IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 3, Issue 1 (July-Aug. 2012), PP 20-23
www.iosrjournals.org

Modified Pure Radix Sort for Large Heterogeneous Data Set

A. Avinash Shukla[1], B. Anil Kishore Saxena [2]

Abstract: *We have proposed a Modified Pure Radix Sort for Large Heterogeneous Data Set. In this research paper we discuss the problems of radix sort, brief study of previous works of radix sort & present new modified pure radix sort algorithm for large heterogeneous data set. We try to optimize all related problems of radix sort through this algorithm. This algorithm works on the Technology of Distributed Computing which is implemented on the principal of divide & conquer method.*

I. Introduction

Sorting is a computational building block of fundamental importance and is the most widely studied algorithmic problem. The importance of sorting has led to the design of efficient sorting algorithms for a variety of architectures. Many applications rely on the availability of efficient sorting routines as a basis for their own efficiency, while some algorithms can be conveniently phrased in terms of sorting. Radix sort is an algorithm that sorts numbers by processing individual digits. n numbers consisting of k digits each are sorted in O $(n \cdot k)$ time. Radix sort can either process digits of each number starting from the least significant digit (LSD) or the most significant digit (MSD). The LSD algorithm first sorts the list by the least significant digit while preserving their relative order using a stable sort. Then it sorts them by the next digit, and so on from the least significant to the most significant, ending up with a sorted list. While the LSD radix sort requires the use of a stable sort, the MSD radix sort algorithm does not (unless stable sorting is desired). MSD radix sort is not stable. It is common for the counting sort algorithm to be used internally by the radix sort; Hybrid sorting approach, such as using insertion sort for small bins improves performance of radix sort significantly.

II. Review Of Related Literature

Rajeev Raman [1] illustrated the importance of reducing misses in the standard implementation of least-significant bit first in (LSB) radix sort, these techniques simultaneously reduce cache and TLB misses for LSB radix sort, all the techniques proposed yield algorithms whose implementations of LSB Radix sort & comparison- based sorting algorithms. **Danial** [2] explained the Communication and Cache Conscious Radix sort Algorithm (C3-Radix sort). C3-Radix sort uses the distributed shared memory parallel programming Models. Exploiting the memory hierarchy locality and reduce the amount of communication for distributed Memory computers. C3-Radix sort implements & analyses on the SGI Origin 2000 NUMA Multiprocessor & provides results for up to 16 processors and 64M 32bit keys. The results show that for small data sets compared to the number of processors, the MPI implementation is the faster while for large data sets, the shared memory implementation is faster. **Shin-Jae Lee** [3] solved the load imbalance problem present in parallel radix sort. Redistributing the keys in each round of radix, each processor has exactly the same number of keys, thereby reducing the overall sorting time. Load balanced radix sort is currently the fastest internal sorting method for distributed-memory based multiprocessors. However, as the computation time is balanced, the communication time becomes the bottleneck of the overall sorting performance. The proposed algorithm preprocesses the key by redistribution to eliminate the communication time. Once the keys are localized to each processor, the sorting is confined within processor, eliminating the need for global redistribution of keys & enables well balanced communication and computation across processors. Experimental results with various key distributions indicate significant improvements over balanced radix sort. **Jimenez- Gonzalez** [4] introduced a new algorithm called Sequential Counting Split Radix sort (SCS-Radix sort). The three important features of the SCS-Radix are the dynamic detection of data skew, the exploitation of the memory hierarchy and the execution time stability when sorting data sets with different characteristics. They claim the algorithm to be 1:2 to 45 times faster compare to Radix sort or quick sort. **Navarro & Josep** [5] focused on the improvement of data locality. CC-Radix improved the data locality by dynamically partitioning the data set into subsets that fit in cache level L2. Once in that cache level, each subset is sorted with Radix sort. The proposed algorithm is about 2 and1:4 times faster than Quick sort and Explicit Block Transfer Radix sort. **Nadathur Satish** [6] proposed the high-performance parallel radix sort and merge sort routines for many-core GPUs, taking advantage of the full programmability offered by CUDA. Radix sort is the fastest GPU sort and merge sort is the fastest comparison-based sort reported in the literature. For optimal performance, the algorithm exploited the substantial fine-grained parallelism and decomposes the computation into independent tasks. Exploiting the high-speed on chip shared memory provided by NVIDIA's GPU architecture and efficient data-parallel primitives, particularly parallel scan, the algorithms targeted the GPUs. **N. Ramprasad and Pallav Kumar Baruah** [7] suggested an optimization for the parallel radix sort algorithm, reducing the time complexity of the algorithm and ensuring balanced load on all processor. [16] Implemented it on the "Cell processor", the first implementation of the Cell Broadband Engine Architecture (CBEA). It is a heterogeneous multi-core processor system. 102400000 elements were sorted in 0.49 seconds at a rate of 207 Million/sec. **Shibdas Bandyopadhyay and Sartaj Sahni [8]** developed a new radix sort algorithm, GRS, for GPUs that reads and writes records from/to global memory only once. The existing SDK radix sort algorithm does this twice. Experiments indicate that GRS is 21% faster than SDK sort while sorting 100M numbers and is faster by between 34% and 55% when sorting 40M records with 1 to 9 32-bit fields. **Daniel Jiménez-González, Juan J. Navarro, Josep-L. Larrba-Pey [9]** proposed Parallel in-memory 64-bit sorting, an important problem in Database Management Systems and other applications such as Internet Search Engines and Data Mining Tools. [9] The algorithm is termed Parallel Counting Split Radix sort (PCS-Radix sort). The parallel stages of the algorithm increases the data locality, balance the load between processors caused by data skew and reduces significantly the amount of data communicated. The local stages of PCS-Radix sort are performed only on the bits of the key that have not been sorted during the parallel stages of the algorithm. PCS-Radix sort adapts to any parallel computer by changing three simple algorithmic parameters. [9] Implemented the algorithm on a Cray T3E-900 and the results shows that it is more than 2 times faster than the previous fastest 64-bit parallel sorting algorithm. PCS-Radix sort achieves a speed up of more than 23 in 32 processors in relation to the fastest sequential algorithm at our hands. **Daniel Cederman and Philippas Tsigas [10]** Showed at GPU-Quick sort, an efficient Quick sort algorithm suitable for the highly parallel multi-core graphics processors. Quick sort had previously been considered an inefficient sorting solution for graphics processors, but GPU-Quick sort often performs better than the fastest known sorting implementations for graphics processors, such as radix and bitonic sort. Quick sort can thus be seen as a viable alternative for sorting large quantities of data on graphics processors.

III. Proposed Modified Radix Sort

It is observed that no single method is optimal to all available data sets with varying complexity of size, number of fields, length etc. Thus an attempt is made to select a set of data set & optimize the implementation by modifying the basic algorithm. Above mentioned problems of Sorting algorithm are optimized by proposed algorithm. The algorithm is dependent on the distributed Computing Environment. Its implementation is proposed on many core machines. Given heterogeneous list is divided into two main process one is numeric and other is string. These two process work simultaneously. Suppose p1, p2 are the two main process. Each process has a unique processor. Process p1 is further distributed in different sub list according to equal length of elements in a list. These lists are sorted simultaneously on the logic of even & odd logic. Passes are transferred alternatively on the digits. After sorting these lists, combine all this & again sort this combined list. In the case of p2, make a pattern. Using the unique pattern, get the selected strings. Among these strings, same string provides same numeric values. Now proposed algorithm applies on these numeric values for sorting the given strings

IV. Algorithm For "Mrs Sort "

1. Import large heterogeneous database which is in the form of any of these format (excel sheet, oracle, Ms-Access, SQL Server etc.)
2. Make 2 clusters (Numeric & String) of similar data set on given heterogeneous database.
3. These clusters are stored in separate list. Each list has unique processor viz; p1 & p2 respectively.
4. Process p1.

ii) Separates all elements in Different sub lists according to their equality of length.

iii) These sub lists are sorted separately through new proposed algorithm. In this proposed algorithm (MRS sort) List are sorted by bypassing a digit. For Example, after completing one digit cycle in unit place, next Cycle is hundredth place digit {bypassing the tenth place digit} after completing all the given no. of digits, list is sorted. Same process is employed for sorting all the lists.

iv) Conquer (Merge) these entire sub Lists in a single combined list & sort Again as explained in step 4(iii)

5. Process p2.

i) According to unique pattern we try to search the relevant data in a list.

ii) After getting relevant data providing the unique numbers to these data if Data are same then it gets same number.

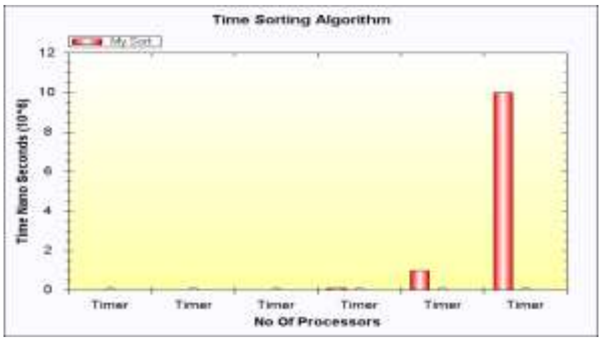
iii) Now apply the modified radix sort step 4(iii) to solve these problems.

iv) After this it will be sorted.

6. End of Algorithm.

i.) Finds the length of all elements present in this list.

V. Results And Discussions



Above graph clearly shows the result of proposed algorithm. Here 6 sub lists are used each have its own elements (10, 1000, 10000, 100000, 1000000, 10000000). These sub-lists are sorted separately, 6 processors are considered with the help of pure modified radix sort. After that all these sub-lists are merged & sorted through proposed algorithm & generate the common graph which is shown as Time v/s Number. The result is arrived at using multi- core machine. Now proposed MRS algorithm is runs on two different machines & the results were observed. Results have shown clearly that MRS Sort is best sort for heterogeneous data set on both the machines always. After MRS Sort GPU Quick Sort is the best option. Both Sorting techniques are complete by themselves, but there are slight differences between these two sorting methods as given below. This algorithm runs on two different machines, the results are as follow in the form of graph.

First this algorithm runs on Intel Pentium P6200,Intel HD Graphics,2GB DDR3 RAM,500 GB HDD Operating system :- Windows 7. The results (Graph Representation) of this machine are as follow.

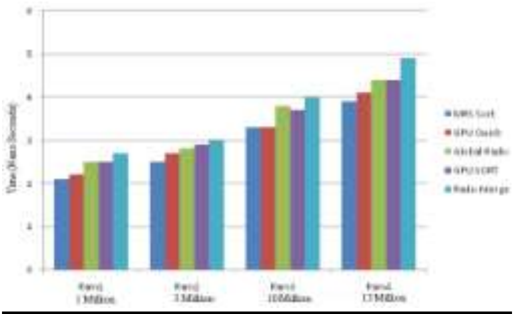
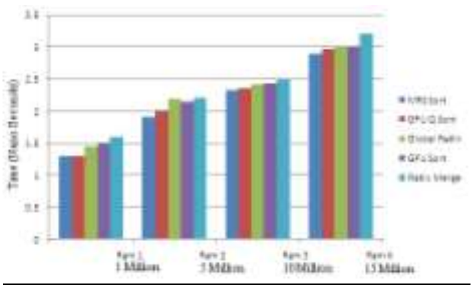


FIG 1.1

Here four groups are present Ram1, Ram2, Ram3, and Ram4. Each group has separate heterogeneous data set. Ram1 represent 1 million heterogeneous data sets. Other groups have 5 millions, 10 millions, 15 millions & 20 millions, heterogeneous data set respectively. All these groups are shown on X- axis on the graph & y-axis shows time taken (Nano Seconds) for each group.

1. Second time this algorithm runs on Intel Xeon Server Board, Intel HD Graphics, 5GB DDR3 RAM, 500 GB HDD, Operating system : - windows server 2008 R2. The results (Graph Representation) of this machine are as follow.



Here four groups are presents whose name like Ram1, Ram2, Ram3, Ram4. Each group has separate heterogeneous data set. Ram1 represent 1 millions of heterogeneous data sets. Other groups having 5 millions, 10 millions, 15 millions & 20 millions heterogeneous data set respectively. All these groups are shown on X-axis on the graph & y-axis shows time taken (Nano Seconds) for each group. The results clearly shows that in some conditions MRS Sort & GPU Quick Sort give same results & in some conditions MRS Sorts is just better then GPU Quick Sort algorithm.

VI. Concluson

Now, it is clearly seen that proposed algorithm can do much better job over existing sorting algorithms. Both time & space complexities are optimized with this algorithm. The various algorithm prepared so far for sorting of large heterogeneous data set are discussed. It can be seen that none of the algorithm is optimized universally for all types of data set. Thus approach to develop optimized algorithm for affliction data set are being discussed and proposed. The new Algorithm proposed optimized the time & space complexity for heterogeneous data set, comprising of both alphanumeric values & string in different formats. The results shows an improvement of 10-20% in computational complexity compound with MRS sort & GPU Quick sort.

References

[1]. Naila Rahman, Rajeev Raman, “Adapting Radix Sort to the Memory Hierarchy”, Journal of Experimental Algorithmic (JEA), 6, p.7-es, 2001.

[2]. Danial, Navarro, Guinovart & Larriba pay,” Sorting on the SGI Origin 2000: Comparing MPI & Shared memory Implementations”, 19th IEEE Conference of the Chilean Computer Science Society (1999) current version available on 6th August 2002.

[3]. Shin-Jae Lee, Minsoo Jeon, Dongseung Kim & Andrew Sohn, “Partitioned Parallel sort 1”, IDEA**L** Journal of Parallel & Distributed Computing (2002) jpdc.

[4]. Danial, Jimenez- Gonzalez, J. J. Navarro, Josep [2002]”, the Effect of Local Sort On parallel Sorting Algorithms”, 10th IEEE Euromicro Workshop on Parallel Distributed & Network Based Processing(Euromicro-PDP’ (02) 2002.

[5]. Daniel, Navarro & Josep, “CC- Radix: a Cache Conscious Sorting Based on Radix Sort”, 11th IEEE Conference on Parallel, Distributed & Network-Based Processing(Euro-PDP’03) 2003.

[6]. Nadathur Satish “Designing Efficient Sorting Algorithm for many-core GPUs”,23rd IEEE International Parallel and Distributed Processing Symposium, May 2009.

[7]. N. Ramprasad and Pallav Kumar Baruah.2007.” Radix Sort on the Cell Broadband Engine” ,In Int.l Conf. High Perf. Comuting (HiPC) –Posters, 2007.

[8]. Shibdas Bandyopadhyay and Sartaj Sahni. 4-Feb 2011,” GRS - GPU Radix Sort For Multifield Records”, IEEE Explore High Performance Computing(Hipc), 2010 International Confrence on19-22 Dec. 2010 ,1-10, Dona Paula, 11824284,Dept. of CSE, University of Florida, Gainesville, FL 32611.

[9]. Daniel Jiménez-González, Juan J.Navarro, Josep-L. Larrba-Pey.2001**”,** Fast Parallel in-memory 64-bit sorting” Proceeding ICS '01 Proceedings of the 15th international conference on Supercomputing ACM New York, NY, USA ©2001.

[10]. Daniel Cederman and Philippas Tsigas.2008,”On sorting and load balancing on GPUs”, Newsletter ACM SIGARCH Computer Architecture News archive Volume 36 Issue 5, December 2008 ACM New York, NY, USA.

About The Authers

[1]**Avinash Shukla** was born in Jabalpur on 14th Oct.1979. He has done Msc.in Information Technology from MCNUJC Bhopal in 2007. He is presently doing PhD. in Computer Science & Engineering in from CMJ University, Shilong (Meghalaya), India.

[2]**Dr A.K Saxena, ME,** PhD (Engg.) from ABV- IIITM. He is the Member Board of Studies in Electronics & CSE, RGTU (2005-2008).Electrical Engg., Jiwaji University(1999-2001) and have the Administrative experience as Nominee for BE and MCA courses at various institutes, Assistant coordinator(PET).He is the member of FIETE, MIE, MISTE Societies and have the Industrial Experience at SICO, Indian Railway Construction Co. Ltd. He had presented 10 technical papers in various national and international conferences. His research area includes Multimedia Technology and Digital Watermarking; Microprocessor based system development and instrumentation.