ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

Sorting in Numerical Dataset to Minimize The Complexity in Multi-Core Processors

J.Dheebika[1], Dr.T.Meyyapan[2]

[1]*M.Phil Research Scholar,* [2]*Professor*
Department of Computer Science and Engineering, Alagappa University,
Karaikudi-600 003, Tamilnadu, India.
[1]dheebika94@gmail.com,[2]meyyappant@alagappauniversity.ac.in

Abstract— **Sorting problem is a common problem in parallel computing environment. The performance of parallel sorting algorithm based on the data size and its values. Day by day the computing cost is increase gradually due to utilization of advance computing architecture. Middle level organizations are facing difficulties to build the advance multi core architecture. Most of the Researches contribute the algorithms for improving the sorting technique in parallel computing architecture. Today's single Parallel sorting algorithm is applied over various large numerical dataset and compare with other algorithms. Even the researchers contribute novel or improved sorting techniques, time complexity increases due to size and values of dataset. The ideas of our contribution, multi parallel sorting algorithms are applied over in single large dataset in multi core environment and analysis the performance with single parallel sorting algorithm which is implementing in single dataset. The analysis result shows the better performance compare with existing parallel sorting algorithm and utilization of hardware resources improves the optimal.**

Keywords— **Sorting algorithm, Parallel processing, Multi core architecture.**

I. INTRODUCTION

Sorting algorithms are widely used in numerical problems. Various types of sorting algorithms implements in to different architectures for gaining efficiency and optimize the utilization of system resources.. They are having distinct properties for various architecture models. Nowadays, data collection and transformation handles in Giga bytes level of size. In addition, the system architecture reach the new dimension of nano electronic system. New algorithms or modify the existing methods requires to improve the efficiency based on the utilization of advanced hardware architecture. Currently, multi core architecture is one of the advance hardware architecture. Parallel programming concept is very suitable to engage the multi core architecture. Various types of parallel sorting algorithm accustom in to parallel computing environment. The well known sorting algorithms with parallel structure are parallel quick sort, parallel merge sort, odd sort, bitonic sort. Even parallel sorting algorithms adapts on multi core architecture. The level of efficiency decides from on characteristics of numerical data. Some algorithms achieve the good performance and others fail on same hardware environment. The reason of this conflict arises based on data. Hence, sometimes, optimal utilization of hardware resources is unsuccessful. In this paper, Implements the parallel sorting algorithms simultaneously in each core in parallel not a single parallel sorting algorithm for all core. Measure the time complexity of proposed method and existing method. The time complexity metric value shows the utilization of core architecture optimized compare with other methods.

II. SORTING ALGORITHMS

Data-driven algorithms especially use sorting to gain an efficient access to data. Many sorting algorithms with distinct properties for different architectures have been developed. It should be noted that some sorting algorithms for sequential hardware platforms should be modified when implemented on parallel architectures. Some algorithms have a parallel structure, making them easier to adapt for parallel hardware architecture. There are a few well-known hybrid algorithms on GPU cards and multi-core systems. Peters [3] implemented an adaptive bitonic sorting algorithm with a bitonic tree based on a tables structure. In order to increase the speed of sort, multiple steps of bitonic merge have to be executed with a single kernel invocation. This can be achieved with multistep bitonic sort [4]. Interval Based Rearrangement (IBR) bitonic sort [5] is based on adaptive bitonic sort implemented by Bilardi *et al.* [9]. Cederman [1] adapted a quick sort for the GPU platform. Bitonic sort is a sorting-network algorithm developed by Batcher [6]. A sorting network is a sorting algorithm where the sequence of comparisons is predetermined and data-independent. The Bitonic algorithm sorts a sequence of N elements and consists of logN stages. Multiple steps of bitonic merge have to be executed with a single kernel invocation. This can be achieved with multistep

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

bitonic sort [4]. Adaptive bitonic sort operates on the idea, that every bitonic sequence can be merged, if first half of a bitonic sequence and second half of a bitonic sequence are ring-shifted by a value called q. Value q can be found with a variation of a binary search. Merge sort is based on the divide-and-conquer approach. It follows this approach by splitting the sequence into multiple subsequences, sorting them and then merging them into sorted sequence. If the algorithm for merging sorted sequences is stable, than the whole merge sort algorithm is also stable [10]. Sathish [2] presented the parallel merge sort with variation. quicksort is also based on the divide-and-conquer basis. In divide step, the pivot has to be determined. This pivot is used to partition the input sequence A[p...r] into 2 subsequences. When the partitioning is done, pivot has to be placed in A[q], where all the elements in subsequence A[p...q − 1] have to be lower or equal to A[q] and all the elements of A[q + 1...r] have to be greater than A[q] [11]. For a good performance an effective sorting algorithm has to be used, which is usually counting sort [2,8,10]. Sample sort is a sorting algorithm, which splits the input sequence into multiple smaller buckets, until they are small enough to be sorted by alternative sort. For the parallel implementation we chose variation of the sample sort by Dehne *et al.* [6], because it is more robust for sorting different types of input sequences than Leischner's *et al.* [7, 8]. Radix sort is one of the fastest sorting algorithms for short keys and is the only sorting algorithm in this report which is not comparison based. Its sequential variation first splits the elements being sorted (numbers, words, dates, ...) into d r-bit digits.

III. Proposed Work

Advance processors consist of more than two central processing unit. This architecture can be executed multiple programs at the same time. Multiple CPU integrated into single hardware architecture. Processors companies have designed several multi-core processors for sharing memory, inter-core communication and message passing. Mainly parallel programs are executed in multi-core processor environment. Sorting problem is one of the common in numerical computation. Sorting algorithm execute the large count of number in parallel that is called parallel sorting algorithm. The design of parallel sorting algorithm adapts on the parallel computing architecture. Existing processing models execute the sorting algorithm in parallel of all core sequentially. The diagram shows the implementation of parallel sorting algorithm in each core sequentially.

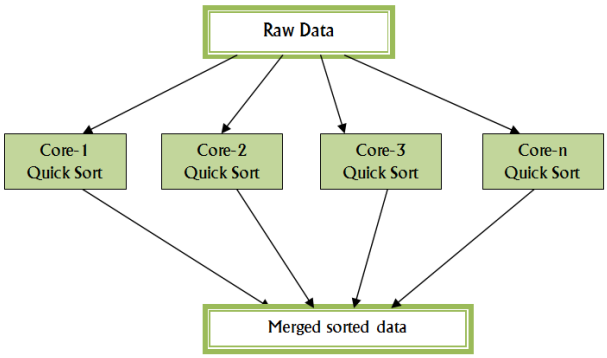

Fig. 1 Parallel quick sort algorithm

A. *Multi parallel sorting algorithm*

Quick sort is one of the types of sorting algorithm and its efficiency is good that compares with other sorting algorithm. Mostly single parallel sorting algorithm executes in parallel sequentially on multi core processor. The computational complexity may be vary in different sorting algorithm or different core architecture. The diagram shows the implementation of multi parallel sorting algorithm in multi-core architecture.

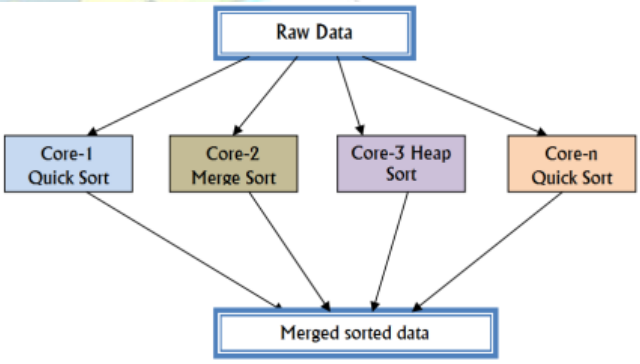

Fig. 2 Multi Parallel Sorting algorithm

The obtained computational complexity of the algorithm is the equal sequential complexity divided by the number of parallel multiprocessors – n/p _ $log2n$.In this work, there are three types of sorting algorithm participate into multi core architecture. Initially, large amount of numerical data split into numbers of piece. The number of pieces is equal to number of core processors engaged in parallel computing. Each piece allocates into each core. Three different sorting algorithms such as quick sort, merger sort, and heap sort participate to sorting the piece of numerical data in each core. The multi sorting algorithm computational complexity is (average computational complexity of three sorting algorithm – n _ log (n), memory complexity – 2 _ n).

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

IV. Experimental Results

The proposed method implements in Matlab-8.1. The parallel computing tool box helps to design the multi-core architecture. There are 3 core is used for testing process and three different algorithm applies into the multi core architecture. The thyroid dataset has taken from UCI machine learning repository. The TSH attribute values split into three parts. First part of the data applies into core-1, second part of the data connects with core-2 and third part of the data works with core-3. Quick sort applies in to core-1, Mergesort executes with core-2 and heapsort implements into core-3. The time complexity is calculated and result shows in the table 1.

TABLE I
Time Complexity Results

Core -1 Quick Sort (6000 Recs)	Core-2 Merge Sort(8000 Recs)	Core -3 Heap sort (10000 Recs)
0.37 ms	0.41 ms	0.53 ms

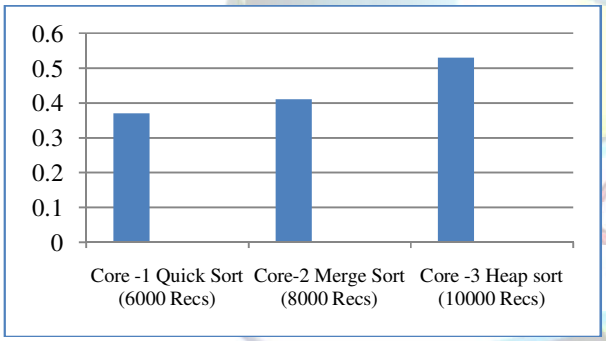

Fig. 3 Graphical Representation of Complexity results.

In fig3, It is examined that the proposed method, the time complexity of core 1 process is reduced than others. The time period of execution is increased while apply the heap sort in 10000 records.

V. Conclusions

Previous works on parallel algorithms, single parallel sorting algorithm applied into each core of the multi-core architecture. Sometimes, the time complexity increases based on the numerical dataset. In this paper, multi parallel sorting algorithms implement in each core of the multi-core architecture. This work encourages increase the efficiency of computing performance and reduces the usage of hardware resources. In future, this hybrid sorting algorithms will be apply in multi-core architecture and allocate the core processor dynamically that is based on the characteristics of dataset.

References

[1] Cederman D, Tsigas P. GPU-quicksort: A practical quicksort algorithm for graphics processors. J. Exp. Algorithmics Jan 2010; 14:4:1.4–4:1.24,.

[2] Satish N, Harris M, Garland M. "Designing efficient sorting algorithms for manycore GPUs". Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, IPDPS '09, IEEE Computer Society: Washington, DC, USA, 2009; 1–10,

[3] Batcher KE. Sorting networks and their applications. Proceedings of the April 30–May 1968, Spring Joint Computer Conference, AFIPS '68 (Spring), ACM: New York, NY, USA,

[4] Peters H, Schulz-Hildebrandt O, Luttenberger N. Fast in-place, comparison-based sorting with CUDA: A study with bitonic sort. Concurr. Comput. : Pract. Exper. May 2011; 23(7):681–693,

[5] Peters H, Schulz-Hildebrandt O, Luttenberger N. A novel sorting algorithm for many-core architectures based on adaptive bitonc sort. 26th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2012, Shanghai, China, May 21-25, 2012, 2012;

[6] Dehne F, Zaboli H. Deterministic sample sort for GPUs. CoRR 2010; abs/1002.4464. URL http://dblp. uni-trier.de/db/journals/corr/corr1002.html #abs- 1002- 4464.

[7] Leischner N, Osipov V, Sanders P. GPU sample sort. 24th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, Georgia, USA, 19-23 April 2010 - Conference Proceedings, 2010; 1–10

[8] Cudpp: CUDA data parallel primitives library. https://github.com/cudpp/cudpp/, 2015

[9] Bilardi G, Nicolau A. Adaptive bitonic sorting: An optimal parallel algorithm for shared memory machines. Technical Report, Ithaca, NY, USA 1986.

[10] Cormen T H, Leiserson C E, Rivest R L, Stein C. Introduction to Algorithms, Third Edition. 3rd edn., The MIT Press, 2009.

[11] Hoare C A R. Quicksort. The Computer Journal Jan 1962; 5(1):10–16.