Research Article

Parallelized Hybrid Merge Sorting Implementation Based On Ram Performance Using Mpich++

Leta Tesfaye Jule[1], A. Sampath Kumar[2], Krishnaraj Ramaswamy[3]

[1]Centre for Excellence in Indigenous Knowledge, Innovative Technology Transfer and Enterpreneurship and, Department of Physics, College of Natural and Computational Science, Dambi Dollo University, Ethiopia.
[2]Department of Computer Science and Engineering, Dambi Dollo University, Ethiopia..
[3]Centre for Excellence in Indigenous Knowledge, Innovative Technology Transfer and Enterpreneurship and, Department of Mechanical Engineering, College of Natural and Computational Science, Dambi Dollo University, Ethiopia.

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract— Parallel handling alludes to the idea of accelerating the execution of a program by separating program into different sections that can execute all the while, each on its own processor. Consolidation sort is a proficient gap and-vanquish arranging calculation, consequently better comprehension of union sort parallelization can add to better comprehension of gap and-overcome parallelization by and large. This paper manages the execution of equal consolidation arranging method under MPI utilizing MPICH++for correspondence between the centers and for the calculation. Since it is particularly appropriate to execute in LINUX frameworks. The expert and the slave speak with one another utilizing MPICH++. In the calculation which we have carried out is for converge on a few hubs, it very well might be for just at least one slaves. One of equal cycles is assigned as an expert and remaining goes about as slaves. Unsorted rundown of components is gotten with randomized strategy. The expert partitions the unsorted rundown of components into the information parts equivalent to the quantity of slaves. We plan to assess and contrast these insights and the time taken to tackle a similar issue in sequential execution to show correspondence overhead engaged with equal calculation. We plan to analyze and assess the measurements acquired for various sizes of RAM under equal execution in a solitary hub including just two centers, where one goes about as expert and other as slave. Further shows reliance of sequential execution on RAM for similar issue by implementing its sequential form under different sizes of RAM. The proposed system results are analyzed in two phases whereas, Serial Merge Sort with 256 MB RAM achieves 126.1472 peak range and 1000 MB RAM achieves2.0674with the time difference -124.0798sec. In the second phase parallel merge sorting with 256 MB RAM achieves 126.1472peak range and with 1000 MB RAM achieves 2.0674 with time difference -124.0798

Keywords—Parallel Execution, SMP, Cluster Computing, MPI, RAM.

1. Introduction

Parallel preparing alludes to idea of accelerating execution of a program by separating program into various pieces that can execute all the while, each on its own processor. Consolidation sort is a productive separation and-vanquish arranging calculation. Since combine sort is more obvious than other valuable separation and-vanquish strategies. One regular illustration of equal handling is the execution of the union sort inside an equal preparing climate. This paper bargains how to deal with blend sort issue that can be parted into sub-issues and each sub-issue can be addressed all the while. With PCs being organized today, it has gotten conceivable to share assets like records, printers, fax machines, scanners, email workers, and so forth One such asset that can be shared however is for the most part not, is the CPU. The present processors are profoundly cutting-edge and extremely quick, fit for a great many activities each second. On the off chance that this figuring power is utilized cooperatively to take care of more serious issues, time taken to take care of the issue can diminish radically. Anyway the entire activity of equal preparing likewise relies upon RAM accessible to processors for their calculation.

A. Existing Methods

1) MPI: In April 1994, the MPI specification 1.0 was finalized. The message passing interface is now available. It was a local effort to define a center message library's language structure and semantics which would be useful to a large number of clients and would be implemented in a broad range of MPP.

2) MPI2: All significant PC merchants upheld the MPI standard and work started on MPI-2, where new usefulness, dynamic cycle the board, uneven correspondence, agreeable I/O, C++ ties, Fortran 90 options, broadened aggregate tasks, and different other usefulness were summed to MPI-1 standard.

3) Openmp: It was the standard for fair programming in common memory. Openmp API provides program technicians with a simple way to encourage fair registration applications for common memories.

4) MPICH2: A whole new MPI execution to support both MPI-1 and MPI-2. The MPICH2 plan is clearly calculated to achieve superior and flexibility, and to facilitate experimentation. In MPICH2, aggregates are much faster than "exemplary" MPI and MPICH renditions and have low overhead communication.

B. Proposed System Frameworks

The fundamental point is to shape single hub model for MPI which shows the exhibition reliance of equal consolidation sort on RAM of hubs (work area PCs) utilized in equal registering. Single hub model comprises of a customer, an expert, equipped for taking care of solicitations from the customer, and a slave, fit for tolerating issues from the expert and sending the arrangement back.

2. Related Works

Generally, various processors were given inside an uncommonly planned "equal PC"; thusly, Linux currently upholds SMP Pentium frameworks in which different processors share a solitary memory and transport interface inside a solitary PC. It is additionally workable for a gathering of PCs to be interconnected by an organization to shape an equal handling group. In [1] proposed Cluster based equal figuring system which depends on the Master-Slave processing worldview and it copies the equal registering climate. The creator in [2] utilized the double center Window-based stage to consider the impact of equal cycles number and furthermore the quantity of centers on the exhibition of three MPI equal executions for some arranging calculations. the exhibition of [3] assessed and speedup of equal consolidation sort calculation. A versatile system towards investigating the equal consolidation sort is proposed by author[4]. In [5] introduced the functional execution correlation of equal arranging calculations on homogeneous organization of workstations. Additionally the creator of [6] carried out the common memory, message passing, and cross breed blend sorts for independent and grouped SMPs. In [7] led investigation of PMSA and assessed exhibition of equal consolidation sort calculation on inexactly coupled engineering and contrasted it and hypothetical examination. It has been discovered that there is no significant distinction between hypothetical execution examination and real outcome. We exhibit the presentation gain and misfortunes accomplished through equal preparing utilizing MPI. We likewise shows the presentation reliance of equal applications on RAM of the hubs (work area PCs) utilized in equal figuring. In [8] the creator evaluated a useful assessment based orchestrating computation for CUDA-enabled GPUs. Their examination deduced that notwithstanding the way that bitonic sorter has a by and large high estimation unpredictability, it is as yet capable when organizing little groupings. The makers offer confirmation that their computation displays up to 30% favored execution over past upgraded assessment based estimations for input progressions with countless parts. The maker in [9] enhances Paterson's variation of the AKS organizing network by tuning the invariant to be kept up. Their assessment gives an indisputable and essential segregated appearance of the extraordinary huge goodness transformation of the AKS result. The work in [10] proposed a direct masterminding designing whose guideline feature is the pipelined usage of an orchestrating association of fixed I/O size p to sort a discretionarily gigantic educational record of N segments. Their assessment contemplated that the time execution of their arrangement is for all intents and purposes self-governing of the cost and significance of the crucial orchestrating association. In Hybrid Cuckoo, Modified Honey Bee [11], Parallel Lion Optimizations [12] are Search algorithm has beenused to found the better accuracy by eliminating the local minimum optimal minimum problem. In [13] proposed a proficient execution of the MMP calculation, and a quick calculation for geodesic distance estimation. In [14] improved the presentation of the CH calculation by sifting wavefront windows and keeping a need line. In [15] proposed an equal CH calculation that engenders countless wavefront windows simultaneously. Additionally, In [16] sped up the wavefront proliferation for both the MMP calculation and the CH calculation by synchronous engendering of various windows. In. [17] proposed a triangle-situated wavefront engendering calculation with a window pruning methodology to improve computational speed. Another class of calculations, called chart based techniques, first pre-figures a meager diagram that encodes the geodesic data of the surface. Geodesic distance inquiry is then performed by figuring a most limited way on the chart. In [18] proposed the discrete geodesic diagram (DGG) to inexact geodesic distance with given precision and exactly straight computational time. In [19] precomputed a vicinity chart between a bunch of test focuses on a superficial level, and increased the diagram with the source and target vertices to figure the most limited way and rough the geodesic distance [20]. For chart based techniques, albeit the diagram development should conceivably be possible in equal, the distance inquiry is inherently consecutive

3. System Requirement

A. Hardware Requirements

- Two RAM: 256MB and 1GB
- Processor: Pentium 4 (3 G Hz)
- Hard Disk Space:5 GB

B. Software Requirements

- OS: Linux
- Compiler: GCC
- Version: Fedora Core 14
- Communication: MPI

4. System Design

Framework is to be planned to such an extent that it exhibits presentation reliance of equal and sequential execution on RAM and furthermore it shows accompanying:

• How a customer can present whole issue to an expert and gathers arrangement back from it without fretting over how it has been tackled.

• How expert distinguishes accessible slaves on organization, and how it recognizes framework load on machine to decide if it merits sending an assignment to that specific customer.

• How an issue can be submitted to slaves.

• How arrangements of given issue can be recovered from slave.

• How slaves take care of given issue.

Plan was made secluded for example product is coherently parceled into segments that perform explicit capacities and sub-capacities. Expert is planned to such an extent that it has usefulness to oversee association and correspondence with slave, It examines and recognizes every one of centers or slaves accessible on hub here it is just one slave to be distinguished. It at that point appoints processor positions to distinguish centers. Expert doles out issue to slave. It likewise needs to acknowledge outcomes sent back by slave after they finish calculation of sub-errands relegated to them. At that point got result must be collected organized appropriately to acquire answer for fundamental issue. Slave is intended to have usefulness to peruse issue (in event of single slave)/sub-issue sent by expert, assess issue (if there should arise an occurrence of single slave)/sub-issue and send outcome back to expert. Fundamental issue is taken by expert center and allocates undertaking into slave center. Slave center send back arrangements of doled out errand or issue.

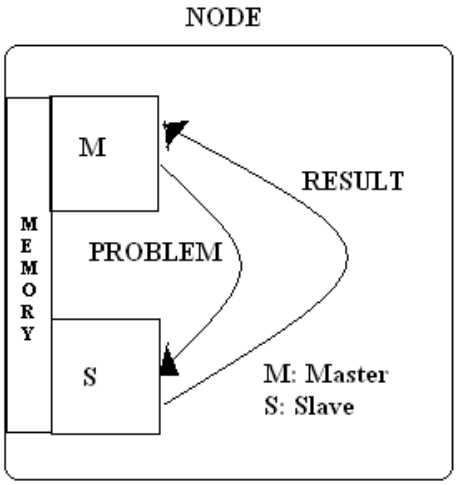

Figure 1. Architecture for presenting Parallel Computing Framework.

C. Configuration of MPI

Download the following commands to MPICH++package and sort.
Unload tar record and go to high level index:
tar xzf mpich2-1.3.2.tar.gz
cd mpich2-1.3.2
Design MPICH2 determining establishment index:
./model - prefix=/home/<USERNAME>/mpich2-introduce |& tee c.txt
Assemble MPICH2:make 2>&1 | tee m.txt
Introduce MPICH2 commands: Introduce 2>&1 | tee mi.txt
Add receptacle establishment index to your way in your startup script
PATH= home <USERNAME>/mpich2-introduce/bin:$PATH; send out Way

5. Implementation

Execution is most urgent stage in accomplishing a fruitful equal framework. Issue to be addressed must be parallelized so calculation time is diminished. System comprises of a customer, an expert center, fit for dealing with demands from customer, and slave, equipped for tolerating issues from expert and sending arrangement back. Expert and the slave speak with one another utilizing MPICH-2. Issue must be split to such an extent that the correspondence between the worker and the customer is least. The absolute calculation time to take care of the issue totally is affected by correspondence time between hubs.

A. Parallel Merge Sorting Architecture

Expert can screen advance and figure and report time taken to take care of issue, taking into account time spent in allotting issue into slave as well as sending outcomes alongside correspondence delays. Zero cushioning for equivalent burden adjusting is made .At that point slaves utilize the consecutive form of union sort to sort their own information. Arranged sub-records are shipped off expert. At long last, expert unions every one of arranged sub-records into one arranged rundown. Fig.2 shows the progression of tasks engaged with equal union arranging. The diagram for the execution of consolidation sort is appeared in Fig.3. Henceforth, we need to execute equal frameworks comprising of set of free work area laptops interconnected by quick LAN helpfully cooperating as a solitary coordinated figuring asset in order to give higher accessibility, dependability and adaptability. Yet, to show the exhibition reliance on Slam we are thinking about just single hub with two centers, one go about as expert and other as slave. There is no division of issue, rather whole unsorted rundown is given to single accessible slave.

6. Results and Analysis

We have broke down the presentation of equal technique against customary sequential strategy. The outcomes are arranged and analyzed. We determined the ideal opportunity for arranging unordered rundown utilizing both sequential and equal consolidation sort calculation. From Table I we can presume that Presentation of sequential execution nearly stays same even after increment in Smash size. There are irrelevant calculation time varieties for expansion in Smash size. This is on grounds that Sequential execution is performed by actual centers with irrelevant Slam utilization and furthermore because of no correspondence required between centers. Thus it is free of Slam. From Table II we can presume that Presentation of equal execution radically increments when there is expansion in Slam size. It shows intense reduction in calculation time with expansion in Slam. Since equal execution frequently utilizes Smash for correspondence among centers and furthermore it includes parcel of send and get activities and briefly putting away aftereffect of issue allocated to centers. Higher size of unordered rundown time distinction in Table II is high, on grounds that higher info size more will be sends and gets bringing about need of higher use of Smash. So for more modest Smash calculation time will be more and bigger Slam size, calculation time will be less in equal implementation at long last bringing about better execution. Fig.4 is diagram showing exhibition reliance of equal union arranging on Slam.

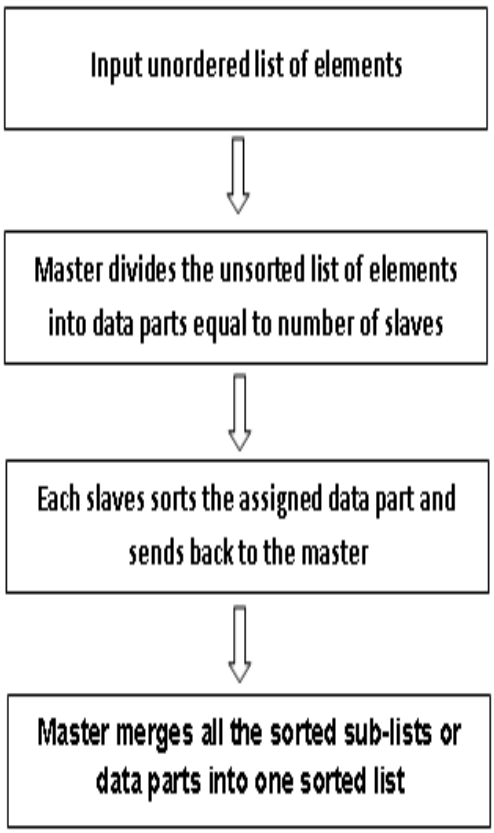

Figure 2. Merge Sorting Architecture

```
/* Merge Sort */
1. Merge_sort(sub-list[], start, last)
2. {  Allot spaces for "sublist1" and "sub-list2" of size
      each (last-start)/2;
3.    mid = (first+last)/2;
4.    lcount = mid - first + 1;
5.    ucount = last - mid;
6.    if (last == first) { return};
                              7.    else {
8.            sub-list1=merge_sort(sub-list[], first, mid);
9.            sublist2=merge_sort(sub-list[], mid+1, last);
10.           merge(sublist1, lcount, sublist2,ucount);
11.       }
12. }
```

Figure 3. Merge sort implementation outline

In Table 1 and 2, results of a sequential execution with 1000MB Smash and equivalent execution with 1.000MB Slam are seen to be faster than a sequential execution for the most part, but the sequential execution is faster than equal performance at a solitary hub with 2 centers with various unordered rundown sizes. This can be resolved by expanding hub, but is actually not within our reach, and as function in future. This can be overcome by overall coordination involving an equal implementation. time to bind to a slave, time, time of connection.

Table 1. Comparison of Serial Merge Sort

Unordered list size (number of elements)	1000000	2000000	3000000	4000000	5000000
Serial execution with 256 MB RAM	0.2274	0.5442	0.9829	1.2241	1.5664
Serial execution with 1000 MB RAM	0.2247	0.5534	0.9799	1.2173	1.5851
Time difference	-0.0037	+0.0082	-0.0040	-0.0078	+0.0197

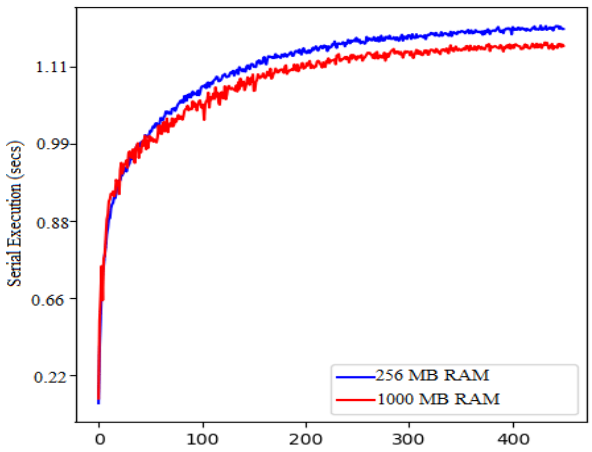

Figure 4 Serial Merge Sort on Different Ram Size

Table 2. Performance Dependency of Parallel Merge Sort

Size of unordered list (number of elements)	1000000	2000000	3000000	4000000	5000000
Serial execution with 256 MB RAM	4.7574	12.1161	21.1655	40.7849	126.1472
Serial execution with 1000 MB RAM	0.4265	0.8972	1.2281	1.6348	2.0674
Time difference	-4.3309	-11.2189	-19.9374	-39.1501	-124.0798

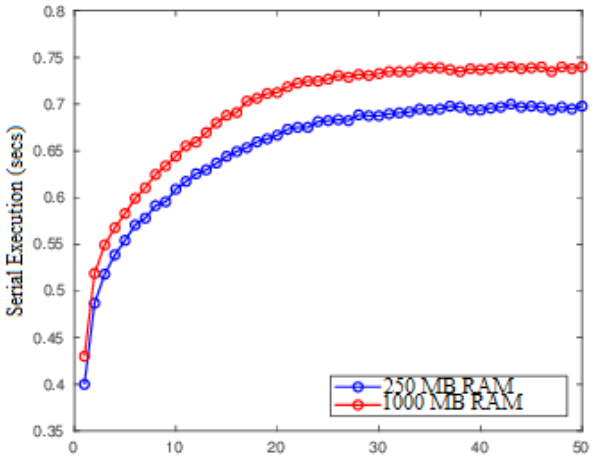

Figure 5. Comparison of Serial Execution in secs

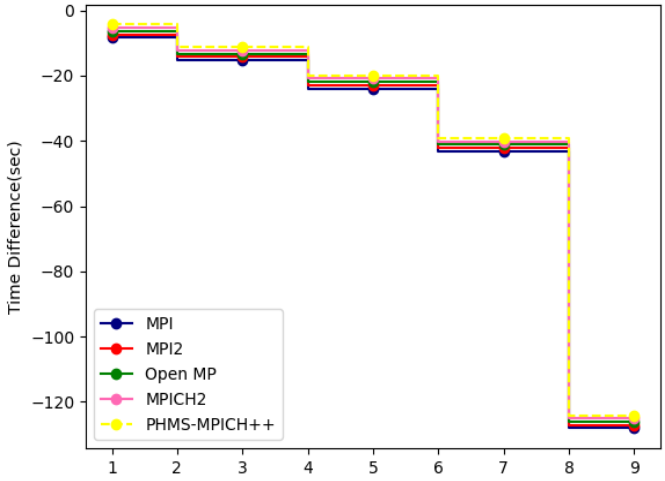

Figure 6. Comparison of Time difference

When the query is sent along with slave time inputs to find the customer's solutions, it takes the time to assimilate the received data. The results of the sorter sample list of 1000000 elements as seen in Fig.5.

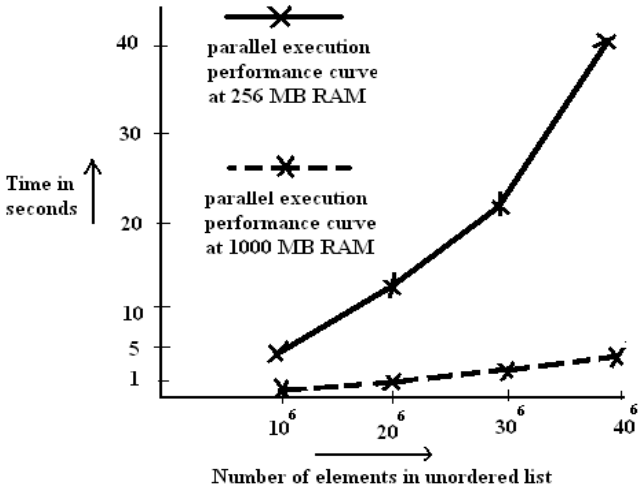

Figure7. Performance parallel merge sorting

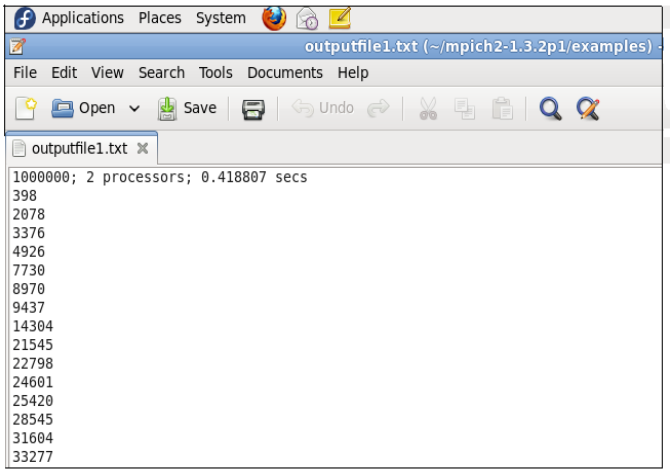

Figure 8. Sorting unsorted list

7. Conclusion

We introduced a model that shows presentation gain and misfortunes accomplished through equal handling. Consolidation arranging is executed sequentially and furthermore in equal. We introduced a system utilizing MPI that showed assessment of exhibition reliance of equal applications on Smash. We additionally showed reliance of sequential implementation on Smash for similar issue by implementing its sequential form under various sizes of Slam. The proposed system results are analyzed in two phases whereas, Serial Merge Sort with 256 MB RAM achieves 126.1472 peak range and 1000 MB RAM achieves2.0674with the time difference -124.0798sec. In the second phase parallel merge sorting with 256 MB RAM achieves 126.1472peak range and with 1000 MB RAM achieves 2.0674 with time difference -124.0798

8. Future Works

We contrasted outcomes and runs on single hub as it were. It very well may be stretched out to more number of hubs to assess exhibition reliance on Slam with expansion in number of hubs utilizing same equal consolidation sort calculation. Despite fact that strategy that has been utilized here can be sent to take care of bigger measured issues, it is lumbering to give data contribution for bigger size. Thus this work can be stretched out to give contribution from records for bigger measured unordered rundown. It can likewise be stretched out to take care of other comparative issues like discovering determinant, n sovereigns issue, and other backtracking issues. The examination is additionally helpful for making a legitimate proposal to choose best calculation identified with a specific equal application. In event that hubs are broadened, hub disappointment can be a difficult that must be handled.

References

[1] Marszałek, Zbigniew. "Parallelization of modified merge sort algorithm." *Symmetry* 9.9 (2017): 176.

[2] Singh, Dhirendra Pratap, Ishan Joshi, and Jaytrilok Choudhary. "Survey of GPU based sorting algorithms." *International Journal of Parallel Programming* 46.6 (2018): 1017-1034.

[3] Karpiński, Michał, and Marek Piotrów. "Encoding cardinality constraints using multiway merge selection networks." *Constraints* 24.3 (2019): 234-251.

[4] Blelloch, Guy E., et al. "The parallel persistent memory model." *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. 2018.

[5] Najafi, M. Hassan, et al. "Low-cost sorting network circuits using unary processing." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.8 (2018): 1471-1480.

[6] Abdel-Hafeez, Saleh, and Ann Gordon-Ross. "An Efficient O ($ N $) Comparison-Free Sorting Algorithm." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.6 (2017): 1930-1942.

[7] Abdulazeeza, Adnan Mohsin, Lana Latif Nahmatwllab, and Diyar Qader. "Pipelined Parallel Processing Implementation based on Distributed Memory Systems." *International Journal of Innovation* 13.7 (2020): 12.

[8] Putri, Revina Awalia, et al. "Implementation of Resilience as a Service for Parallel Computing." *2020 International Electronics Symposium (IES)*. IEEE, 2020.

[9] Samardzic, Nikola, et al. "Bonsai: high-performance adaptive merge tree sorting." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

[10] A. Sampathkumar, Mulerikkal, J. & Sivaram, M. Glowworm swarm optimization for effectual load balancing and routing strategies in wireless sensor networks. Wireless Networks, vol. 26, no. 6, 4227–4238 (2020). https://doi.org/10.1007/s11276-020-02336-w

[11] A. Sampathkumar, Rastogi, R., Arukonda, S. Achyut Shankar, Sandeep Kautish & M. Sivaram "An efficient hybrid methodology for detection of cancer-causing gene using CSC for micro array data" Journal of Ambient Intelligence and Human Comput (2020),Springer. https://doi.org/10.1007/s12652-020-01731-7.

[12] A. Sampathkumar, Vivekanandan. P "Gene Selection Using PLOA Method In Microarray Data For Cancer Classification" Journal of Medical Imaging and Health Informatics (2019). 9, 1294-1300.

[13] Sampathkumar. A, Vivekanandan. P "Gene Selection Using Multiple Queen Colonies In Large Scale Machine Learning" Journal of Electrical Engineering (2018). 9 (6), 97-111.

[14] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," ACM Trans. Graph., vol. 24, no. 3, pp. 553–560, 2005.

[15] S.-Q. Xin and G.-J. Wang, "Improving Chen and Han's algorithm on the discrete geodesic problem," ACM Trans. Graph., vol. 28, no. 4, pp. 104:1–104:8, 2009.

[16] X. Ying, S.-Q. Xin, and Y. He, "Parallel Chen-Han (PCH) algorithm for discrete geodesics," ACM Transactions on Graphics, vol. 33, no. 1, pp. 9:1–9:11, 2014.

[17] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L0 gradient minimization," ACM Trans. Graph., vol. 30, no. 6, pp. 174:1–174:12, 2011.

[18] Y. Qin, X. Han, H. Yu, Y. Yu, and J. Zhang, "Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation," ACM Trans. Graph., vol. 35, no. 4, pp. 125:1–125:13, 2016

[19] X. Wang, Z. Fang, J. Wu, S.-Q. Xin, and Y. He, "Discrete geodesic graph (DGG) for computing geodesic distances on polyhedral surfaces," Computer-Aided Geometric Design, vol. 52, pp. 262–284, 2017.

[20] S. Xin, W. Wang, Y. He, Y. Zhou, S. Chen, C. Tu, and Z. Shu, "Lightweight preprocessing and fast query of geodesic distance via proximity graph," Computer-Aided Design, vol. 102, pp. 128–138, 2018