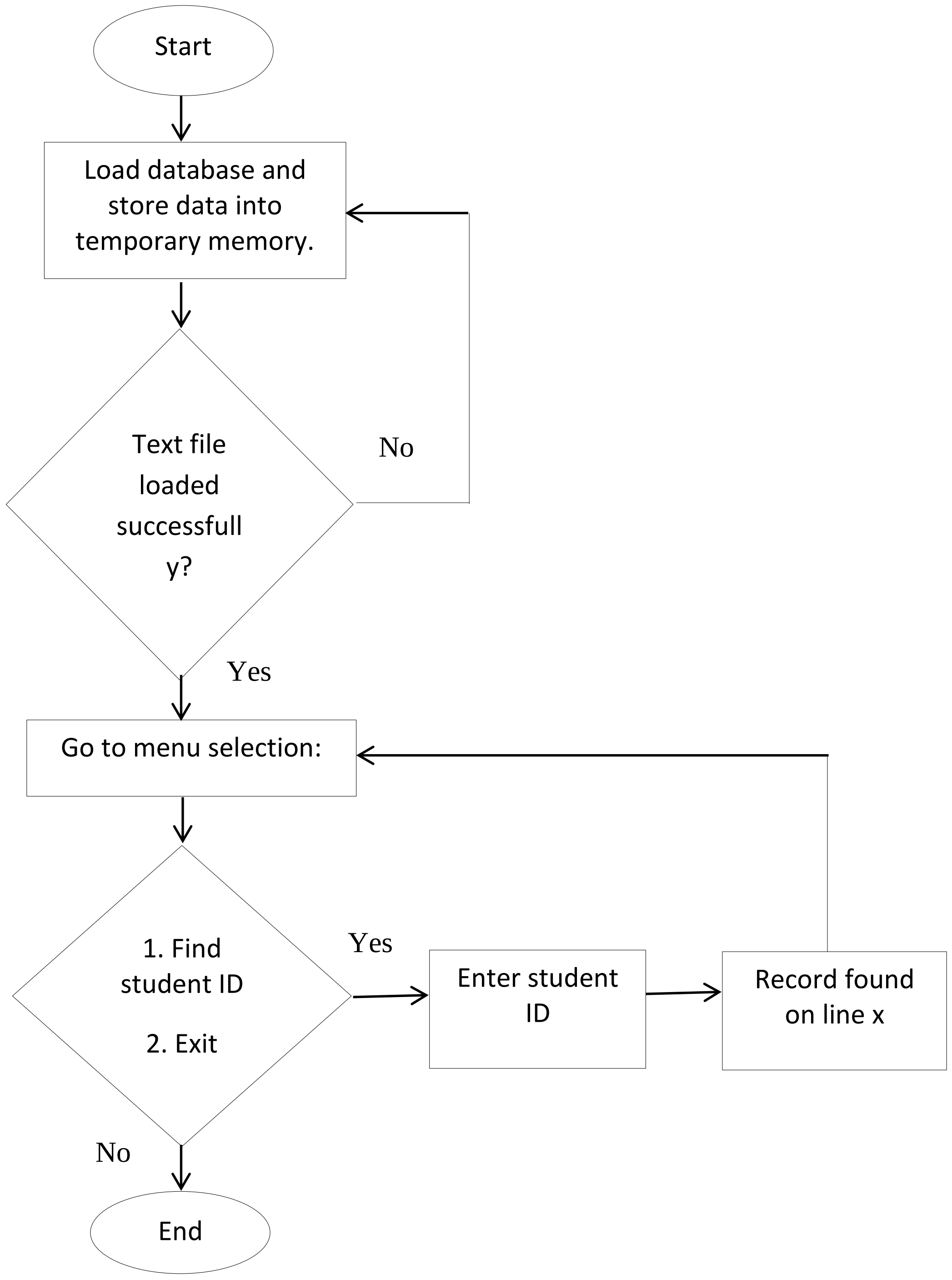


Figure 5: Flowchart for Serial Text Searching Algorithm

Serial code to retrieve a string in text file.

First of all, I have to construct serial code to retrieve a string from text file. As mention in Tools part, I choose to develop in C programming language. This code basically used to retrieve a specific student ID from a large data stored in a text file.

Basically, there are there important functions in this code. The first one is to find student Id and declared as **int FindStudId(long int p)**. Then a function to check on lines of input strings and been declared as **int chkstrdig (char str[], int range)**. Another important function is to load text file and compute the data to temporarily store in RAM. This function is not returning any value and will only prompt user whether the text is successful loaded or otherwise. The function is declared as **void LoadDB(void).** Below are more details regarding functions involved in Serial Text Searching Algorithm;

- int FindStudId(long int p)

```c
int FindStudId(long int p)
{
int k, studid_found_flag= -1;
    for(k=1; k < add_count + 1; k++)
    {   if (add_count != 0)
      {   //if (k != 0 && (k%15) == 0)
              if (k == 0)
            {    getch();}
        if (p == studid[k])
        {  printf("Student ID [%-4d] was found in
record No. [%3d  ]\n",studid[k],k+1);
           studid_found++;
           studid_found_flag = 0;
        }
      }
    }
    if (studid_found_flag == 0)
    { return(0);  }
    else
    { return(-1); }
}
```

- int chkstrdig (char str[], int range);

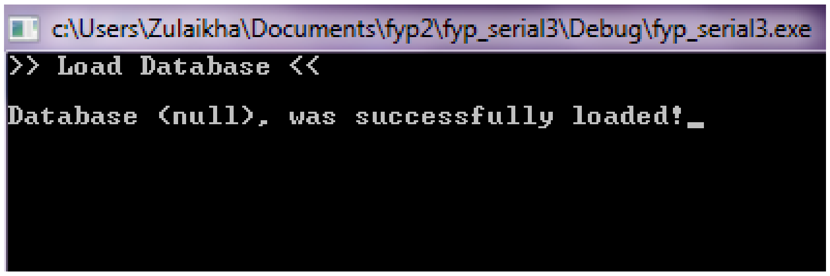
```
int chkstrdig (char str[], int range)
{
int lenght=0,k;
    lenght = strlen(str);
    if (lenght > range)
    {       return(-1);}
    if (lenght <= range)
    {  for (k=0; k < lenght; k++)
       {   if (isdigit(str[k]) == 0)
         { return(-2); }
       }
    }
    return(0);
    }
}
```

- void LoadDB(void)

```
void LoadDB(void)
{
int count,dbfilecount=0;
char finstudid[280];
int error_junk;
long int l_finstudid;

FILE *f1;
     f1 = fopen ("saya.txt","r");
     if (f1==NULL)
     {
            fprintf(stderr,"There was an error reading your
database file!\n");
            getch();
            exit;
     }
     else
     {      for (count=0; count < MAXDB; count++)
            {
             if (!feof(f1)) /* if not end of file continue to
input data */
                  {     fscanf(f1,"%s\n",&finstudid);
                        error_junk = chkstrdig(finstudid,4);

             if (error_junk == -1 || error_junk == -2)
              {  printf("Sorry that was an invalid
database\n");
                    printf("\nNow working in RAM MODE!");
                      getch();
                        break;
                  }

          l_finstudid = atol(finstudid);
            studid[count] = l_finstudid;
            dbfilecount++;
                  }
            }
        if (error_junk ==0)
       {
            printf("\nDatabase %s, was successfully
loaded!",dbload);
            getch();
            add_count = dbfilecount;
       }
     }
     fclose(f1);
}
```

Below are some print screens for Serial Text Searching Algorithm

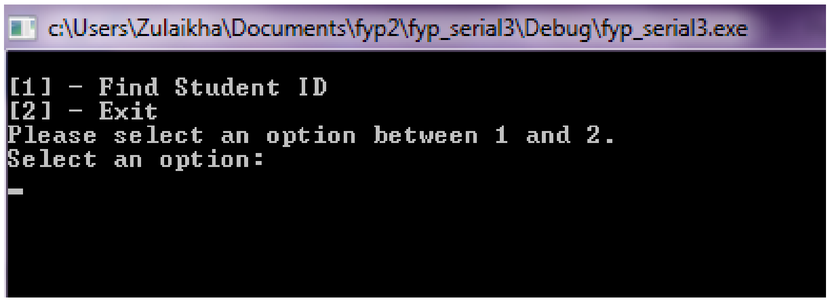
- To indicate whether data is successfully loaded or not



```
c:\Users\Zulaikha\Documents\fyp2\fyp_serial3\Debug\fyp_serial3.exe
>> Load Database <<

Database (null), was successfully loaded!_
```

Print Screen 1

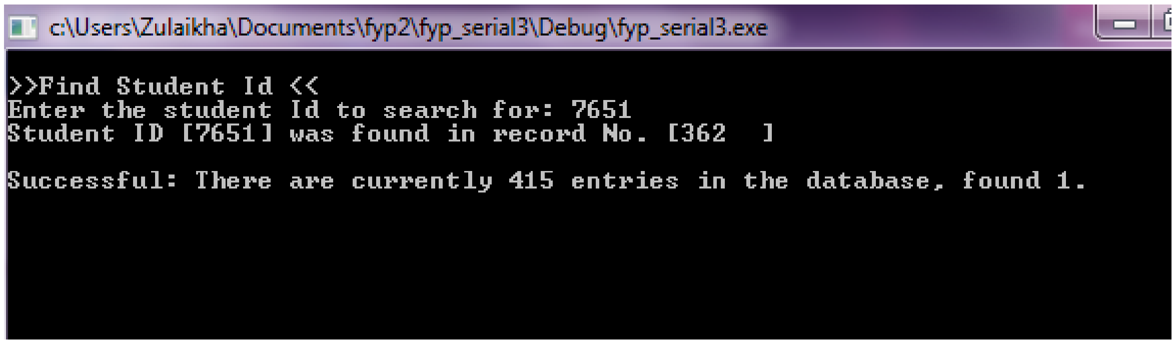
- Display menu option



```
c:\Users\Zulaikha\Documents\fyp2\fyp_serial3\Debug\fyp_serial3.exe

[1] - Find Student ID
[2] - Exit
Please select an option between 1 and 2.
Select an option:
_
```

Print Screen 2

- Instruction to get string input and match string input with data in temporary storage



```
c:\Users\Zulaikha\Documents\fyp2\fyp_serial3\Debug\fyp_serial3.exe

>>Find Student Id <<
Enter the student Id to search for: 7651
Student ID [7651] was found in record No. [362   ]

Successful: There are currently 415 entries in the database, found 1.
```

Print Screen 3

Result for Serial Text Searching Algorithm

Processor (GHz)	Power (W)	Voltage (V)	Time (s)
1.6	2	0.8	0.72
2.0	4	1.0	0.60
2.7	6	1.2	0.55

Figure 6: Result for Serial Text Searching Algorithm

Energy Consumption Graph on Serial Text Searching Algorithm

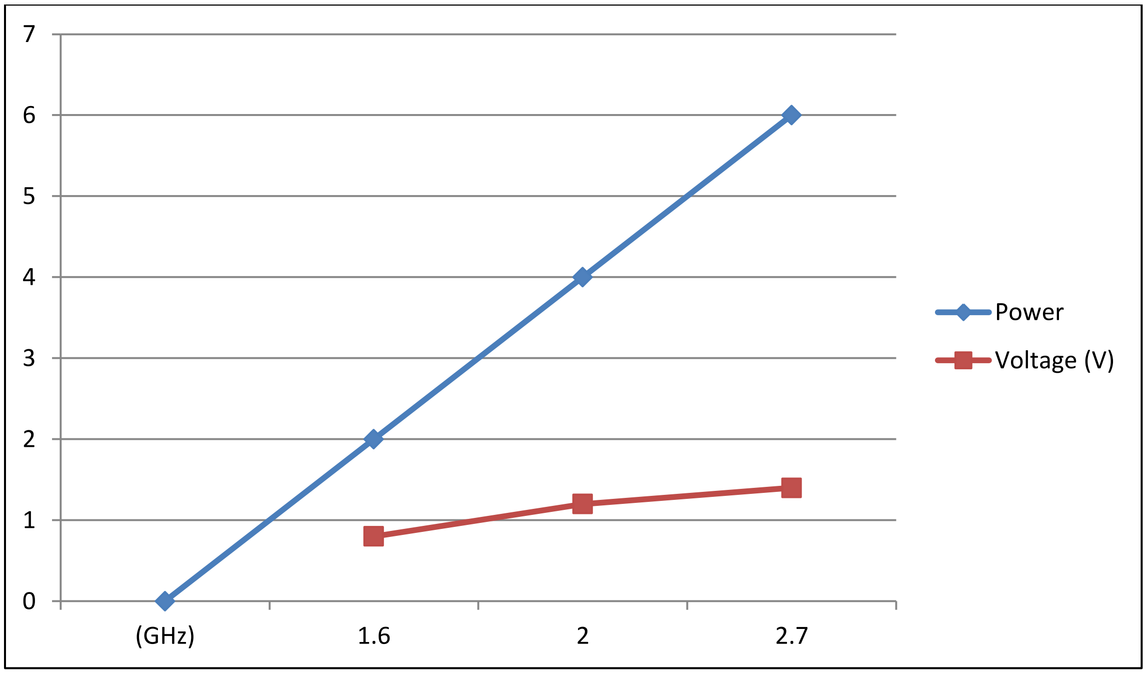


Figure 7: Number of processor (GHz) against Voltage (V) and Power Consumed (W)

<u>Performance Graph on Serial Text Searching Algorithm</u>

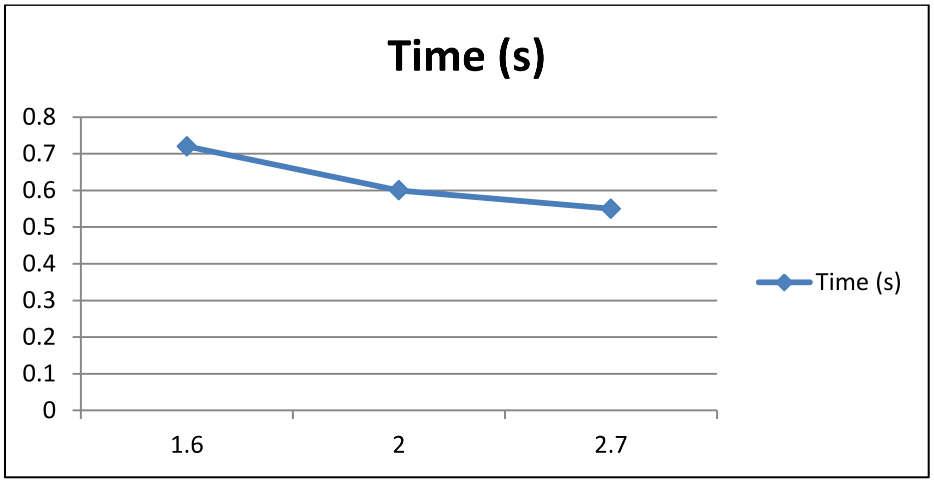


Figure 8: Number of processor (GHz) against Time taken to process data (s)

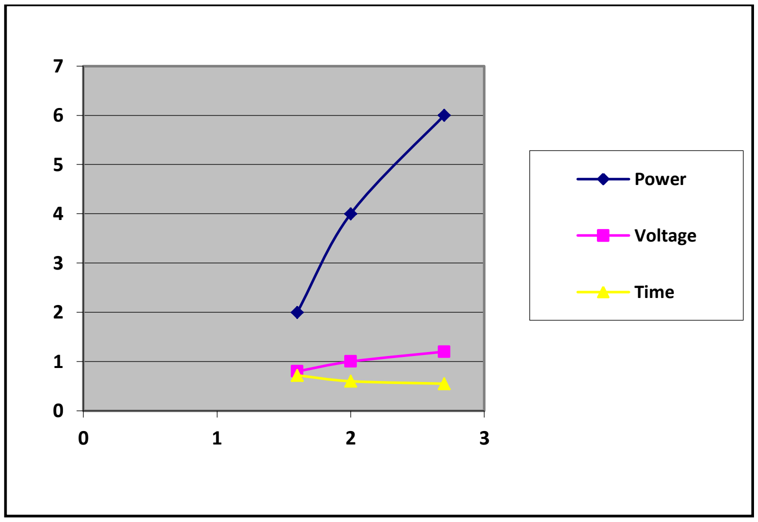


Figure 9: Performance graph for Serial Text Searching Algorithm

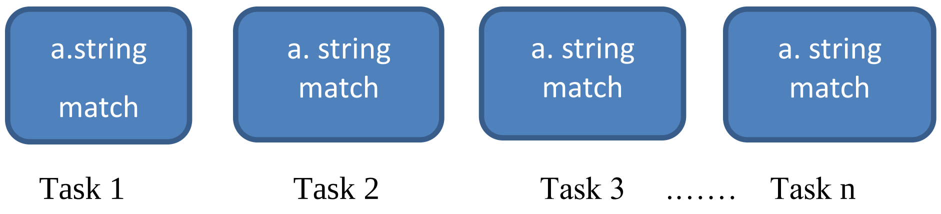
Based from the result recorded, I can conclude that using a single node to compute Serial Text Searching Algorithm is saving energy and reducing power usage. The smaller processer speed used, the less energy consumed. In contrast, their performance is reducing by using smaller processor speed.

Parallel Text Searching Algorithm

Task parallelism on distributed memory is allowed by using MPI library. There will be no communication between the running nodes during a computation (Constantinescu Z. and Vladoiu M., 2011). Thus, I have to choose Single Processing Multiple Data, SPMD to divide the task.

Single Program Multiple Data

- This will process similar task upon different type of data.



For parallel series of experiments, we implement distributed memory architecture where each processor has its own local memory.

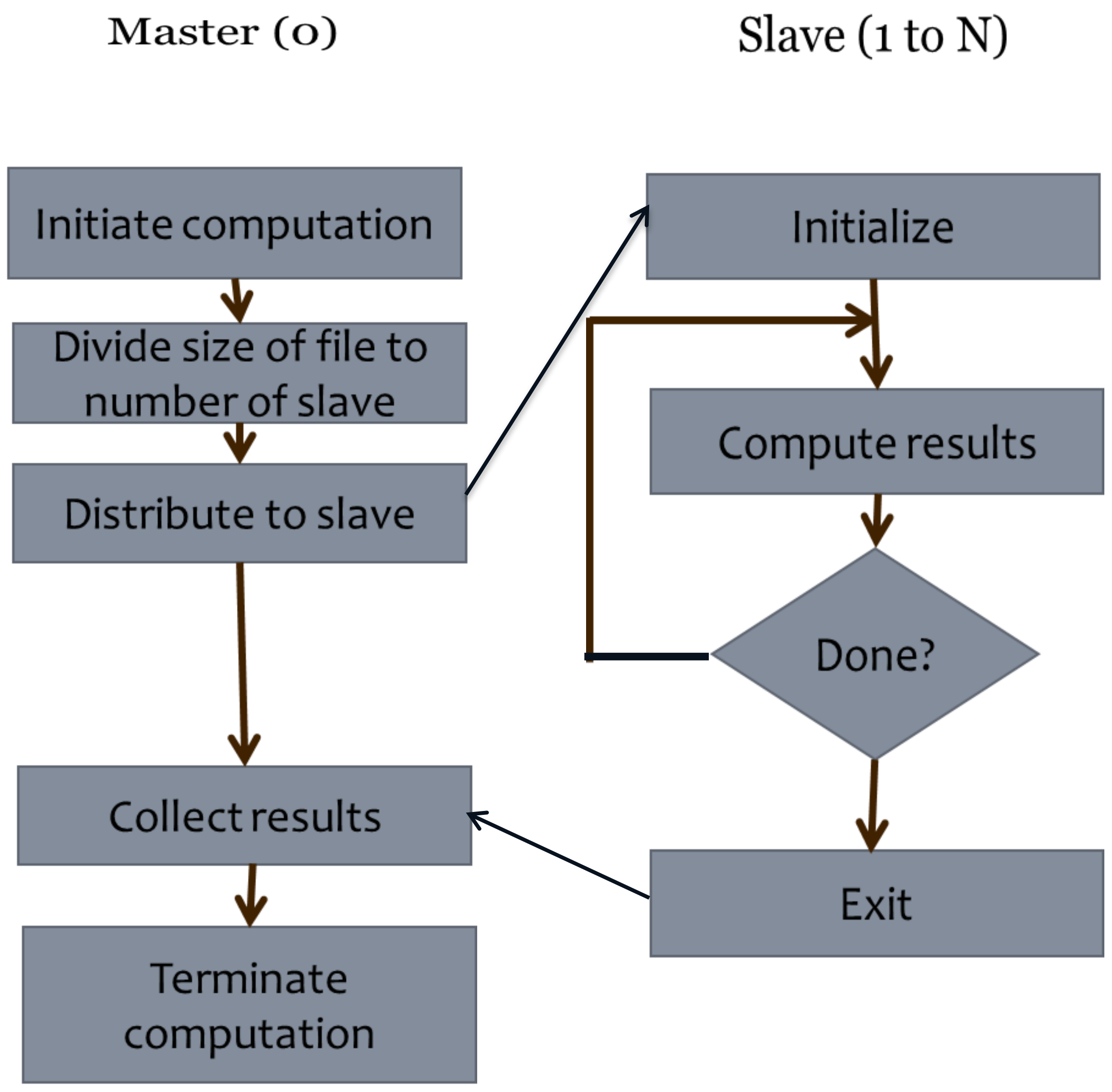


Figure 10: Master-Slave architecture

There are several basic functions used in Parallel Text Searching Algorithm.

- MPI_Init(&argc,&argv);
- MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
- MPI_Comm_size(MPI_COMM_WORLD,&mysize);
- MPI_File_close(&myfile);
- MPI_Finalize();

- To open file (file name – search.txt)

```
MPI_File_open(MPI_COMM_WORLD, "search.txt",
              MPI_MODE_RDONLY,
              MPI_INFO_NULL, &myfile);
```

- To divide file into several chunk to be distributed to available nodes

```
MPI_File_get_size(myfile, &size);
size = size / sizeof(int);
c = size/mysize + 1;
buf = (int *) malloc(c * sizeof(int));
```

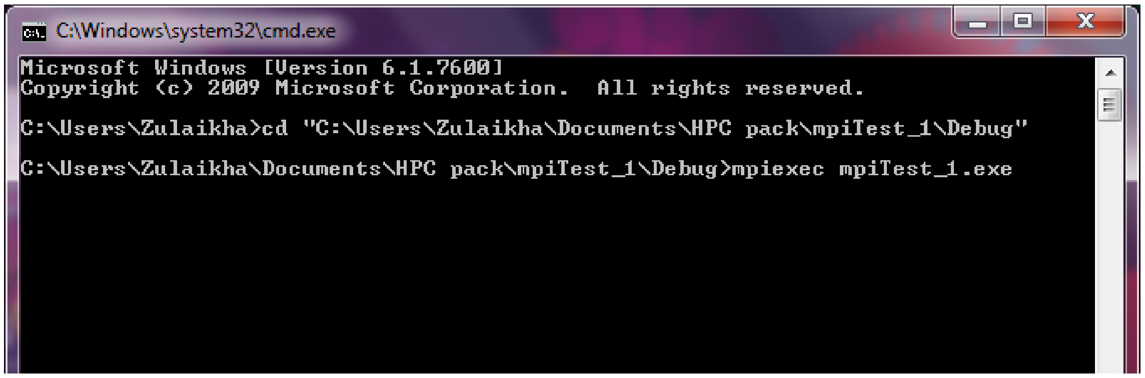
- To scatter data

```
  chunk = (int *)malloc(c * sizeof(int));
  MPI_Scatter(buf, c, MPI_INT, chunk, c, MPI_INT, 0,
MPI_COMM_WORLD);
  free(buf);
  buf = NULL;
```

- To compute execution time

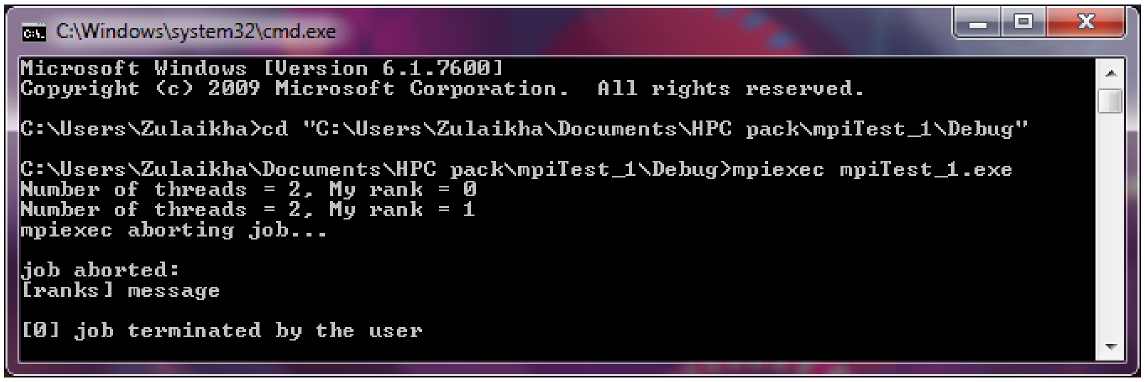
```
MPI_Barrier(MPI_COMM_WORLD);
elapsed_time = - MPI_Wtime();
```

<u>Below are some print screens for Parallel Text Searching Algorithm</u>



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Zulaikha>cd "C:\Users\Zulaikha\Documents\HPC pack\mpiTest_1\Debug"

C:\Users\Zulaikha\Documents\HPC pack\mpiTest_1\Debug>mpiexec mpiTest_1.exe
```

Print Screen 1



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Zulaikha>cd "C:\Users\Zulaikha\Documents\HPC pack\mpiTest_1\Debug"

C:\Users\Zulaikha\Documents\HPC pack\mpiTest_1\Debug>mpiexec mpiTest_1.exe
Number of threads = 2, My rank = 0
Number of threads = 2, My rank = 1
mpiexec aborting job...

job aborted:
[ranks] message

[0] job terminated by the user
```

Print Screen 2

Flowchart for Parallel Text Searching Algorithm

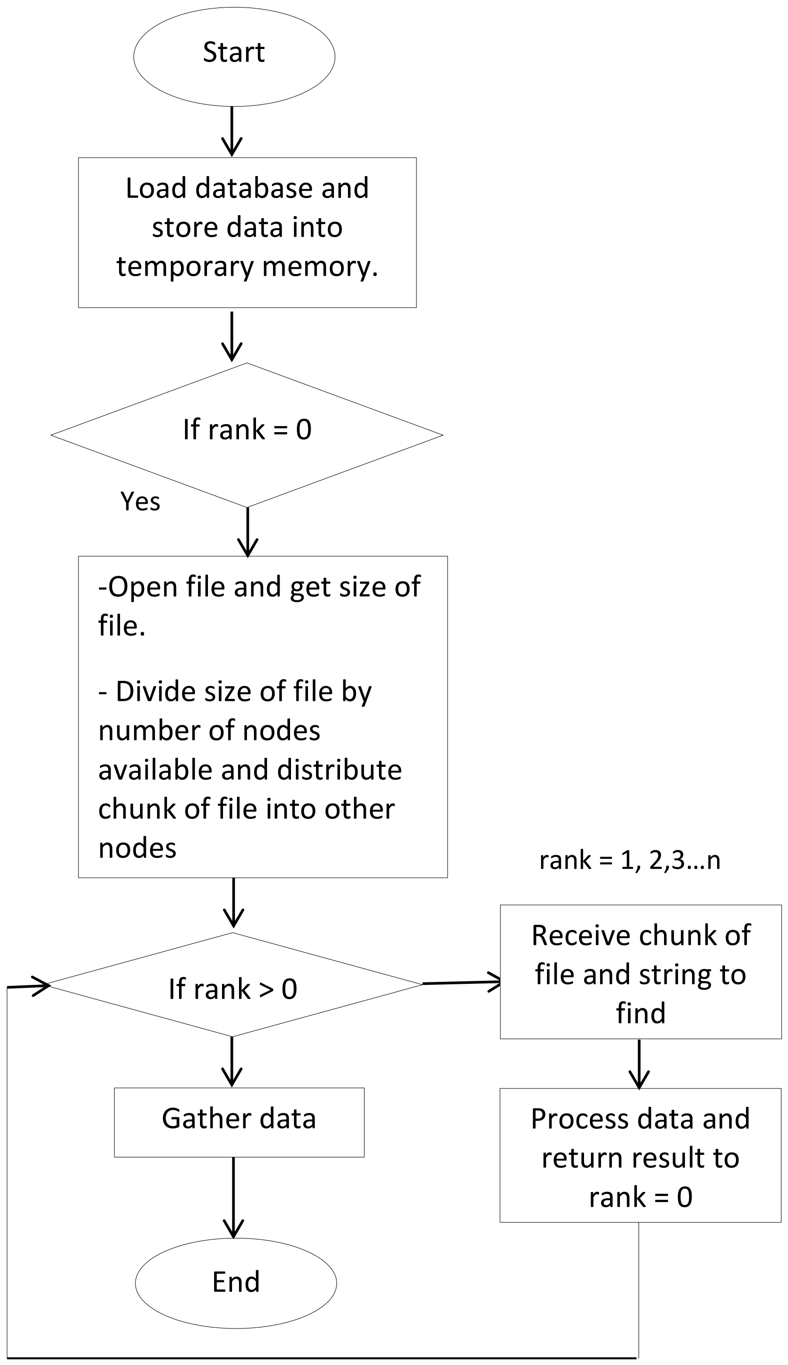

Start
Load database and store data into temporary memory.
If rank = 0
Yes
-Open file and get size of file.
- Divide size of file by number of nodes available and distribute chunk of file into other nodes
rank = 1, 2,3…n
If rank > 0
Receive chunk of file and string to find
Gather data
Process data and return result to rank = 0
End

Figure 11: Flowchart for Parallel Text Searching Algorithm

Result for Parallel Text Searching Algorithm

Number of node	Power (W)	Time (s)
2	4	0.20
4	6	0.12
6	18	0.06

Figure 12: Result for Parallel Text Searching Algorithm

Energy Consumption Graph on Parallel Text Searching Algorithm

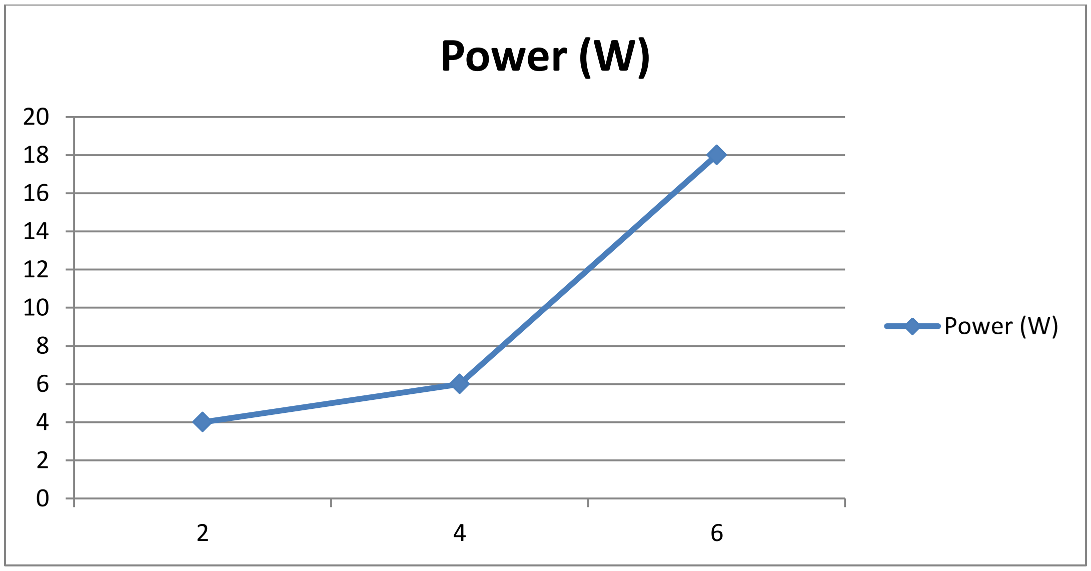


Figure 13: Number of node against Power consumed (W)

<u>Performance Graph on Parallel Text Searching Algorithm</u>

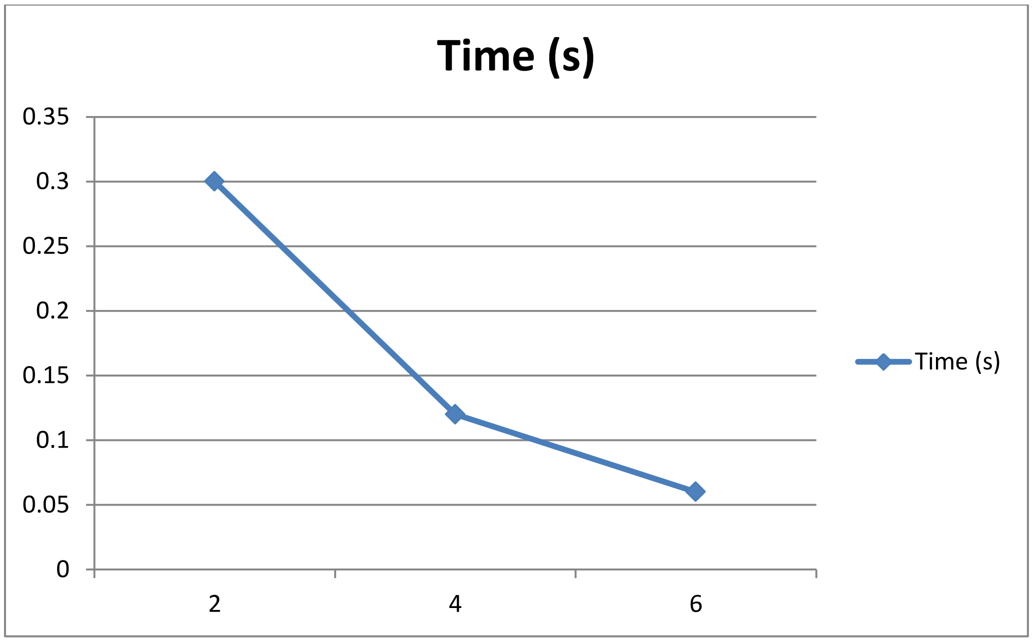


Figure 14: Number of node against Time taken to process data (s)

Based from result on Parallel Text Searching Algorithm, there are larger amount of energy consumed by this parallel experiment. However, it is approved that the performance of parallel experiment is far better than single node performance.

I further this experiment to identify optimal node for Text Searching Algorithm in trade-off between energy consumption and performance.

1 node		2 nodes		4 nodes		8 nodes	
Time (s)	Power (W)	Time (s)	Power (W)	Time (s)	Power (W)	Time (s)	Power (W)
4	10	2	27	1.2	36	0.83	44
3.3	8	1.85	23	1.0	31	0.8	40
3	10	1.3	20	0.9	29	0.8	39
2.9	12	1.2	18	0.85	28	0.75	37
2.9	12	0.9	16	0.7	27	0.5	33

Figure 15: Result for optimal node in Parallel Text Searching Algorithm

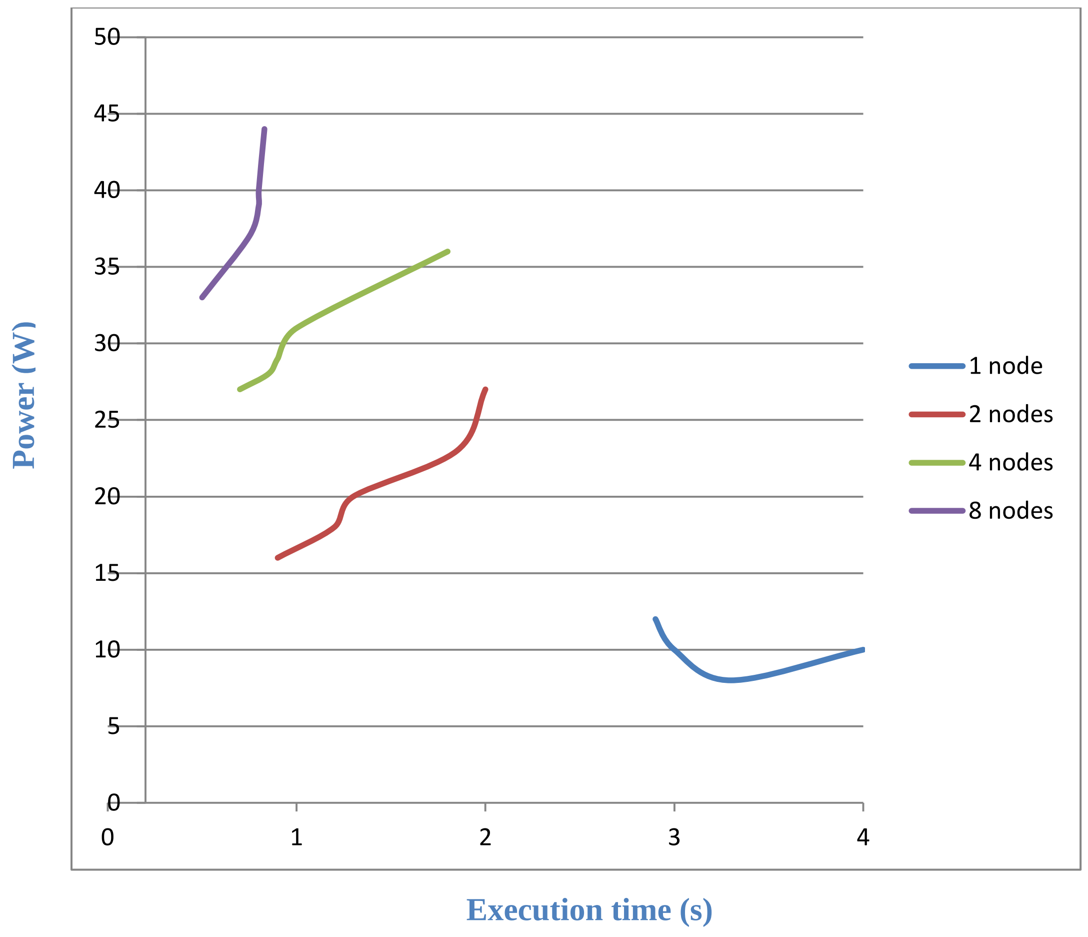


Figure 16: Graph for optimal node in Parallel Text Searching Algorithm

Based from data on Figure, we found out Text Searching Algorithm is experiencing almost to a good speedup level. This is because the fastest node with execution time 0.5 seconds is when using 8 nodes but they consume 33 watts or power. However by using 4 numbers of nodes, the energy consumed is reduced to 27 watts which is equivalent to 20% of power reducing. Then, time taken is increased to 0.7 seconds which is 20% of increased in time taken. For nodes 2 with 16 watts of power consumed can save up to 40% of energy usage compared to nodes of 4 and the execution time is increase only 20%. Thus, it showed that with node of 2 can reduce energy usage more rather than increase of time execution.

CONCLUSION

This paper investigates the trade-off between energy and performance in both serial and parallel using Text Searching Algorithm. This algorithm is chosen due to high frequency of usage in our daily activity. Measuring this searching activity will help us to learn amount of energy we produce every day. Besides, we can also find out the optimal node for parallel computing to execute searching activity

Based on the series of experiments, we found out that the increasing number of nodes in parallel computing can reduce execution time but consumed more energy. In contrast, serial processing computing which is using a single processor consumed less energy but it will take longer time to compute data. However, there is one part we discover that amount of energy produced is at minimum in a given of shorter execution time. This node is considered as an optimal node to execute Text Searching Algorithm.

For future work, we believe this experiment can be continued on more complex Searching Algorithm that mostly applied in Web Search Engine. To make experiments more accurate, a more complex and larger data set can be used to study the energy consumption in real world. Besides, other metrics like slack and misses per operation (MPO) can be used to enhance accuracy of experiment.

REFERENCES

1. Stout, Q. F., (2000-2012): What is Parallel Computing? A Not Too Serious Explanation. University of Michigan. Retrieved October 4, 2012, from http://web.eecs.umich.edu/~qstout/parallel.html
2. Barney, B., (2012): Introduction to Parallel Computing. Lawrence Livermore National Laboratory. Retrieved October 3, 2012, from https://computing.llnl.gov/tutorials/parallel_comp/
3. Hoffman, F. M. and Hargrove W. W.: High Performance Computing: An Introduction to Parallel Programming. Oak Ridge National Laboratory in Oak Ridge, Tennessee. Retrieved October 6, 2012, from http://climate.ornl.gov/~forrest/osdj-2000-11/
4. Murphy, B. (2011): Science and The Future of Computing: Parallel Processing to Meet Tomorrow's Challenges. The National Academic Press. Retrieved October 6, 2012, from http://notes.nap.edu/2011/04/13/science-and-the-future-of-computing-parallel-processing-to-meet-tomorrows-challenges/#.UIdnHG_R6So
5. Lars S. Nyland, Jan F. Prins, et al. (2000): A Design Methodology for Data-Parallel Application (Vol 26). Retrieved October 12, 2012, from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=844491
6. Elwaer A., Harrison A. et al. (2011): Attic: A case Study for Distributing Data in BOINC Projects. Anchorage, Alaska USA. Retrieved October 4, 2012, from http://ieeexplore.ieee.org/stamp/
7. Guru 99. Agile Testing, Scrum and eXtreme Programming. Retrieved October 4, 2012, from http://www.guru99.com/agile-scrum-extreme-testing.html

8. Sauter, V. L. (2012). Agile Methodologies. University of Missouri. Retrieved October 4, 2012, from http://www.umsl.edu/~sauterv/analysis/6840_f09_papers/Nat/Agile.html

9. Demko, A. B. (2011).A Parallel Library for User-friendly Applications. University of Manitoba. Retrieved October 4, 2012, from http://mspace.lib.umanitoba.ca/handle/1993/5046

10. Solomon, S. (2012). Parallel Algorithm Design and Implementation of Regular/ Irregular Problems: An In-depth Performance Study on Graphics Processing Units. University of Manitoba. Retrieved October 4, 2012, from http://mspace.lib.umanitoba.ca/handle/1993/5046

11. Vedyadhara, (2010). Classification of Parallel Computers. Retrieved October 4, 2012, from http://mspace.lib.umanitoba.ca/handle/1993/5046

12. You, S., Lu, H. and et al. (2009) Research on Rule Definition and Engine for General Text Processing. Beijing. Retrieved October 4, 2012, from http://ieeexplore.ieee.org/stamp/

13. Polponij, J. (2009). An ontology-based Text Processing Approach for Simplifying Ambiguity of Requirement Specifications. University of Wollongong, Australia. Retrieved October 4, 2012, from http://ieeexplore.ieee.org/stamp/

14. Create a Virtual Campus Supercomputing Center (VCSC). (2012). Retrieved October 4, 2012, from http://boinc.berkeley.edu/vcsc.php

15. Balasubramanian, K. and Lowenthal, D. K. (2002). Efficient Support for Pipelining in Distributed Shared Memory Systems. University of Georgia. Retrieved October 4, 2012, from http://boinc.berkeley.edu/vcsc.php

16. Hwang, K. (2005). Advanced Computer Architecture: Parallelism, Scalability, Programmability. California: McGraw-Hill.

17. Wilson E. K. (2007). Using Computers at Home, Volunteers Participate in Big Science. Retrieved 9 November, 2012 from http://pubs.acs.org/isubscribe/journals/cen/85/i14/html/8514sci2.html

18. Constantinescu, Z. and Vladoiu, M. (2011). Using Open Source Desktop Grids in Scientific Computing and Visualization. PG University of Ploiesti, Romania. Retrieved 21 November, 2012 from http://www.intechopen.com

19. Trifa, Z., Labidi, M. and et al. (2011). Arabic Cursive Characters Distributed Recognition using the DTW Algorith on BOINC : Performance Analysis. University of Sfax, Tunsia. Retrieved 21 November, 2012 from http://ijacsa.thesai.org/

20. Tuovinen, L. and Röning, J. (2010). Everybody wins: Challenges and Promises of Knowledge Discovery through Volunteer Computing. University of Oulu, Finland. Retrieved 29 November, 2012 from http://www.journalogy.org/

21. Amato, N. M. and Dale, L.K. (1999). Probabilistic Roadmap Methods are Embarrassingly Parallel. A & M University, Texas. Retrieved 29 November, 2012 from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=770055&tag=1

22. De, M., De, S. and et. al. (2008). Parallel Architecture and Algorithms for Space Weather Prediction. University of Kalyani, India. Retrieved 29 November, 2012 from http://nopr.niscair.res.in/handle/123456789/2497

23. Kasabov, A. and Kerkwijk, J. V. (2011). Distribution GPU Password Cracking. University of Amsterdam. Retrieved 29 November, 2012 from http://staff.science.uva.nl/~delaat/rp/2012-2013/index.html

24. Capizzi, S., (2008). A Tuple Space Implementation for Large-Scale Infrastructures. University of Bologna, Padova. Retrieved 13 December, 2012 from www.cs.unibo.it/pub/techreports/2008/2008-07.pdf

25. Barney, B., (1995). Introduction to Parallel Programming. Maui High Performance Computing Center. Retrieved 13 December, 2012 from

http://phi.sinica.edu.tw/tyuan/old.pages/pcfarm.19991228/aspac/instruct/workshop/html/parallel-intro/ParallelIntro.html

26. Crochemore, M. and Rytter, W. (1994). Text Algorithms. Retrieved 13 December, 2012 from http://monge.univ-mlv.fr/~mac/REC/B1.html

27. Jiang, L. Z. and Zeng, Z., (2011). A BOINC based System for Global Topographic Structure Extraction Using SRTM Digital Elevation Models. Beijing. Retrieved 13 December, 2012 from http://wenku.baidu.com

28. Mighell, K. J. (2009). A Parallel-processing Computational Framework for Embarrassingly-parallel Image-analysis Algorithms. North Cherry Avenue, Tucson. Retrieved 13 December, 2012 from http://arxiv.org/abs/1008.2192

29. (2007). Research projects involving BOINC. Retrieved 13 December, 2012 from http://boinc.berkeley.edu/trac/wiki/ResearchProjects

30. Freeh V. W., Lowenthal D. K. and et. al. (2007). Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications. Retrieved 13 December, 2012 from http://ieeexplore.ieee.org/

31. Külekci M. O. (2007). TARA: An Algorithm for Fast Searching of Multiple Patterns on Text Files. Anakara, Turkey. Retrieved 11 March, 2013 from http://ieeexplore.ieee.org/

32. Obaja A. (2012). Text Searching Algorithms (CS3230R Presentation). Retrieved 11 March, 2013 from www.comp.nus.edu.sg/~rahul/allfiles/aldrian-text-searching.pdf

33. Son S. W., Malkowski K. and et. al.(2007). Reducing Energy Consumption of Parallel Sparse Matrix Applications Through Integrated Link/CPU Voltage Scaling. Pennsylvania, United States. Retrieved 11 March, 2013 from www.eecs.northwestern.edu/~sson/paper/TJS07.pdf

34. CERN opeblab. (2008). Reducing Data Center Energy Consumption. Intel Coporation, USA. Retrieved 20 March, 2013 from http://www.cs.berkeley.edu/~istoica/classes/cs294/09/CERN_Whitepaper_r04.pdf

35. Carbonfund.org Foundation. How to Reduce your Carbon Footprint. Bethesda Metro Center, Bethesde. Retrieved 20 March, 2013 from http://www.carbonfund.org/reduce

36. Brin S. and Page L. (1999). The Anatomy of a Large-Scale Hypertextual Web Search Engine. Stanford University, Stanford. Retrieved 2 March, 2013 from http://infolab.stanford.edu/~backrub/google.html