Parallel sorting on ILLIAC array processor

MASUMEH DAMRUDI, KAMAL JADIDY AVAL
Computer Science
Islamic Azad University of Firuzkuh
Firuzkuh
IRAN

Abstract: Nowadays, we need to speed up solving computer problems such as sorting. Because of limitations in processor's speed, using parallel algorithms is inevitable. In parallel algorithms, because of cost limitations and architecture complexity, it's not suitable to increase the number of processors to speed up. This is also infeasible where we have plenty of data. In this paper we issue a new sorting algorithm. The algorithm is implemented on ILLIAC architecture. The aim of issued algorithm is to reduce total cost of sorting with a trade off between the number of processors and execution time.

Key-Words: Parallel algorithm, Sorting, Array processors, SIMD, ILLIAC, NSI.

1 Introduction

Sorting algorithms are very important in computer science. Using parallel architectures, parallel sorting algorithms are more important due to processors' connectivity. Sorting algorithms are basic part of sequential and parallel computations; they also have great usage in database engines. Such algorithms are studied for many years.

Donald Knuth has reported that computer manufacturers of the 1960s estimated that more than 25 percent of the running time on their computers was spend on sorting, when all their customers where taken into account[1]. There are many computational tasks which about half of them are sorting tasks.

Many parallel sorting algorithms are proposed recently with low execution time on complicated networks. They have great cost using many interconnected processors. These algorithms aren't efficient on huge amount of data. Some of these algorithms are overviewed as following:
Suel issued a sorting algorithm based on shuffle network. The algorithm is named shuffle sorting algorithm and has a cost of $\Omega(\log^2 n / \log\log n)$, where n is the number of data elements [2].

In Bitonic algorithm, in addition to complexity of computation, we should be aware of connectivities' complexity. The order of this algorithm is $O(\log^2 n)$[3]. Because of great connectivities and its growth by increasing data, this algorithm is not suitable for great data. The lower bound of sorting networks and shellsort algorithm is $\Omega(n\log^2 n/(\log\log n)^2)$ [4, 5]. Shellsort is one of sequential algorithms which is used in low depth networks. Although the structure of shellsort based algorithms is simple, analyzing the execution of these algorithms is difficult.

Quicksort algorithm's complexity is $O(n\log n)$ [6]. Kider proposed a sorting algorithm on heap tree where the worst running time is $O(n\log n)$. Batcher's Bitonic and Odd-Even merge sorts' cost order is $O(\log^2 n)$[7]. A recent implementation of this algorithm is done in 2005 [8].

A parallel sorting algorithm that merges k sorted lists on CREW PRAM model has a cost of $O(\log N + N\log k/P)$. P and N are respectively number of processors and total entrance time of lists. Sen and Scherson show an improvement on SIMD mesh with $O(\sqrt{N})$ steps which is called shearsort [9]. In 2004 a $\sqrt{n}\times\sqrt{n}$ mesh containing n data with $\sqrt{n}(\log n + 1)$ sorting time was proposed by Allen and Wilkinson [10]. A new architecture named Multi-Mesh of Trees (MMT) is invented recently. This architecture is a combination of multi mesh and mesh of trees. This architecture uses n^4 processors. Just one algorithm is issued on this architecture which is named Esort [11]. The algorithm's complexity is $O(\log n)$.

We're always searching for such an algorithm that doesn't have problems like: comprehension complexity, great amount of processors and impossibility of data elements increment using fixed number of processors.
Issuing a new algorithm on a simple architecture with a trade off between number of processors and execution time is the main aim of this paper.

2 ILLIAC topology

This topology was introduced by Unger in 1958. In this topology each processor has direct connectivity with four processors like mesh. In ILLIAC each processor has connectivity with next and previous processor and also with a processor in a distance of $\sqrt{N}$ next and $\sqrt{N}$ previous. N shows number of processors. An instance of this topology is shown in Fig. 1. Maximum distance between two processors is $\sqrt{N}$ and the number of processors should be power of two.

In ILLIAC, the following functions are used to connect processors to each other.

Assuming $m = \sqrt{N}$:

(1) $II_{+1}(x) = x + 1 \mod N$

(2) $II_{-1}(x) = x - 1 \mod N$

(3) $II_{+n}(x) = x + m \mod N$

(4) $II_{-n}(x) = x - m \mod N$

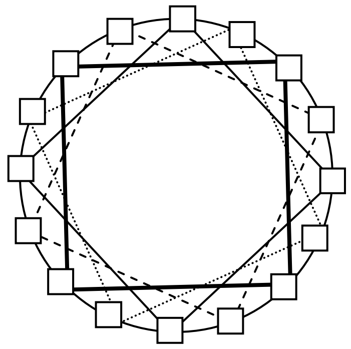

Fig. 1 ILLIAC network

3 Neighborhood Algorithm

Neighborhood sorting algorithm is one of odd-even transposition sorting algorithms. The number of processors in them is less than the number of data elements, so data elements are divided into processors [12].

This algorithm is implemented on linear architecture. Four data elements are stored in each processor. The running procedure of this algorithm is as following:

At each step, the processor sorts it's data elements and passes its two greater data elements to the next processor, resorts its data elements and passes its two lower data to the previous processor.

The above steps are iterated until the list becomes sorted.

Because of simplicity and the fact that SIMD architecture runs just one operation on many data elements, neighborhood algorithm is the base of our sorting algorithm. ILLIAC network has more connectivity in comparison to the Ring and Mesh architectures.

4 NSI algorithm

Sorting algorithms which store one data element in each processor are simple but their cost is not suitable. There are many data elements to sort in application programs and this number of processors is unavailable. Assume that N is the number of processors and n is the number of data elements. The relation between them is $n = 4N$. The number of data elements is quadruple of processors, so each processor needs four registers. The $\sqrt{N}$ last processors should have two local memories.

As each processor has four registers, on more data elements you can add registers or processors into the architecture.

Executing NSI[1] algorithm should follow these steps:

1. Four data elements enter each processor all at once.
2. A primary sorting executes in each processor and sorts the data stored in registers inside processor.
3. Each processor such as i (which i is less than $N - \sqrt{N}$) sends its two greater data elements to $i + \sqrt{N}$ and processors that are in the range of $N - \sqrt{N}$ to N store two greater data elements in their local memory. Each processor sorts its data elements.
4. Processor $i + \sqrt{N}$ sends its two smaller data elements to processor i and processors that are in the range of $N - \sqrt{N}$ to N restore two greater data elements from their local memory into their empty registers. Each processor sorts its data elements again.
5. Each processor such as i sends its two greater data elements to $i + 1$ and processor $N - 1$ stores two greater data elements in its local memory. Each processor sorts its data elements.
6. Processor $i + 1$ sends its two smaller data elements to processor i and processor $N - 1$ restores two greater data elements from its local memory into its empty registers. Each processor sorts its data elements again.

[1] Neighborhood Sort on ILLIAC

The above steps should iterate until data list becomes sorted.

In this algorithm each data element can move $\sqrt{N}+1$ in each execution. In the worst case the algorithm should run $\sqrt{N}-1$ times to sort data list. This will cause $2\sqrt{N}-2$ data exchanges between processors. The time order of NSI sorting on ILLIAC is $t_n = O(\sqrt{N})$ and the cost of processors is $p_n = O(N)$. The total cost will be $C_n = O(N\sqrt{N})$.

4.1 Analysis

We have described an algorithm to sort using ILLIAC architecture. In this architecture the maximum distance between two processors is $\sqrt{N}$, so there are $\sqrt{N}-1$ steps between the furthest processors. The algorithm is based on step by step passing data elements to their locations in the sorted list.

In this architecture, the worst case is where data elements are $\sqrt{N}-1$ steps far from their location in sorted list. In this case, executing the algorithm on list will cause $2\sqrt{N}-2$ data transpositions between processors. At last, all of data elements are in their proper location of sorted list.

The costs of previously issued algorithms and our algorithm are presented in the following table for comparison. In all cases N is the number of processors and n is the number of data elements.

NSI is an in-place sorting algorithm and doesn't need extra space to sort data elements. The algorithm is issued on EREW model so there is no need to gain access to the same memory location.

It seems that the total cost of NSI is not better than Multiway of merge and Count algorithms. Multiway of merge algorithm is a merge algorithm and input lists should be sorted, so the cost of sorting input lists should be considered too [13]. On the other hand this algorithm is issued on CREW model. Count algorithm is not sorting the list by comparing data elements and just can sort integer data elements.

As seen there are no such limitations in our algorithm. As shown in Table 1, NSI offers a better total cost in comparison to other algorithms.

Sorting	Execution time O( )	No. of Processor	Total cost O( )	Description
ILLIAC	$2^{\frac{1}{2}\log_2^N}\log_2^N$	N	$N2^{\frac{1}{2}\log_2^N}\log_2^N$	In 1977
Rank	$n\log n$	$n(n-1)$	$n^3\log n$	-
Bubble	$n^{1/2}\log n$	N	$Nn^{1/2}\log n$	-
Quick	N	N	N^2	-
Meshsort	$n\log^{k-1} n$	n^2	$n^3\log^{k-1} n$	K= length of path
Shell	$\frac{n\log^2 n}{(\log\log n)^2}$	n	$\frac{n^2\log^2 n}{(\log\log n)^2}$	-
Shuffle	$\frac{\log^2 n}{\log\log n}$	$n\log n$	$\frac{n\log^3 n}{\log\log n}$	-
Multiway of Merge	$\log n + \frac{n\log k}{N}$	N	$N\log n + n\log k$	on CREW PRAM -k sorted lists
Batcher's Bitonic	$\log^2 n$	$n\log n$	$n\log^3 n$	-
Esort	$\log n$	n^2	$n^2\log n$	In 2004
Heap	$n\log n$	$\log n$	$n^2\log n$	In 2004 by Kider
Count	n	$\log n$	$n\log n$	Just integers
Mesh	$\sqrt{n}(\log n)$	n	$n\sqrt{n}(\log n)$	In 2004 by Wilkinson & Allen
NSI	$\sqrt{N}$	N	$N\sqrt{N}$	$n = 4N$

Table 1 Comparison table [14, 15]

4.2 Simulation result of NSI algorithm

A result of simulating NSI algorithm is shown in Fig. 2. We wrote a program to simulate our algorithm. The program counts data element transpositions between processors in the process of sorting the list.

This simulation sorts 1024 randomly generated data elements using 256 processors. Number of processors for sorting simulation can be specified when running the program. Each processor in our simulation program uses four registers; so number of data elements is quadruple of processors.

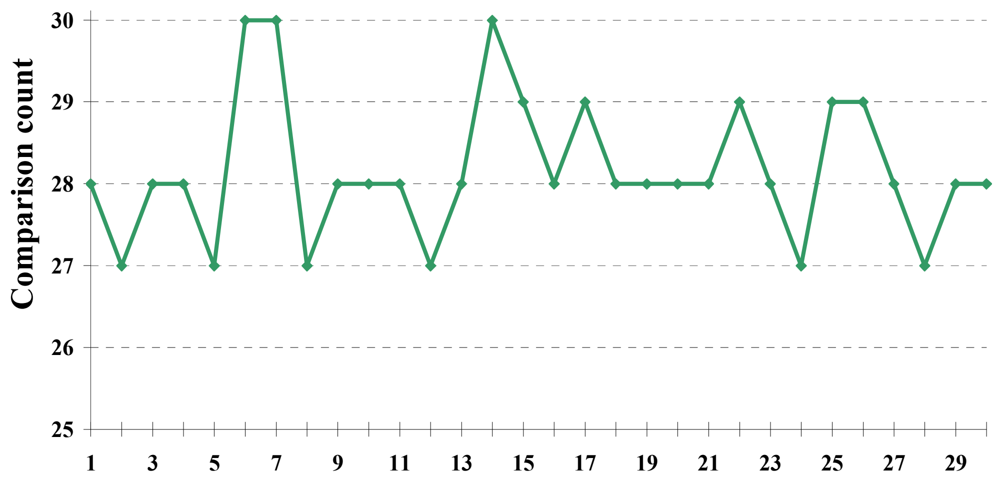

Fig. 2 Results of simulation for 30 iterations

5 Conclusion

There are many sorting algorithms which use parallel architectures. Recently issued algorithms are implemented on new architectures like MMT to improve execution time. MMT architecture is cost consuming on initialization. Many processors plus connection complexity increase its total cost too.

In this paper we issued an algorithm based on ILLIAC architecture named NSI.

NSI algorithm is an extension of neighborhood algorithm on ILLIAC. Neighborhood algorithm cost on linear architecture is $O(N)$.

NSI algorithm sorts the list in-place. Also the use of ILLIAC architecture causes faster sort in comparison to neighborhood algorithm on linear architecture. This is because of more connectivity in ILLIAC architecture. The sorting cost of NSI algorithm is $O(\sqrt{N})$.

References:

[1] Knuth, D.E. *sorting and searching*(Second Ed.), Volum 3 of *The Art of Computer Programming.* Addison-Wesley, Reading. MA, USA, 1998.

[2] Torsten Suel, M.S.C.S, *Routing and Sorting on Fixed Topologies*, Presented to the Faculty of the Graduate School of The University of Texas at Austin December, 1994.

[3] D. Nassimi and S. Sahni, Bitonic sort on a mesh-connected parallel computer, *IEEE Trans. Comput.* C27 (1), 1979, pp. 2-7.

[4] R. E. Cypher. A lower bound on the size of Shellsort sorting networks. *SIAM J. Comput,* 22, 1993, pp. 62-71.

[5] B. Poonen, The worst case in Shellsort and related algorithms. *Journal of Algorithms,* 15, 1993, pp.101-124.

[6] R. S. Francis and L. J. H. Pannan. A parallel partition for enhanced parallel quicksort. *Journal of Parallel Computing*, 18, 1992, pp.543-550.

[7] Batcher, K. E. Sorting networks and their applications. *AFIPS Spring Joint Computer Conference,* 32, 1968, pp. 307-314.

[8] Ahmed Shamsul Arefin, Mohammed Abul Hasan, An Improvement of Bitonic Sorting for Parallel Computing, *WSEAS Transactions on Information Science and Applications*. Issue 7, Volume 2, July 2005.

[9] I. D. Scherson and S. Sen, Parallel sorting in two-dimensional VLSI models of computation, *IEEE Trans. Comput.* 38,Feb.1989, pp.238-249.

[10] B. Wilkinson & M. Allen, *Parallel Programming Techniques & Applications Networked Workstation & Parallel computers* 2nd Ed, 2004 Pearson Education INC.

[11] Prasanta K. Jana, Multi-mesh of trees with its parallel algorithms, *Journal of Systems Architecture* 50, 2004, pp. 193-206.

[12] Park, A., and Balasubramanian, K., Reducing Communication Costs for Sorting on Mesh-Connected and Linearly Connected Parallel Computers, *Journal of Parallel and Distributed Computing*, 9, 1990, pp. 318-322.

[13] Z. Wen, Multiway merging in parallel,*IEEE Trans. Parallel Distrib. Comput.* 7, January 1996, pp. 11-17.

[14] D.E. Muller, F.P. Preparata, Bounds of complexities of networks for sorting and switching, *Journal of Assoc. Comput. Mech. 22,* April 1975, pp.195–201.

[15] S.G. Akl, *Parallel Sorting Algorithms*, Prentice-Hall, Orlando, FL, 1989.