

Comments on Integer Sorting on Sum-CRCW

Hazem M. Bahig

Computer Science Division, Department of Mathematics,
Faculty of Science, Ain Shams University, Cairo, Egypt.
email: hbahig@asunet.shams.edu.eg

Abstract- Given an array X of n elements from a restricted domain of integers $[1, n]$. The integer sorting problem is the rearrangement of n integers in ascending order. We study the first optimal deterministic sublogarithmic algorithm for integer sorting on CRCW PRAM. We give two comments on the algorithm. The first comment is the algorithm not runs in sublogarithmic time for any distribution of input data. The second comment is the cost of the algorithm is not linear. Then, we modify the algorithm to be optimal in sense of cost with a restriction on the input data. Our modification algorithm has time complexity $O(\frac{\log n}{\log \log n})$ using $\frac{n \log \log n}{\log n}$ Sum-CRCW processors. Also, the algorithm has linear space.

I. INTRODUCTION

Parallel computing are used to solve large problems faster than sequential computation. There are many parallel computational models are introduced. One of these models is the Parallel Random Access Machine (PRAM). The PRAM model is the most common general purpose model of parallel computations. It consists of an unbounded number, p , of processors working synchronously, where each processor is a random access machine (RAM) with respect a set of registers rather than a local memory. All processors share an unbounded global memory, via which they communicate.

During a given cycle each processor may read an element from global memory into its local register set, write a value from its local register set to global memory, or perform any RAM operation on data in its local register set.

Various PRAM models have been introduced, differing in the conventions regarding to concurrent reading and writing [1], [2]. These models are:

- (1) Exclusive Read Exclusive Write (EREW) PRAM: for each memory location, it may only be read from or written to by one processor in each cycle;
- (2) Concurrent Read Exclusive Write (CREW) PRAM: for each memory location, it may be

read from by several processors, or written to by only a single processor in each cycle.

- (3) Concurrent Read Concurrent Write (CRCW) PRAM: for each memory location, it may be read from or written to by several processors at the same time.

In case of concurrent writing, different assumptions are made about which processor's value is written into the memory location to resolve write conflicts [2], [7].

- Common CRCW: all processors must attempt to write the same value.
- Arbitrary CRCW: one of the processors succeeds.
- Priority CRCW: the smallest numbered among the processors succeeds.
- Sum CRCW: sum of all values being written to that location at that time.
- Min CRCW: the smallest of the values to be written is selected by using minimum.
- Tolerant CRCW: the contents of a cell do not change.

After solving the problem on one of these models, the parallel algorithm is measured by some criteria such as work, cost, and optimality [1], [2].

- The cost of a parallel algorithm, A , is defined as the product of the time complexity of A by the number of processor.
- The total number of operations performed by processors for solving problem Q by an algorithm A is called the work of the parallel algorithm, $W_p(n)$.
- A parallel algorithm is said to be optimal if the time-processor product matches the number of operations for the fastest known sequential algorithm for the problem.

The problem will be study in this paper is defined as follows: Given an array A of n elements such that the elements of A are taken from the restricted domain $[1, n]$. The integer sorting is rearrangement of n elements in ascending order. The integer sorting problem has many applications combinatorial problems and simulation.

The integer sorting is solved by strategies such as

bucket, count, and radix. The running time of integer sorting is linear [12].

In this paper, we study the first optimal sublogarithmic deterministic algorithm for integer sorting and comments on the algorithm. Finally, we modify it to be optimal in cost if either of the input data are uniformly distributed or $|x \in [\frac{(i-1)\log n}{\log \log n} + 1, i\frac{\log n}{\log \log n}]| = O(\frac{\log n}{\log \log n}), \forall 1 \leq i \leq \frac{n \log \log n}{\log n}$.

The paper consists of six sections. In Section 2, we give the previous works for integer sorting on CRCW PRAM. Akl and Chen algorithm was given in Section 3. In Section 4, we give the disadvantages of Akl and Chen algorithm. We try to solve some of these disadvantages in Section 5. Finally, in Section 6 we give the conclusion of our work and an open question.

II. PREVIOUS RESULTS

In this section we will give some of previous results for integer sorting and related problems.

(1) Prefix Sums [1], [2]: The prefix sums problem of an array $A = \{a_1, a_1, \dots, a_n\}$ of n nonnegative integers is computing the n prefix sums $s_i = \sum_{j=1}^i a_j, \forall 1 \leq i \leq n$. This problem can be performed in $O(\frac{\log n}{\log \log n})$ time using $\frac{n \log \log n}{\log n}$ CRCW PRAM processors.

(2) Integer Sorting on CRCW PRAM: Many algorithms are suggested to solve the integer sorting on CRCW PRAM [3], [5], [6], [8], [9], [11], [13], [14], [16], [17]. Some of these algorithms are optimal as in table 1.

Ref.	$p(n)$	$T_p(n)$	Kind	$W_p(n)$
[14]	$\frac{n}{\log n}$	$O(\log n)$	R	$O(n)$
[9]	$\frac{n}{\log n}$	$O(\log n)$	R	$O(n)$
[15]	$\frac{n}{\log n}$	$O(\log n)$	R	$O(n)$
[5]	$\frac{n \log \log n}{\log n}$	$O(\frac{\log n}{\log \log n})$	R	$O(n)$
[3]	n	$O(\frac{\log n}{\log \log n})$	D	$O(n)$

Table 1: Optimal Parallel Algorithms on CRCW PRAM for Integer Sorting.

Where, R=Randomized and D=Deterministic

III. AKL AND CHEN ALGORITHM

From Table 1, We observe that Akl and Chen algorithm [3] is the first optimal deterministic sublogarithmic for integer sorting. The algorithm is based on counting technique that computes the repetition of each element. The algorithm consists of the following steps:

Step 1: Initialize the elements of an auxiliary array B with 0, i.e., $b_i = 0, \forall 0 \leq i \leq n$.

Step 2: Processor p_i write 1 into $b_{a_i}, \forall 0 < i \leq n$.

Step 3: Apply prefix sum computation on array B and store the result in array C .

Step 4: For $0 < i \leq n$, do the following in parallel: each processor p_i test if $c_{i-1} < c_i$, then write i into $d_{c_i}, \dots, d_{c_i-b_i+1}$, where D is array represents the ordering of A .

lemma 1[3]. Linearly-ranged integers can be sorted in $O(\frac{\log n}{\log \log n})$ time with $O(n)$ work on a Sum-CRCW PRAM.

The following example illustrate the execution of the algorithm, where $n = 16$:

Given $A = \{9, 2, 14, 8, 4, 1, 5, 13, 7, 9, 11, 12, 6, 4, 10, 16\}$

Step 1: $B = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$

Step 2: $B = \{0, 1, 1, 0, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1\}$

Step 3: $C = \{0, 1, 2, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 15, 16\}$

Step 4: $D = \{1, 2, 4, 4, 5, 6, 7, 8, 9, 9, 10, 11, 12, 13, 14, 16\}$

IV. COMMENTS OF AKL AND CHEN ALGORITHM

Akl and Chen algorithm is very simple and consists of few steps. But, when we applying the algorithm on different data distribution, we found that the algorithm has some disadvantages.

The first comment: the algorithm is not runs in sublogarithmic time for any distribution of input data.

We prove this disadvantage by the following counter example. Let $n = 16$ and

$A = \{9, 2, 14, 9, 2, 9, 5, 4, 7, 9, 11, 9, 6, 9, 9, 5\}$

By applying the algorithm:

Step 1: $B = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$

Step 2: $B = \{0, 0, 2, 0, 1, 2, 1, 1, 0, 7, 0, 1, 0, 0, 1, 0\}$

Step 3: $C = \{0, 0, 2, 2, 3, 5, 6, 7, 7, 14, 14, 15, 15, 15, 16, 16, 16\}$

Step 4: $D = \{2, 2, 4, 5, 5, 6, 7, 9, 9, 9, 9, 9, 9, 11, 14\}$

It is clear that the repetition of the integer 9 is greater than $O(\frac{\log n}{\log \log n}) = 2$. So, the processor p_9 will do more than $O(\frac{\log n}{\log \log n})$ iterations to reallocate the integer 9 in step 4. Therefore, the algorithm not run in $O(\frac{\log n}{\log \log n})$ in general case.

In general, if the value of c_i is greater than $\frac{\log n}{\log \log n}$, say $O(\log n)$, then the processor p_i requires $O(\log n)$ time to do step 4. Therefore, the running time of the

algorithm is $O(\log n)$. This means that Akl and Chen algorithm does not run in $O(\frac{\log n}{\log \log n})$ time in general.

The second comment: the cost of the algorithm is not linear. The cost of the algorithm is equal to $O(\frac{n \log n}{\log \log n})$. Therefore, the algorithm is not optimal in the sense of cost.

V. MODIFICATION

In this section, we try to solve some of these comments. We use $\frac{n \log \log n}{\log n}$ Sum-CRCW PRAM processors. Also, assume that the integers are distributed uniformly over the range $[1, n]$ or the number of integers over the subrange $[\frac{(i-1) \log n}{\log \log n} + 1, \frac{i \log n}{\log \log n}]$ are equal, $\forall 1 \leq i \leq \frac{n \log \log n}{\log n}$.

We use the following variables in the algorithm:

- R is an array of length n to represent the repetition of each element in the array A .
- S is an array of length n to represent the prefix sums for R .

Our modification consists of the following steps:

Step 1: Initialize the elements of an array R with zero by using $\frac{n \log \log n}{\log n}$ processors.

Step 2: Repeat the following, $1 \leq i \leq \frac{\log n}{\log \log n}$ times:

update the value of $R(a_j)$ by writing 1 using processor p_j , where $\forall 1 \leq i \leq \frac{n \log \log n}{\log n}$ and $a_j = \frac{(i-1)n \log \log n}{\log n} + j$.

Step 3: Compute the prefix sum S for the inputs: $0, R(1), R(2), \dots, R(n-1)$ by using $\frac{n \log \log n}{\log n}$ processors, where $S(1) = 0$ and $S(i) = \sum_{l=1}^{i-1} R(l), \forall 2 \leq i \leq n$.

Step 4: Reallocate the non zero elements of R to the array A by using $\frac{n \log \log n}{\log n}$ processors. Each processor p_i starts from position $S(\frac{(i-1) \log n}{\log \log n} + 1)$, where $1 \leq i \leq \frac{n \log \log n}{\log n}$.

lemma 2. Linearly-ranged integers can be sorted in $O(\frac{\log n}{\log \log n})$ time with $O(n)$ optimal cost using $\frac{n \log \log n}{\log n}$ Sum-CRCW PRAM such that the n inputs are distributed uniformly, or the number of elements over the subranges $[\frac{(i-1) \log n}{\log \log n} + 1, \frac{i \log n}{\log \log n}]$ are equals, $\forall 1 \leq i \leq \frac{n \log \log n}{\log n}$.

VI. CONCLUSION

Many optimal integer sorting algorithm on CRCW PRAM are suggested. We have studied one of them which is Akl and Chen algorithm. We have given two comments of Akl and Chen algorithm. Then, we have proposed a correction of their algorithm under a condition. The algorithm is correct under either of them the input data are uniform distribution or the

number of elements over the subranges, $[\frac{(i-1) \log n}{\log \log n} + 1, \frac{i \log n}{\log \log n}]$, are equals. The proposed algorithm is optimal, deterministic, and running in sublogarithmic time. Also, the algorithm uses linear space. There exists a open question which is: *Can we design an optimal deterministic algorithm in sublogarithmic time without any restriction on Sum-CRCW?*

Note added in proof. Since this manuscript was accepted, we have been able to extend the results given here to any distribution of input data.

REFERENCES

- [1] S. Akl. Parallel Sorting Algorithms. Academic Press Inc. 1985.
- [2] S. Akl. Parallel Computation: Models and Methods. Prentice Hall, Upper Saddle River, New Jersey, 1997.
- [3] S. Akl and L. Chen. On the Power of Some PRAM Models. Journal of Parallel Algorithms and Applications. Vol 13, No 4, (1999), 307–319.
- [4] H. Bahig. Sublogarithmic Integer Sorting on CRCW. In Preparation.
- [5] H. Bast, and T. Hagerup. Fast Parallel Space Allocation, Estimation, and Integer Sorting. Information and Computation **123** (1995), 72–110.
- [6] P.C.P. Bhatt, K. Diks, T. Hagerup, V.C. Prasad, T. Radzik, and S. Saxena. Improved Deterministic Parallel Integer Sorting. Information and Computation **94**: (1991), 29–47.
- [7] D. Campbell, A survey of Models of Parallel Computation. Technical Report, Department of Computer Science, University of York, 1997.
- [8] B. Chlebus. A Parallel Bucket Sort. Information Processing Letters **27**: (1988), 57–61.
- [9] B. Chlebus. Parallel Iterated Bucket Sort. Information Processing Letters **31** (1989), 181–183.
- [10] R. Cole. Parallel Merge Sort. SIAM J. Computing **17** (1988), 770–785.
- [11] T. Hagerup. Towards Optimal Parallel Bucket Sorting. Information and Computation **75**: (1987), 39–51.
- [12] D. Knuth. The Art of Computer Programming: Sorting and Searching. Addison-Wesley, 1973.
- [13] Y. Matias, and U. Vishkin. On Parallel Hashing and Integer Sorting. In Proc. of 17th ICALP, Springer LNCS **443** (1990), 729–743.
- [14] S. Rajasekaran, and J.H. Reif. Optimal and Subalgorithmic Time Randomized Parallel Sorting Algorithms. SIAM J. Computing **18** (1989), 594–607.
- [15] S. Rajasekaran, and S. Sen. On Parallel Integer Sorting. Acta Informatica 29: 1–19, 1992.
- [16] R. Raman. The Power of Collision: Randomized Parallel Algorithm for Chaining and Integer Sorting. In Proc. 10th Conference on Foundation Software Technology and Theoretical Computer Science, LNCS **472** (1990), 161–175.
- [17] S. Saxena. Parallel Integer Sorting and Simulation Amongst CRCW Models. Acta Informatica, Vol. **33**, Issue 7, (1996), 607–619.