Contrastive Analysis of Bubble & Merge Sort Proposing Hybrid Approach

Sehrish Munawar Cheema, Nadeem Sarwar, Fatima Yousaf
School of Computing & IT
UOG Sialkot Sub Campus, Sialkot, Pakistan
Sehrish.munawar@uogsialkot.edu.pk, Nadeem_srwr@yahoo.com, Fatima.yousaf@uogsialkot.edu.pk

Abstract—**A sorting algorithm is one that puts elements of a list in a certain order. It makes easy searching and locating the information. The most-used orders are numerical order and lexicographical order. An efficient sorting algorithm is that takes less time and space complexity. In this paper I make contrastive analysis of bubble sort and merge sort and tried to show why required some new approach to get best sorting results. In this regard we proposed a hybrid approach that will take minimum number of comparisons, with less time and space complexity to sort. Divide and conquer (merge sort) with bi-directional bubble sort approach is used and an example data is sorted with such hybrid approach.**

***Keywords*— Sorting, Bubble Sort, Merge Sort, Hybrid Approach, Bi-directional bubble sort, auxiliary array, time & space complexity.**

I. INTRODUCTION

Sorting is ordering a list of objects. We can distinguish two types of sorting. If the number of objects is small enough to fits into the main memory, sorting is called internal sorting. If the number of objects is so large that some of them reside on external storage during the sort, it is called external sorting.

A calculation is portrayed that permits log(n) processors to sort n records in a little more than 2n compose cycles, together with appropriate equipment to bolster the calculation. The calculation is a parallel adaptation of the straight consolidation sort. The moderate documents of a serial consolidation sort are supplanted by first-in first-out lines. The processors and lines might be executed in customary strong rationale innovation or in air pocket innovation. A cross breed innovation is additionally suitable.[2]

Sorting has been a significant zone for the algorithmic analysts and numerous assets are contributed to recommend more works for sorting calculations. For this reason, numerous current sorting calculations were seen as far as the effectiveness of the algorithmic many-sided quality.. The proposed work tried on two standard datasets (content document) with various size. The sizes of each of these sub-exhibits are chosen depending on various components with the same number of characters in the information exhibit. We executed MPI utilizing Intel center i7-3610QM,(8 CPUs),using two methodologies (vectors of string and cluster 3D). At last, we get the information structure impacts on the execution of the calculation for that we decision the second approach.[3]

A calculation is an all-around characterized way that takes some contribution to the type of specific qualities, forms them and gives certain qualities as yield. Despite the fact that there is a vast assortment of sorting calculations, sorting issue has bid an extraordinary arrangement of examination; in light of the fact that successful sorting is essential to improve the utilization of other algorithms.A novel sorting calculation in particular „V-Re-Fr (VRF) Sorting Algorithm‟ is proposed to address the impediments of the current mainstream sorting calculations. The objective of this paper is to propose another calculation which will give enhanced usefulness also, lessen calculation complexities. The perceptions upheld by writing study demonstrates that proposed calculation is significantly more proficient as far as number of swaps or cycles than alternate calculations having O(n2) intricacy, like insertion, choice and air pocket sort calculations[8].

Sorting is a vital and broadly concentrated on issue, where the execution time and the required assets for calculation is of great significance, particularly on the off chance that it is managing ongoing information preparing. In this manner, it is critical to ponder and to think about in points of interest all the accessible sorting calculations. In this anticipate, an escalated examination was led on five calculations, to be specific, Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort calculations. Four gatherings of information components were made with the end goal of correlation procedure among the distinctive sorting calculations. All the five sorting calculations are connected to these gatherings. The most exceedingly terrible time multifaceted nature for every sorting system is then processed for every sorting calculation. The sorting calculations were ordered into two gatherings of time multifaceted nature, O (n2) gathering and O(nlog2n) bunch. The execution time for the five sorting calculations of every gathering of information components were registered. The speediest calculation is then controlled by the evaluated esteem for every sorting calculation, which is processed utilizing direct minimum square relapse. The outcomes uncovered that the Merge Sort was more effective to sort

978-1-5090-2000-3/16/$31.00 ©2016 IEEE

information from the Quick Sort for O(nlog2n) time many-sided quality gathering. The Insertion Sort had more effectiveness to sort information from Selection Sort and Bubble Sort for O (n2) bunch. Bubble Sort was the slowest or it was less effective to sort the information. All in all, the effectiveness of sorting calculations can be positioned from most astounding to least as Merge Sort, Quick Sort, Insertion Sort, Selection Sort and Bubble Sort [9].

They consider a straightforward model of uncertain examinations: there exists some $\delta > 0$ such that when a subject is given two components to analyze, if the estimations of those components (as saw by the subject) contrast by at any rate δ, then the correlation will be made effectively; when the two components have values that are inside δ, the result of the examination is eccentric. This model is propelled by both imprecision in human judgment of qualities furthermore by limited yet conceivably antagonistic blunders in the results of brandishing competitions. Our model is firmly identified with various models regularly considered in the psychophysics writing where δ relates to the Just Noticeable Difference (JND) unit or distinction edge. In test brain science, the technique for matched correlations was proposed as a methods for positioning inclinations among n components of a human subject. The strategy requires playing out all (n2) correlations, then sorting components as per the quantity of wins. The substantial number of examinations is performed to counter the conceivably broken basic leadership of the human subject, who goes about as a loose comparator. We demonstrate that in our model the technique for combined correlations has ideal precision, minimizing the blunders presented by the uncertain examinations. Nonetheless, it is additionally inefficient on the grounds that it requires all (n2). We demonstrate that the same ideal insurances can be accomplished utilizing 4n3/2 correlations, and we demonstrate the optimality of our technique. We then investigate the general tradeoff between the sureties on the blunder that can be made and number of examinations for the issues of sorting, max-finding, and choice. Our outcomes give solid lower limits and near ideal answers for each of these issues[10].

In this paper displays a quick, precise, and adaptable FPGA-based issue copying stage, specifically FARAVAM that can be misused for AVF investigation in cutting edge chip. The proposed approach gives deficiency infusion capacities supporting programmed alteration of post-blend net-records and presents a profoundly controllable and recognizable transient issue examination environment. The displayed weakness examination stage utilizing both comprehensive and arbitrary flaw copying approaches, gives valuable data to distinguishing zones undermining dependability to make processors more blame tolerant. We connected our stage for removing the best exchange offs amongst accuracy and rate up in helplessness investigation of MIPS processor. The trial results demonstrate that notwithstanding having high accuracy we acquire around seven requests of extent pace up in examination with recreation based helplessness investigation strategies [11].

One of the crucial issues in register science is requesting a rundown of things. In spite of the fact that there is countless calculations, sorting issue has pulled in a lot of examination, on the grounds that productive sorting is imperative to improve the utilization of different calculations. Sorting includes reworking data into either climbing or sliding request. This paper displays another sorting calculation called Input Sort. This new calculation is examined, actualized, tried and thought about and results were promising [12].

A. Bubble Sort

Bubble sort, sometimes referred to as *sinking sort* [1], the algorithm works by analyzing an array of items from left to right. It compares each adjacent pair of elements; swap them if left item is greater than right. In other words, the largest element has bubbled to the top of the array. The algorithm continuously perform this process till it makes a go through the list without swapping any element that means all elements of array or list has been sorted [4].

The algorithm is basically *comparison sort*, is named due to manner that the larger items "bubble" to the bottom of the list. Though bubble sort is a simple technique but I may slower and impractical when data set becomes large as compared to insertion sort [5]. It can become practical if input array may consist of most sort order and infrequently some items out of order.

```
voidbubbleSort(intar[])
{
for (inti = (ar.length - 1); i>= 0; i--)
  {
for (int j = 1; j ≤ i; j++)
    {
if (ar[j-1] >ar[j])
      {
int temp = ar[j-1];
ar[j-1] = ar[j];
ar[j] = temp;
  } } } }
```

Here is one step of the algorithm. The largest element - 7 - is bubbled to the top:

7, 5, 2, 4, 3, 9
5, 7, 2, 4, 3, 9
5, 2, 7, 4, 3, 9
5, 2, 4, 7, 3, 9
5, 2, 4, 3, 7, 9
5, 2, 4, 3, 7, 9

The worst-case runtime complexity is O(n2).

B. Merge Sort

Merge sort is a kind of divide and conquer approach. Actual concept of this method is insight is; it is more rapid to sort two small arrays or lists then merge the sorted ones instead of sorting a large list in place [6]. The algorithm works as; it splits

down the array from middle into sub-arrays each of size n/2. Its keeps on dividing the array until each sub-array contains single element. It then compares each element from left to right in sequence. Algorithm is recursively called to divide the array into sub-arrays. Mergesort is a procedure that is called on each sub-array to sort it. Finally merge procedure is called to merge each sorted sub-array.

Merge-sort(array A, intp, intr)
if(p<r)
then
q= (p + r) / 2
Merge-sort(A, p, q) //sort A[p…q]
Merge-sort(A, q+1, r)//sort A[q+1..r]
Merge(A, p, q, r)

Merge(array A, intp, intq,intr)
intB[p…r];
inti=k =p;
intj =q+1
while(i<= q) and (j <= r)
do if(A[i] <= A[j])
then B[k++] = A[i++]
elseB[k++] = A[j++]
while(i<=q)
do B[k++] = A[i++]
while(j <= r)
do B[k++] = A[j++]
fori= p to r
do A[i] = B[i]

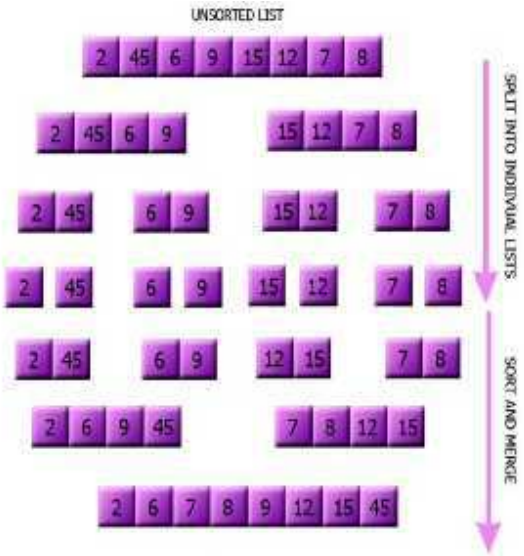

Fig 1: Process of Merge Sort

II. MATERIAL AND METHODS

Contrast between Bubble & Merge Sort:

- Bubble sort algorithm is easy to implement; for best case time complexity is O(n) but it runs O(n2) times both in average and worst case, here 'n' number of items of array that being to be sort. Its O(n2) complexity means that performance slows down dramatically on larger data items in an array. Thus it is avoided to recommend it for such cases. Many sorting algorithms that have nlog(n) running time e.g. merge sort algorithm. Even there exist such O(n2) algorithms that have better performance even in worst cases e.g. insertion sort. Thus bubble sort is not sufficient when number of items to be sort becomes large. It is also inefficient in case of reverse order data items. The only best advantage that its implementation on such a list that has some already sorted items (best-case). Bubble sort should be avoided in the case of large collections but it is a stable sort [10].

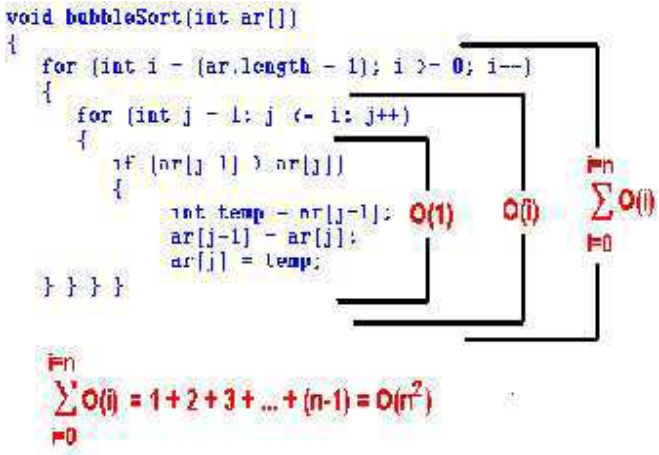

Fig 2: pseudo Code of Bubble Sort

- Merge sort on best and average case time complexity is nlog(n). Its space complexity in worst case is O(n). It requires an extra auxiliary array during sorting. It works better to sort sequential-accessed lists. If there are two equal valued elements in the list then there relative positions would be protected i.e. stable sort. if item A = "book" and item D = "book" then the sorted list will have AC. Unlike quick-sort might not preserve the order. It may be CA [6].

Design Idea of Hybrid method:

The original sort algorithm is one-way adjacent comparison sort. Of course, the direction of sorting from tail to head can also matter. You can also consider an alternative sort design concept, if the comparison of each sort trip would be `a bidirectional comparison of two adjacent elements that is bidirectional bubble sort. Also if heap sort is used for the in-memory part of the merge, its operation can be overlapped with I/O.

Time to copy the auxiliary array can be reduced (but not space) for merge sort algorithm. For it two merge invocations are needed one to copy unsorted input array to auxiliary array and other to copy sorted results to input array. But it a bit recursive and trickery. If recursive calls are arranged in such a way that computation switches roles of input array and auxiliary array at each level [7].

Working:

This method divides large number of data elements array into 7-10 elements smaller lists using merge sort (divide and conquer). These smaller lists are then sorted using a bidirectional Bubble sort technique. Finally all sorted parts are merged into one by invoking merge sort algorithm.

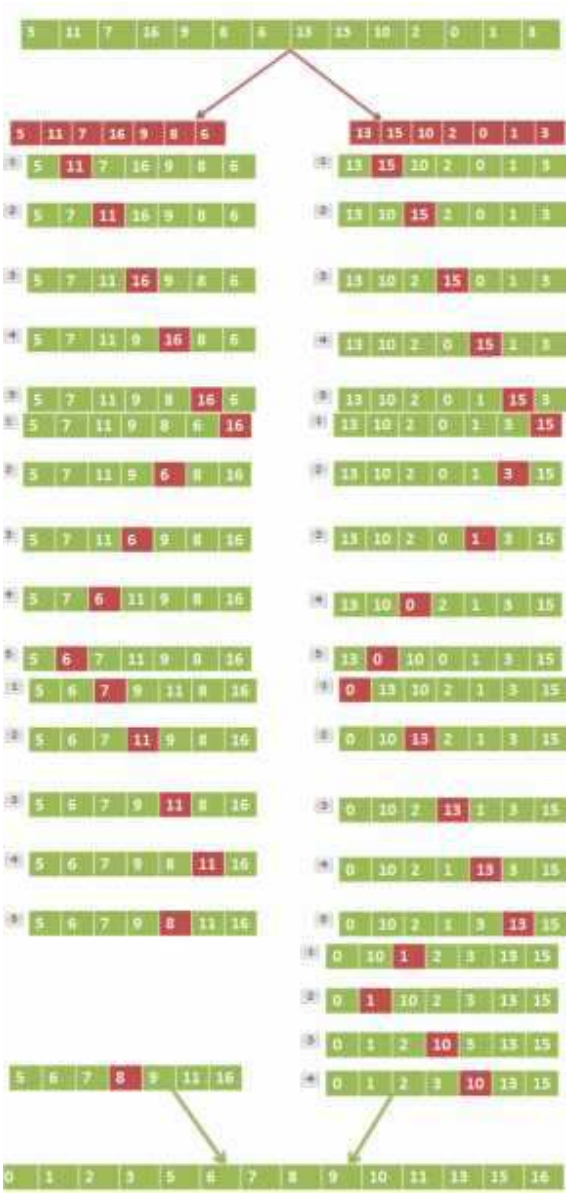

Fig 3: Working o Hybrid approach

III. COMPARISON AND RESULTS

Table 1: The array size is 1000 for each Sorting Algorithm

	Array Size	**Sort Time**
Bubble Sort	1000	0 Milliseconds
Merge Sort	1000	0 Milliseconds
Hybrid approach	1000	0 Milliseconds

After that, the array size set 5000. The result for 5000 items is shown in

Table 2: The array size is 5000 for each Sorting Algorithm.

	Array Size	**Sort Time**
Bubble Sort	5000	145 Milliseconds
Merge Sort	5000	120 Milliseconds
Hybrid approach	5000	97 Milliseconds

As a result, each Sorting Algorithm's total timing tested with different array size. The results shown in Table 3:

Table 3: Sorting Algorithm's timing with different array sizes.

Array Size	**Bubble Sort**	**Merge Sort**	**Hybrid approach**
1000	0	0	0
5000	145	120	97
10000	600	387	290
50000	14050	1985	1291
100000	56900	3870	2516
200000	229440	7740	5031
300000	513340	11610	6546

Variations in numbers (time recorded):

Consider Bubble Sort's time taken for 5000 integers, 145 milliseconds. This is an average value. Due to other processes going on at the same time as comparison, the recorded time varies during each run. It also varies from computer to computer (mine one is a decent one though, Intel i3 with 4 GB of RAM). Like in one execution, above time was recorded as 40 millisecond and in other it was 50 milisec. Same goes for all the time values. In effect I have averaged the time taken in the 4 executions. For greater accuracy keep the number of samples to a minimum 10 (although I leave it to you). And also the code should be executed on a variety of computers and Operating Systems (again I leave that for you). However the differences are so pronounced that a variation of 5-10% does not affect the end result [13].

IV. CONCLUSION

In this paper the merit and demerits of two sorting techniques are discussed. The bubble sort is better than merge sort in practice for small set of data, but as size of input data increases, the performance of bubble sort suddenly drop down. Keeping in view the previous flaws in merge and bubble sort algorithms Hybrid Approach can be used to eliminate flaws. The proposed approach will work in such a way that complexity will be reduced and minimum comparisons will be needed to sort a data. As a consequence time and space complexity will be reduced to sort data.

FUTURE WORK

In this part we focus the possible future development about summary of these proposed approach we implement this approach practically. In this approach all work that we have planned were not completed, we focus only specific problem which is solved using this approach and get successful results and during compression with other approaches it is found best approach.

References

[1] Black, Paul E. (24 August 2009). "bubble sort". Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology. Retrieved 1 October 2014.

[2] Todd, Stephen. "Algorithm and hardware for a merge sort using multiple processors." IBM Journal of Research and Development 22.5 (1978): 509-517.

[3] Alyasseri, Zaid Abdi Alkareem, Kadhim Al-Attar, and Mazin Nasser. "Parallelize Bubble and Merge Sort Algorithms Using Message Passing Interface (MPI)." arXiv preprint arXiv:1411.5283 (2014).

[4] Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition. Addison-Wesley, 1998. ISBN 0-201-89685-0. Pages 106–110 of section 5.2.2:

[5] Huang, Dijiang, et al. "MobiCloud: building secure cloud framework for mobile computing and communication." Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on. Ieee, 2010.

[6] Roy, Sudipta, et al. "International Journal of Advanced Research in Computer Science and Software Engineering." International Journal 3.6 (2013).

[7] Jyoti Mundra et al, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.9, September- 2015, pg. 173-181

[8] Sharma, S., Kumar, V., Singh, P., & Singh, A. (2016). VRF: A Novel Algorithm for optimized Sorting.

[9] Abbas, Z. A. (2016). *Comparison study of sorting techniques in dynamic data structure* (Doctoral dissertation, Universiti Tun Hussein Onn Malaysia).

[10] Abbas, Z. A. (2016). *Comparison study of sorting techniques in dynamic data structure* (Doctoral dissertation, Universiti Tun Hussein Onn Malaysia).

[11] Pathy, S., & Baboo, S. (2016). Analysis of code coverage metrics using eCobertura and EclEmma: A case study for sorting programs. *International Journal*, *4*(2).

[12] Mishra, A., & Goyal, G. (2016). An Optimized Input Sorting Algorithm. *Global Journal of Computer Science and Technology*, *16*(1).

[13] N. Sarwar, IS Bajwa & Rauf Sajjad (2016) "*Automated Generation of EXPRESS-G Models Using NLP*" Sindh University Research Journal (Science Series) Vol. 48 (1) 05-12.

[14] GÜZEL, A., & AKTAŞ, Ö (2016). A Survey on Bad Smells in Codes and Usage of Algorithm Analysis. International Journal of Computer Science and Software Engineering (IJCSSE), Volume 5, Issue 6, June 2016 ISSN (Online): 2409-4285.

[15] Min, W. (2010, July). Analysis on bubble sort algorithm optimization. *Information Technology and Applications (IFITA), 2010 International Forum on* (Vol. 1, pp. 208-211). IEEE.