Comparison of CutShort: A Hybrid Sorting Technique using MPI and CUDA

Harshit Yadav, Shraddha Naik, Ashwath Rao B, Shwetha Rai and Gopalakrishna Kini N

Department of Computer Science and Engineering,
Manipal Institute of Technology,
Manipal Academy of Higher Education,
Manipal, Karnataka, India, 576104

Introduction

- Applications of Sorting are found almost everywhere.
- CutShort algorithm which works on the bit count operation.
- The word “CutShort” is self-explanatory i.e. cutting an input array into smaller pieces of array.
- These sub arrays are then fed to Merge sort or quick sort or Insertion sort algorithm

Literature Survey

- Various algorithms like Quick sort, Merge Sort , Insertion sort were considered[1]
- Cut Short Algorithm[4] + (Quick Sort /Merge Sort/ Insertion Sort)
- When this algorithm is used with CutShort Algorithm the time complexity is reduced to n log(dmax) in best case time complexity.
- Proposed model which would combine the CutShort Algorithm with Parallel Computing framework which would greatly improve the efficiency and achieve scale up.

Cutshort Algorithm

- Works Best when the Max Number of digits have a the same number of bits used to represent them

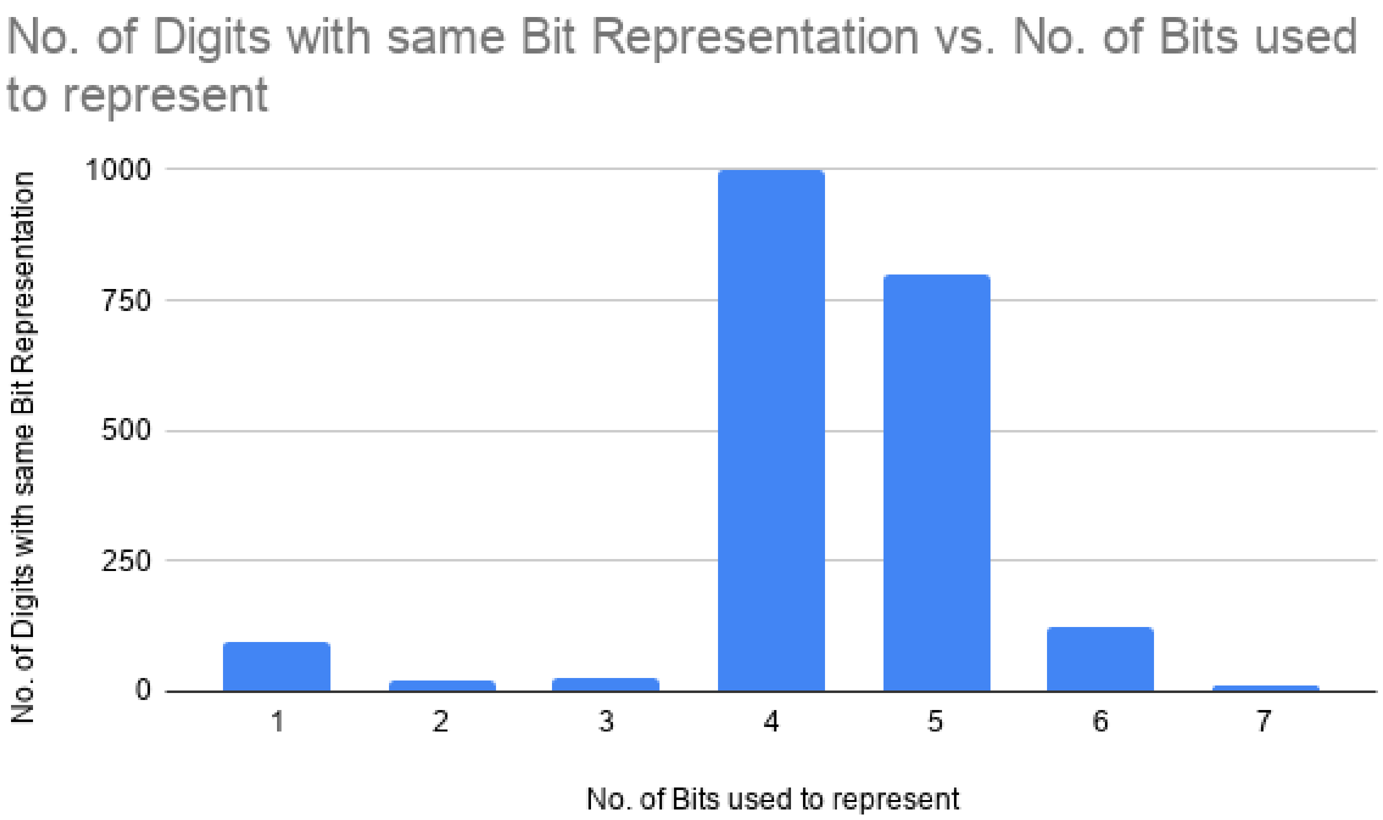


Image :1

Cutshort Algorithm

- Best Case : T(n) = O(n log (n) – n log (Dmax))
- (Dmax) is the maximum number of subarrays obtained from original array.

- Average Case : T(n) = O(n log (n) – n log(D))
- Where D is number of resulting sub-array

- Worst Case : T(n) = O(n.log n)
- In worst case, the time complexity will remain the same O(nlog n)

Methodology

- CutShort algorithm works on four steps as mentioned below.
- 1. Initial step
- 2. Range defining step
- 3. Rearranging
- 4. Sub array sorting

Initial Step

0	1	2	3	4	5	6	7	8	9
22	40	60	52	78	58	18	7	46	82

Image : 2

Number	Binary Representation	O/P of BitCount operation.
22	10110	5
40	101000	6
60	111100	6
52	110100	6

Image : 3

Initial Step

0	1	2	3
3	5	6	7

Image : 4

- Creating a Bit band array storing the count of number of elements having same digits in bit representation

Range defining step

0	1	2	3
3	5	6	7

Image : 5

[0,3)	[3,5)	[5,6)	[6,7)
3	5	6	7

Image : 6

Rearranging

- In this step, the elements of array are rearranged according to the no of bit count values. Integers with same bit count are placed together one after the other in a sequential manner in an array

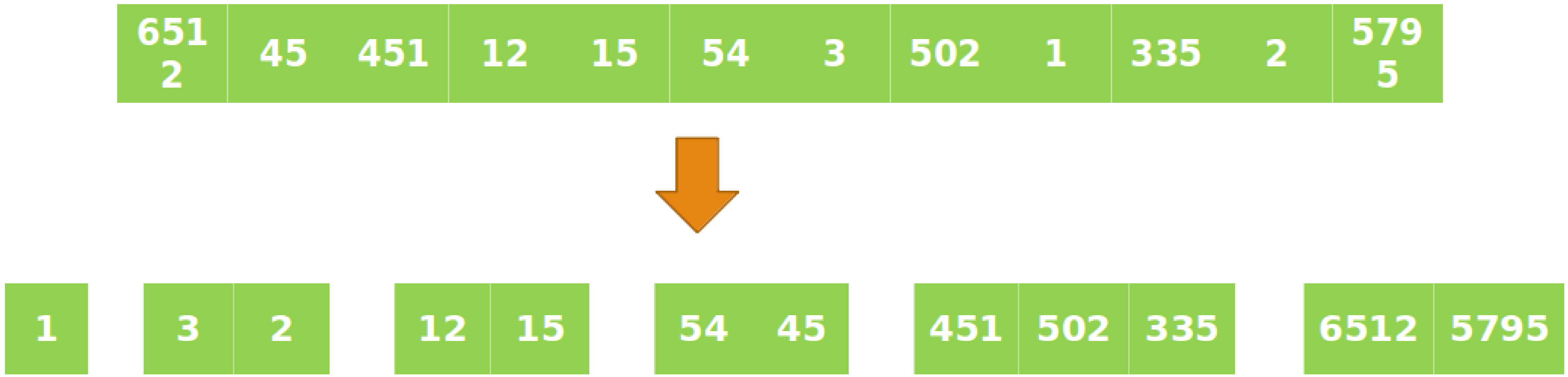
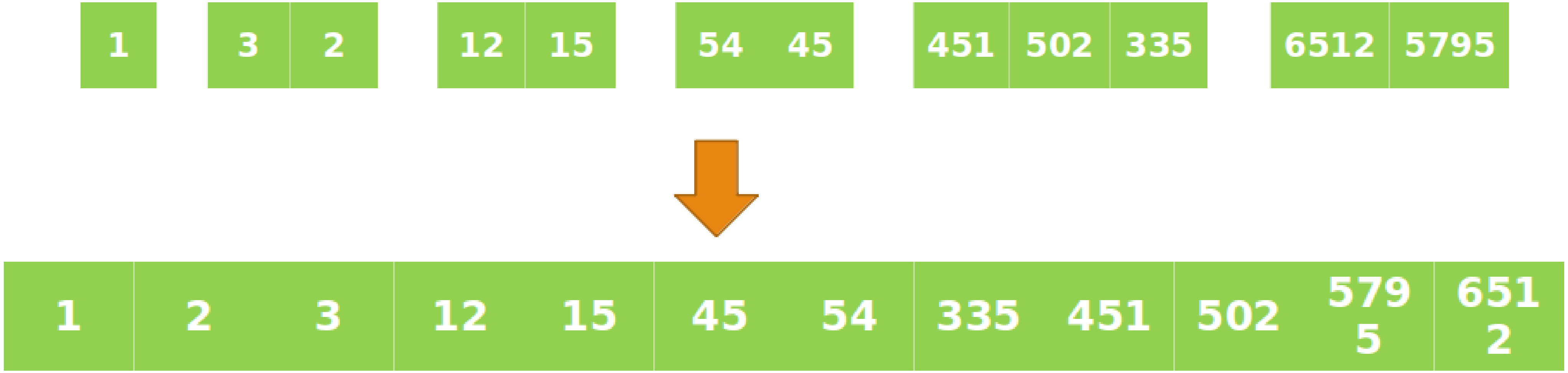


Image : 7

Sub-Array Sorting

- Each of these sub arrays can be sorted using any sorting technique by using the Bitband array to define the different range of bits present in the Input Array. Merge Sort, Quick Sort or Insertion Sort can be used for sorting



Proposed Model

- CutShort Algorithm with MPI
- CutShort Algorithm using CUDA

Message Passing Interface (MPI) Architecture

Message Passing Interface is a framework that creates an environment for parallel programming by providing libraries in C/C++ Programming

MPI gives a large set of inter-process communication.

MPI gives a standard interface to the programmers that allow them to write parallel applications that are supported across various platforms

Proposed Parallel execution Model (CutShort Algorithm with MPI)

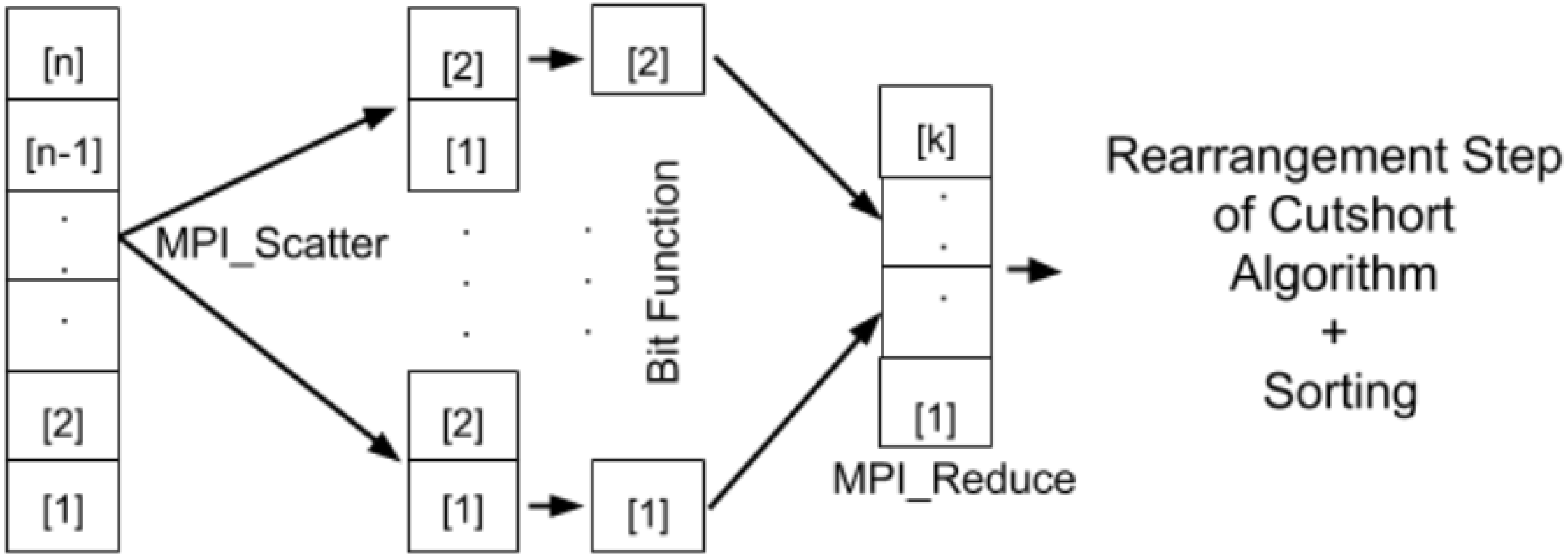
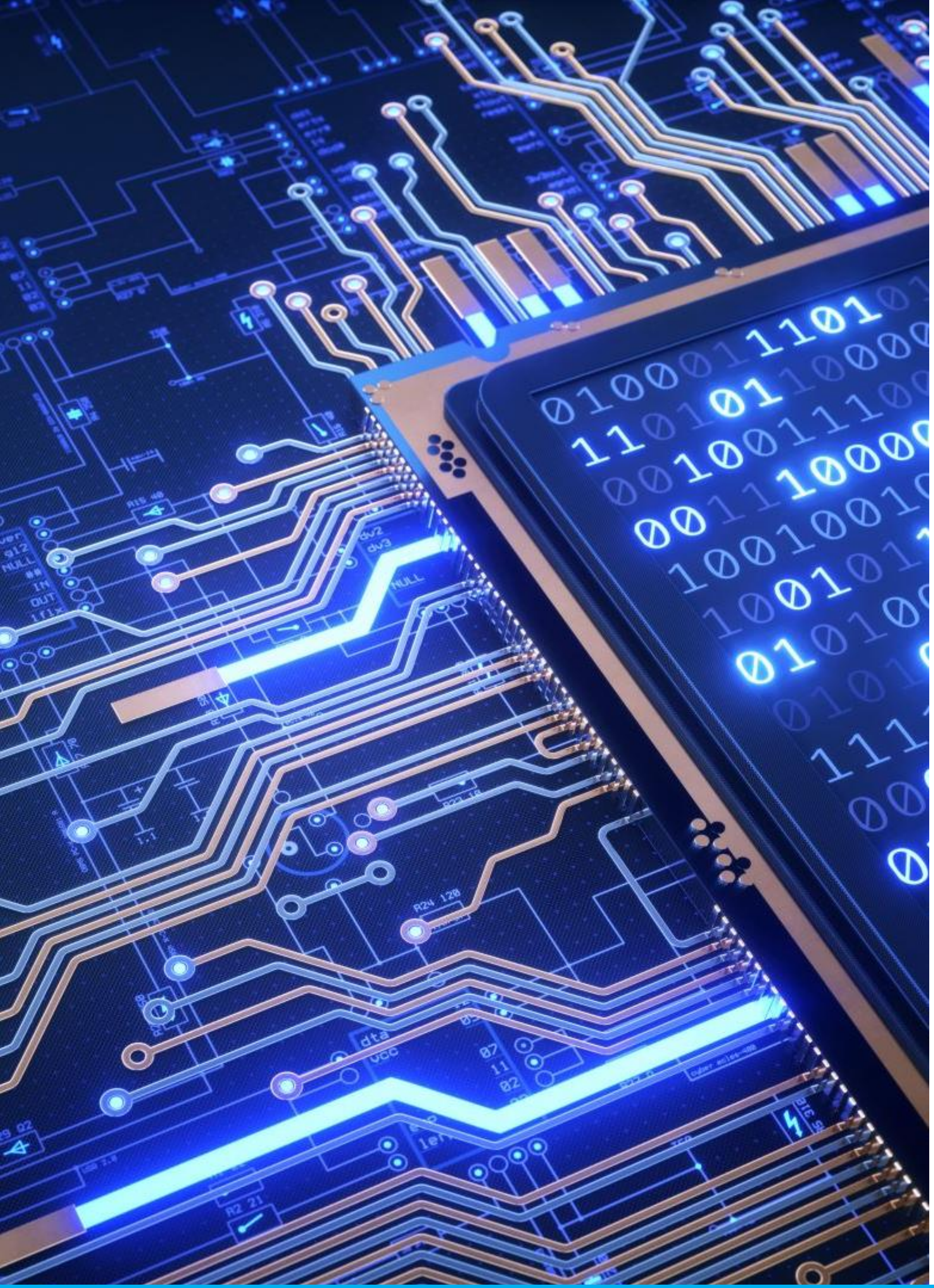


Image : 8



CUDA (Compute Unified Device Architecture)

- CUDA is a parallel computing API Model created by Nvidia
- The CUDA platform is designed to work with programming languages such as C, C++ and FORTRAN.
- CUDA program consists of 1 or more phases that are executed on either CPU or a device such as GPU
- The phases that exhibit little or no parallelism are implemented on host code
- The phase that requires rich amount of parallelism is implemented on device code.

Proposed Parallel execution Model (CutShort Algorithm with CUDA)

- We parallelize the sequential algorithm by sending the input array to the kernel where each thread process the Bitcount function(Initial step of CutShort Algorithm).
- Result is stored in separate bitmap array in global memory which is shared among all the kernel
- Bitmap array obtained after completion of all threads is processed sequentially in Rearranging step of the CutShort algorithm and its result can be sorted with any sequential or parallel sorting algorithm of choice

Experiment

- For the tests a sample space of ten thousand elements was taken which was timed by running sequentially
- Sample Case was Divided into three Categories :

A. Worst Case - sample space are in range from 2^i to 2^{i+1}
B. Random Values - all randomly chosen numbers
C. Best Case - elements can be equally divided into equal bit range buckets

Experiment

- Time Taken by Different Serial Algorithms(in sec)

Test Case	Quick Sort+ CutShort	Insertion Sort + CutShort	Merge Sort + CutShort
1	0.004342	0.035820	0.006833
2	0.004319	0.033692	0.006644
3	0.004367	0.033484	0.006166
4	0.004256	0.028709	0.005831
5	0.004086	0.031815	0.006000
6	0.004119	0.029502	0.005073
7	0.003823	0.026361	0.005370
8	0.003522	0.028389	0.005399
9	0.003773	0.027373	0.005233

Table :1

Experiment

- Time Taken by Different Parallel Algorithms(in sec) in MPI

Test Case	Quick Sort + CutShort	Insertion Sort + CutShort	Merge Sort+ CutShort
1	0.002298	0.027697	0.003462
2	0.002864	0.027697	0.003855
3	0.002310	0.025781	0.003534
4	0.002770	0.022981	0.003526
5	0.002460	0.025772	0.004421
6	0.002301	0.024669	0.003821
7	0.001903	0.024542	0.003662
8	0.002016	0.023974	0.003214
9	0.001900	0.024508	0.003343

Table :2

Experiment

- Time Taken by Different Parallel Algorithms(in sec) In CUDA

Test Case	Quick Sort + CutShort	Insertion Sort + CutShort	Merge Sort + CutShort
1	0.001706	0.016513	0.003116
2	0.002139	0.016508	0.003144
3	0.001684	0.016697	0.003272
4	0.002232	0.015384	0.003096
5	0.001546	0.018439	0.003192
6	0.001604	0.017452	0.003074
7	0.001108	0.015622	0.002464
8	0.001832	0.014504	0.002258
9	0.001296	0.014912	0.002402

Table :3

Experiment

- Time Taken by Different Parallel Algorithms(in sec) In CUDA

Test Case	Quick Sort + CutShort	Insertion Sort + CutShort	Merge Sort + CutShort
1	0.001706	0.016513	0.003116
2	0.002139	0.016508	0.003144
3	0.001684	0.016697	0.003272
4	0.002232	0.015384	0.003096
5	0.001546	0.018439	0.003192
6	0.001604	0.017452	0.003074
7	0.001108	0.015622	0.002464
8	0.001832	0.014504	0.002258
9	0.001296	0.014912	0.002402

Table :4

Result and Analysis

- We achieved a speedup of greater than 30% as shown in Figure by implementing it parallelly in CUDA or MPI individually as compared to running the algorithm sequentially with the same data set.

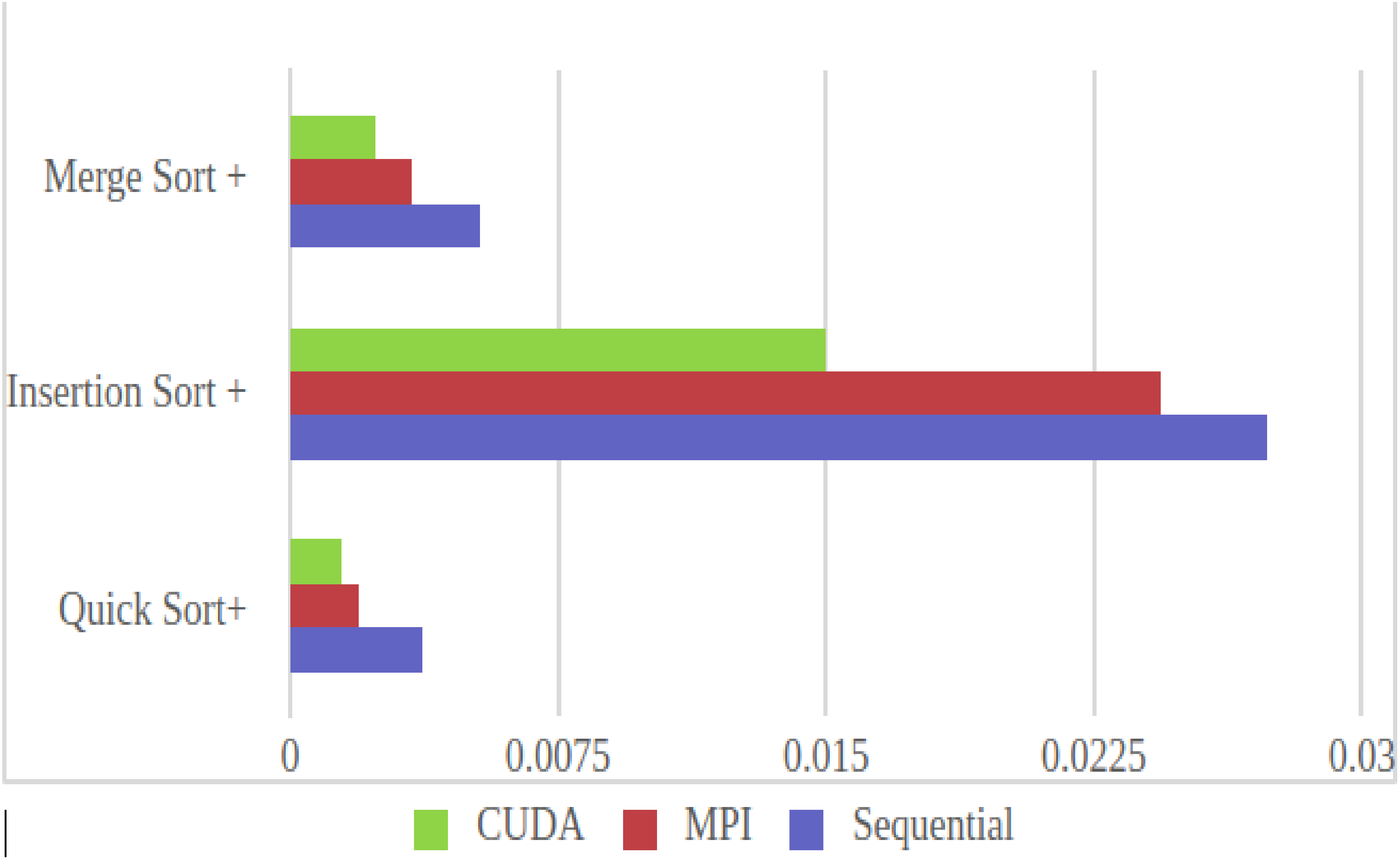


Image : 9

Conclusion

- Speedup achieved is greater in case of CUDA framework due to higher core counts available for parallel processing .
- Using the MPI framework which utilizes the CPU cores a minimum of ~30% of speedup is achieved in the best case scenario .
- The Proposed models provide a significant performance gain over the existing sequential Cutshort algorithms especially in the use case where the data satisfy the cut short algorithm use case criteria
- Even higher speedup can be achieved if the sorting algorithm used is also implemented in parallel

References

1. Thomas H. Coreman, Charles E. Leiserson and Ronald L. Rivest, Introduction to Algorithms, MIT Press, Third edition, 2009.
2. Traditional insertion algorithm – en.wikipedia/wiki/Insertion_sort, three lines implementation and five-lines optimized version by Jon Bentley (1999), Programming Pearls. Addison-Wesley Professional.
3. Bender, Michael A; Farach-Colton, Martín; Mosteiro, Miguel (2006), Insertion Sort is O(n log n) SUNYSB; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.3758
4. A. Garg, S. Goswami and V. Garg, "CutShort: A hybrid sorting technique", 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016. Available: 10.1109/ccaa.2016.7813705 .
5. Link: en.wikipedia.org/wiki/Sorting_algorihm

References

6. Kirk, D., & Hwu, W. Programming massively parallel processors.
7. Using MPI: Portable parallel programming with the message-passing interface. (2000). Computers & Mathematics With Applications, 40(2-3), 419. doi: 10.1016/s0898-1221(00)90207-4
8. RCT Lee, SS Tseng, RC Chang and YT Tsai, Introduction to the Design and Analysis of Algorithms, Mc Graw Hill, 2005
9. Kazennov, A. (2010). Basic concepts of CUDA technology. Computer Research And Modeling, 2(3), 295-308. doi: 10.20537/2076-7633-2010-2-3-295-308.
10. W. Gropp, E. Lusk and A. Skjellum, Using MPI. Cambridge, Massachusetts: TheMIT Press, 2014.

References

11. Zhaoxuan Shen, Jianjian Song and Wenjun Zhuang, "Speedup improvement on general connectivity computation by algorithmic techniques and parallel processing", Proceedings High Performance Computing on the Information Superhighway. HPC Asia '97. Available: 10.1109/hpc.1997.592241

12. S. Rastogi and H. Zaheer, "Significance of parallel computation over serial computation", 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016. Available: 10.1109/iceeot.2016.7755106

13. X. Sun and L. Ni, "Another view on parallel speedup", Proceedings SUPERCOMPUTING '90. Available: 10.1109/superc.1990.130037

Thank You

Any Queries ?