

LP1 Assignment HPC 1

Date: 9th August, 2020

Title: Parallel Computing using CUDA

Problem Definition:

- a) Implement Parallel Reduction using Min, Max, Sum and Average operations.
- b) Write a CUDA program that, given an N-element vector, find-
 - The maximum element in the vector
 - The minimum element in the vector
 - The arithmetic mean of the vector
 - The standard deviation of the values in the vector.

Test for input N and generate a randomized vector V of length N (N should be large). The program should generate output as the two computed maximum values as well as the time taken to find each value

Learning Objectives:

- Learn Parallel Decomposition of Problem
- Learn Parallel Computing using Cuda

Learning Outcomes:

- I will be able to perform parallel computing using CUDA libraries
- I will be able to decompose a problem into sub problems, learn how to use GPUs, and solve problems using threads on GPU

S/W & H/W Packages:

1. Operating System : 64-bit Open source Linux or its derivative
2. Programming Language: C/C++
3. NVidia CUDA-enabled GPU
4. CUDA API
5. Google Colaboratory(uses Tesla K80 GPU)

Related Mathematics:

Let S be the system set: $S = \{s; e; X; Y; F_{me}; DD; NDD; F_c; S_c\}$

s =start state

e =end state

X =set of inputs $X = \{X_1\}$ where:

X_1 = Elements of Vector

Y = Output Set (Minimum / Maximum / Mean / Sum / Standard Deviation)

F_{me} is the set of main functions $F_{me} = \{f_1, f_2, f_3\}$ where:

f_1 = decomposition function

f_2 = function to find Minimum / Maximum / Sum

f_3 = function to merge results

f_4 = function to Mean / Standard Deviation

DD = Deterministic Data Vector of elements

NDD =Non-deterministic data No non deterministic data

F_c =failure case: No failure case identified for this application

Concepts Related to Theory:

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called decomposition. Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem. Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size

Mean:

Sum of all entries divided by number of entries.

Standard Deviation:

The standard deviation measures variability and consistency of the sample or population. In most real-world applications, consistency is a great advantage.

Formulas:

$$\text{Mean} = \frac{\sum fx}{\sum f}$$

$$\text{Standard deviation} = \sqrt{\frac{\sum fx^2}{\sum f} - \left(\frac{\sum fx}{\sum f}\right)^2}$$

References:

1. CUDA Documentation: <https://docs.nvidia.com/cuda/>

Steps for Execution:

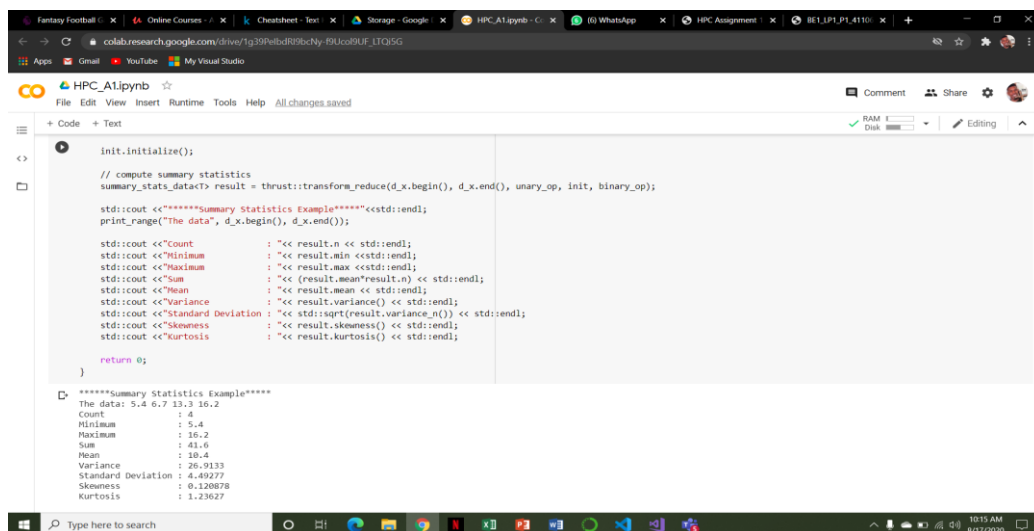
1. Open Google Colaboratory
2. Change Runtime type from CPU to GPU
3. Get nvcc plugin for Jupyter Notebook
4. Start codes with %%cu
5. Compute sum, maximum, minimum, mean and standard deviation of a vector

Code Snippets:

- ```
1. void initialize()
{
 n = mean = M2 = M3 = M4 = 0;
 min = std::numeric_limits<T>::max();
 max = std::numeric_limits<T>::min();
}

2. T variance() { return M2 / (n - 1); }
3. T variance_n() { return M2 / n; }
4. T skewness() { return std::sqrt(n) * M3 / std::pow(M2, (T) 1.5); }
5. T kurtosis() { return n * M4 / (M2 * M2); }
```

## Output:



The screenshot shows a Google Colaboratory notebook titled 'HPC\_A1.ipynb'. The code is written in C++ and uses the Thrust library for parallel processing. It calculates various summary statistics for a dataset. The output is displayed in a structured format.

```
init.initialize();

// compute summary statistics
summary_stats_data<T> result = thrust::transform_reduce(d_x.begin(), d_x.end(), unary_op, init, binary_op);

std::cout << "*****Summary Statistics Example*****" << std::endl;
print_range("The data", d_x.begin(), d_x.end());

std::cout << "Count" << result.n << std::endl;
std::cout << "Minimum" << result.min << std::endl;
std::cout << "Maximum" << result.max << std::endl;
std::cout << "Sum" << (result.mean * result.n) << std::endl;
std::cout << "Mean" << result.mean << std::endl;
std::cout << "Variance" << result.variance() << std::endl;
std::cout << "Standard Deviation" << std::sqrt(result.variance_n()) << std::endl;
std::cout << "Skewness" << result.skewness() << std::endl;
std::cout << "Kurtosis" << result.kurtosis() << std::endl;

return 0;
```

\*\*\*\*\*Summary Statistics Example\*\*\*\*\*  
The data: 5.4 6.7 13.9 16.2  
Count : 4  
Minimum : 5.4  
Maximum : 16.2  
Sum : 41.6  
Mean : 10.4  
Variance : 20.9133  
Standard Deviation : 4.49277  
Skewness : 0.120878  
Kurtosis : 1.23027

## Notebook Link:

[https://colab.research.google.com/drive/1g39PeIbdRI9bcNy-f9Ucol9UF\\_LTQi5G?usp=sharing](https://colab.research.google.com/drive/1g39PeIbdRI9bcNy-f9Ucol9UF_LTQi5G?usp=sharing)

## Conclusion:

I have successfully performed parallel computing using CUDA libraries and computed the mean, maximum, minimum and standard deviation of a vector.

Furthermore, I have also calculated skewness and kurtosis for a data sample using the thrust library in CUDA.