CHAPTER 4

DESIGN ANALYSIS

4.1 General design software

This reseach contains CPU and GPU comparison using 2d matrix. testing is done by giving the task of adding, multiplication and flip (horizontal and vertical) matrix that will be tested on the CPU and GPU. the structure data that is used is 2d array.

The matrix addition algorithm used in this project is the left distribution **A(B+C) = AB + AC** and the psuedo code is

```
1. BEGIN program start by entering elements of matrix a and b //
   matrix 2D//
2. ENTER row and col for matrix a and b
3. SET loop for row using i and loop column using j
4. ADD the elements of a and b in column and store in matrix c
5. PRINT result in matrix c
6. TERMINATE the program
```

The Multiplication matrix algorithm used in this project is scalar multiplication:

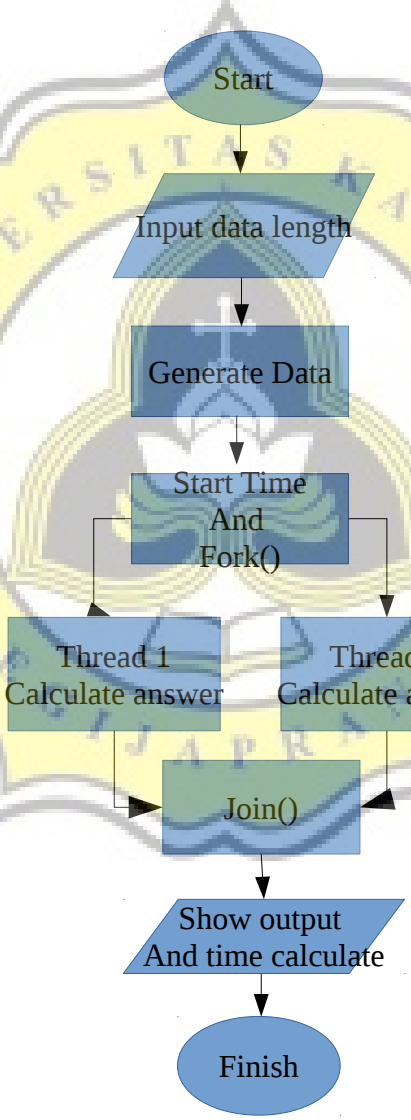
$$\sum_k A_{ik}(B_{kj} + C_{kj}) = \sum_k A_{ik}B_{kj} + \sum_k A_{ik}C_{kj}$$

Below is the pseudo code for the implementation of matrix multiplication

```
1. BEGIN program start by entering elements of matrix a and b //
   matrix 2D//
2. ENTER row and col for matrix a and b
3. SET loop for row using i and loop column using j
4. DECLAR temp variable to store calculation result
5. SET loop helper for row matrix 2
6. MULTIPLICATION the elements to temp
7. END loop helper
8. SET the result matrix c using temp
9. PRINT result in matrix c
10.   TERMINATE the program
```

The last test by testing 2d Flipping matrix horizontally or vertically using algorithm **B = A , A = MAX – index – 1, MAX – index – 1 = B**.

Here is a flowchart for general program implementation in java. the program will start by filling the length of the data, the program will generate dummy data at random and then the program will divide the task (fork) into 2 threads and solve them separately at the same time. So when finished the program will combine data from the thread (serialized/join) and display the output answer and calculation time to overcome it.



4.2 Testing Scenario.

To compare CPU and GPU the following are details of processor and GPU to be used in (data presented in table form).

Table 4.1: Details of GPU and CPU

No	Item Name	Vendor Used		
1	Processing Unit	Intel Dual Core E2110	Intel Core I5 4460	Nvidia 1050Ti
2	Clock Speed	1.6 Ghz	3.2 Ghz	1.2Ghz
3	Memory	4096MB	4096MB	4096MB
4	Core	2	4	768
5	OS	Ubuntu 14.04 LTS		
6	HDD	SSD Samsung Evo 850 120GB		

1. The effect of matrix dimension with time computation in CPU and GPU. (in CPU will test in different processing unit).

   To get matrix data dimension effect with time calculation using CPU and GPU. Implementation is done by assigning multiplication, addition and reversal (horizontal and vertical) and testing will be tested from matrices with dimensions 500x500,1000x1000,5000x5000 and 10000x10000. Each dimension will be tested to different PU ie Dual Core E2110, Core i5 4460 and Nvidia 1050 Ti in Asynchronous programming and Synchronous programming. every time testing process of work will be monitored. For calculation of time monitor in Java programming using this method

```
1. set variable time start using nanoTime
2. Function to do /*
3. ...................
4. ...................
5. */
6. set variable time to end using nanoTime
7. print out put time using alogirithm (time finish- time
   start)/1000000
```

in CUDA programming author used CUDA event method

1. **create variable type float inside class (in this project name time)**
2. **create variable type cudaEvent_t start and stop**
3. **point pointer address to start and stop variable use cuda eventEventCreate(&namevariable)**
4. **start the record using cudaEventRecord**
5. **create block and thread**
6. **call kernel to do**
7. **nameKernel<<<....>>>(parameter)**
8. **stop the record using cudaEventRecord**
9. **point pointer result time to time from start and stop variable type cudaEvent_t use cudaEventElapsedTime**

For each dimension matrix will be tested 3 times to get the best time of 3 experiments.

2. The effect of matrix dimension with processing unit usage in CPU and GPU. (in %)

   The effect of the dimensional matrix on the processing unit (CPU and GPU) can be determined by testing the multiplication, addition and flip (horizontal and vertical) matrix with dimensions 500x500,1000x1000,5000x5000 and 10000x10000. Each dimension will be tested to different PU ie Dual Core E2110, Core i5 4460 and Nvidia 1050 Ti in Asynchronous programming and Synchronous programming. Monitoring PU on CPU Author using htop and Nvidia The author used Nvidia System Management Inteface (Nvidia-smi). For each dimension matrix will be tested 3 times to get the best time out of 3 tests.

3. The effect of matrix dimension with memory usage in CPU and GPU .(in %)

   The effect of dimension matrix on memory usage on CPU and GPU can be determined by testing the matrix multiplication, addition and flip (horizontal and vertical) with dimensions 500x500,1000x1000,5000x5000 and 10000x10000 from the addition of dimension matrix can indicate

whether there is any significant increase or not. Each dimension of the matrix will be tested to different PUs ie Dual Core E2110, Core i5 4460 and Nvidia 1050 Ti in Asynchronous programming and Synchronous programming. Monitoring PU on Author CPU using htop and Nvidia The author uses Nvidia System Management Inteface (Nvidia-smi). For each dimension matrix will be tested 3 times to get the best time out of 3 tests.

4. The effect of time on the number of matrix elements in performing different tasks on GPU.

   To determine the effect of speed to solve against different lengths of matrix elements in assigning different tasks using GPU by testing the matrix multiplication, addition and flip (horizontal and vertical) with dimensions 500x500,1000x1000,5000x5000 and 10000x10000 from the addition of dimension matrix can indicate whether there is any significant increase or not. Each dimension of the matrix will be tested to different PUs ie Dual Core E2110, Core i5 4460 and Nvidia 1050 Ti in Asynchronous programming and Synchronous programming. Monitoring PU on Author CPU using htop and Nvidia The author uses Nvidia System Management

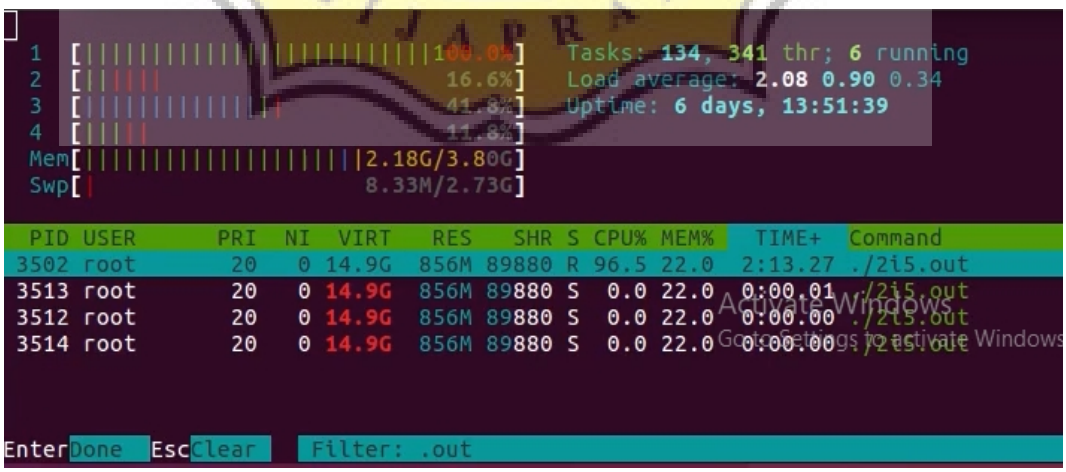


Illustration 4.1: Htop in Terminal

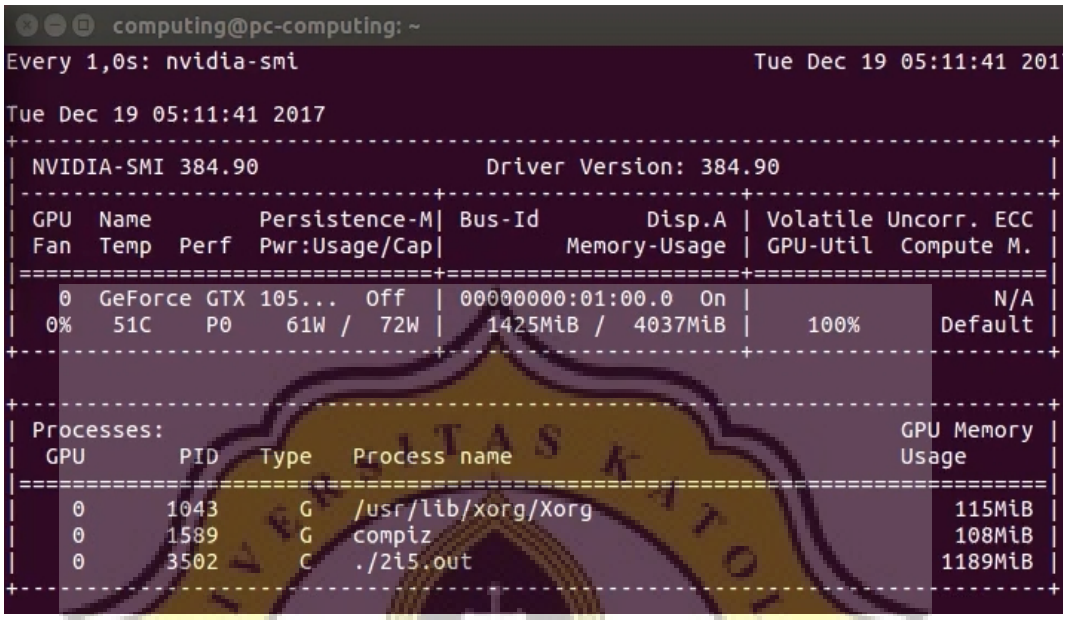


Illustration 4.2: Nvidia SMI