An improved reliability bound of a probabilistic parallel integer sorting algorithm

Alexandros V. Gerbessiotis

CS Department, New Jersey Institute of Technology, Newark, NJ 07102, USA

Abstract

It is shown how one can improve the reliability bound of the parallel sorting algorithm of Rajasekaran and Sen [7] that sorts uniformly distributed integer keys on a CRCW Parallel Random Access Machine (PRAM). The probability of success improves to $1 - 2^{-\Omega(n \log \log n / \log n)}$ from the previous bound of $1 - 2^{-\Omega(n / (\log n \log \log n))}$ while retaining the $\tilde{O}(\log n)$ time bound for sorting $n$ uniformly distributed integers on $n / \log n$ processors.

Keywords: Algorithm design - CRCW PRAM - parallel sorting - uniformly distributed integers - integer sorting

1. Introduction

We analyze the probabilistic algorithm of Rajasekaran and Sen [7, Section 6] for sorting uniformly distributed integers in an arbitrary range and improve upon the probability bound of the algorithm claimed in [7, Theorem 6.1]. In particular, we show that a much smaller fraction of the input sequences fail to be sorted by the algorithm in [7] within the claimed time bound than previously shown. All other input sequences can still be sorted within the claimed [7] bound of $O(\log n)$ using $n / \log n$ processors of a concurrent-read concurrent-write (CRCW) parallel random-access machine (PRAM).

Our improved analysis shows that the algorithm of Rajasekaran and Sen [7] fails with probability $2^{-\Omega(n \log \log n / \log n)}$ thus improving upon the bound of $2^{-\Omega(n / (\log n \log \log n))}$ established by Theorem 6.1 of [7]. This result complements and matches the algorithm of [4, Theorem 2.3] that establishes the same improved probabilistic bound; several improvements over the result claimed in [4] are
explained in more detail in section 3. The result expressed in Theorem 1 to follow highlights the improvement over Theorem 6.1 of [7, page 14]. In the remainder, a randomized algorithm has running time $\tilde{O}(f(n))$ if and only if there exists a constant $c > 0$ such that its running time, on any input of size $n$, is no more than $cf(n)$ with probability at least $1 - 1/n^\alpha$, for any $\alpha \geq 1$. For the remainder, $\log n$ is the natural logarithm of $n$.

**Theorem 1.** There exists an optimal randomized algorithm on the CRCW PRAM for sorting $n$ uniformly distributed integers in an arbitrary range. This algorithm uses $n/\log n$ processors and $\tilde{O}(\log n)$ time. The probability of success is $1 - 2^{-\Omega(n \log \log n / \log n)}$.

In order to establish Theorem 1, the following result related to the tails of the binomial distribution established by Theorem 7 of Chapter I of [1, pages 13-14] is claimed. Let $X_i$ be a sequence of independent Bernoulli random variables with each $X_i$ having mean $P$. Then the random variable $S(N, P) = \sum_{i=1}^{N} X_i$ has a binomial distribution $B(k; N, P)$, $k \leq N$, with parameters $N$ and $P$ ($Q = 1 - P$).

**Theorem 2 ([1]).** (i) Suppose $0 < P \leq 1/2$, $\epsilon PQN \geq 12$, and $0 < \epsilon \leq 1/12$. Then

$$P(| S(N, P) - PN | \geq \epsilon PN) \leq (\epsilon^2 PN)^{-1/2} \exp \left( -\epsilon^2 PN / 3 \right),$$

(ii) If $uQ > 2$, where $Q = 1 - P$, and $PN \geq 1$, then

$$P(S(N, P) \geq uPN) < (e/u)^{uPN}.$$

(iii) If $u \geq e$ and $u^2 PN \geq \log u$, then

$$P(S(N, P) \geq e \frac{u}{\log u} PN) \leq \exp \left( -uPN \right).$$

2. **A more refined probabilistic analysis of [7]**

A brief outline of the six computational steps of the probabilistic parallel sorting algorithm of [7, page 12-13] is provided. For further details we refer to Section 6 of [7]. The $n$ input keys occupy consecutive locations of the shared memory of an $n/\log n$-processor CRCW PRAM.

The algorithm of [7] in its first step sorts the $n$ input keys based on their $\lceil \log n + \log \log n \rceil$ most significant bits. In a second step, the $n$ keys are split into subproblems. A subproblem consists of those keys whose $\lceil \log n + \log \log n \rceil$ most significant bits have the same value. As a result of the first step sorting, keys with the same $\lceil \log n + \log \log n \rceil$ most significant bits occupy consecutive memory
locations. In order to realize this second step, each one of the $n/\log n$ processors is first assigned the same number of $\log n$ consecutive keys, and then these processors collectively scan the keys assigned to them to identify the subproblems, and whether they are small-size with $< C$ keys or large-size with $\geq C$ keys, where $C$ is a predetermined constant. During this operation, processors can collect additional information that will assist in assigning consecutively located small-size subproblems to processors in a balanced way, and coalescing large-size subproblems into a single sequence that will also be stored into consecutive memory locations. Each one of the small-size subproblems will be separately sorted in a fifth step using merge-sort, a stable optimal sequential sorting algorithm. The large-size subproblems are coalesced and copied to consecutive memory locations in the third step and sorted in a fourth step of the algorithm.

In a third step, the processors through a parallel-prefix computation [6] copy the large-size subproblems into consecutive memory locations. Let $M$ denote the overall size of these subproblems. Consequently, in a fourth step, the $M$ keys are sorted in optimal parallel time [3]. The result is a sorted sequence of the $M$ input keys of large-size subproblems.

In the fifth step, the small-size subproblems are sorted with merge-sort, with each processor sorting all the subproblems assigned to it. As a result of the information collected in the second step, each processor will be assigned the same number of consecutive subproblems. The result of the sorting of small-size subproblems by an individual processor is a locally sorted sequence of the keys of the small-size subproblems assigned to that processor. Coalescing all those keys for processors 1, 2, etc, one also obtains a globally sorted sequence of keys of the small-size subproblems. This can be achieved through a parallel-prefix computation in time $O(\log n)$.

In the sixth step, the two sorted sequences of small-size (sorted in the fifth step) and large-size (sorted in the fourth step) subproblems are merged in $O(\log n)$ optimal parallel time by using say, the merging algorithm of [8].

**Time analysis considerations.** The first step requires parallel time logarithmic to the number of keys [7], and so do the second and third steps as well as the merging-based sixth step. In [7], the fifth step requires $O(\log n)$ parallel time since any sequence of at most $C = \Theta(1)$ keys requires $O(C \log C) = \Theta(1)$ time in the worst-case for sorting, and the per processor time for sorting is thus $O(\log n)$ for the subproblems assigned to any given processor. One needs to establish that the
fourth step can be completed in $O(\log n)$ time to confirm the overall optimal-time behavior of the algorithm in [7].

In the remainder it is shown by way of Lemmas 1 and 2 that with probability at least $1 - 2^{-\Omega(n \log \log n / \log n)}$ the number $M$ of keys identified in the third and sorted in the fourth step is $O(n / \log n)$. Thus the running time of the fourth step is indeed $O(\log n)$. Theorem 1 then follows if one also takes into account the discussion of Section 6 of [7]. A suitable value for $C$ can be determined by way of Lemma 1.

**Lemma 1.** If $n$ balls are thrown uniformly at random into $n \log n$ bins, then the number of bins with at least $C = eA$ balls is no more than $Bn / \log n$ with probability at least

$$1 - \exp(-B(Ae - 2)n \log \log n / \log n),$$

for $A \geq 1$ and for any $B \geq e$, $n \geq 1$.

In [7, page 13], an analogue to Lemma 1 was proved in two steps by way of Chernoff bounds [2]. It was first claimed that the probability that any bin contains more than $C$ balls was $O(1 / \log^2 C)$, and then probabilistic arguments were made through Chernoff bounds as well about the number of bins with more than $C$ balls.

**Proof:** We establish and bound the probability $p(U, V)$ that at least $U$ of the bins have at least $V$ balls each. A combinatorial argument [5] shows that for $n$ balls and $n \log n$ bins and thus for $p = 1/(n \log n)$,

$$p(U, V) \leq \binom{n \log n}{U} \binom{n}{V} p^U = p_{UV}$$

and thus for $U = Bn / \log n$ and $V = C = eA$ we obtain after also using \(\binom{n}{k} \leq (ne/k)^k\), $1 \leq k \leq n$, the following.

$$p(Bn / \log n, C) \leq \binom{n \log n}{Bn / \log n} \binom{n}{C} Bn / \log n p^{BCn / \log n} \Rightarrow$$

$$p(Bn / \log n, eA) \leq \left( e \log^2 \frac{n}{B} \right)^{Bn / \log n} \left( \frac{n}{A} \right)^{eABn / \log n} \left( \frac{1}{n \log n} \right)^{eABn / \log n}.$$

The latter for $B \geq e$ and $A \geq 1$ gives.

$$p(Bn / \log n, eA) \leq \left( \log^2 n \right)^{Bn / \log n} \left( \frac{1}{\log n} \right)^{eABn / \log n} = \exp(-B(Ae - 2)n \log n / \log \log n) \quad (1)$$


A consequence of Equation (1) is that \( O(n/\log n) \) bins (subproblems) have at least \( C \) balls (keys). The next Lemma will establish that these bins (subproblems) collectively hold \( O(n/\log n) \) balls (keys) as well.

**Lemma 2.** If \( n \) balls are thrown uniformly at random into \( n \log n \) bins, then the bins with at least \( C = eA \) balls hold no more than \( BDn/\log n \) balls altogether, with probability at least \( 1 - \exp(-B(D - 2)n \log n/\log n) \), for \( A \geq 1 \), and for any \( n \geq 8 \), \( B \geq e \), \( D \geq e \).

**Proof:** Using an argument from [7], consider any collection of \( Bn/\log n \) bins out of the \( n \log n \) bins. It will then be shown that the number of balls falling into these bins is \( O(n/\log n) \) with the same asymptotic probability as that established in Lemma 1. If one fixes a collection \( L \) of \( Bn/\log n \) specific bins, the probability that a ball out of the \( n \) available ones falls into any of the bins in \( L \) is \( B/(\log n)^2 \). The number of balls falling into the bins of \( L \) is denoted by a random variable \( Z \) that is upper bounded by a binomial random variable \( B(k; n, B/(\log n)^2) \) similarly to [7, page 13].

Theorem 2, case (ii), is claimed with \( u = D \log n \) for any constant \( D \) such that \( D \geq e \), and with \( N = n \) and \( P = B/(\log n)^2 \).

For any \( n \geq 8 \) and \( D \geq e \), and \( B \geq e \) and \( A \geq 1 \), it is \( uQ > 2 \) and \( PN \geq 1 \), where \( Q = 1 - P \). Since \( uPN = DBn/\log n \) and \( e/u = e/(D \log n) \) we obtain, by way of Theorem 2, case (ii), the following.

\[
P(Z \geq uPN) \leq (e/u)^{uPN} \Rightarrow \\
P(Z \geq DBn/\log n) \leq (e/(D \log n))^{DBn/\log n} \Rightarrow \\
P(Z \geq DBn/\log n) \leq \exp(-DBn \log n/\log n) \cdot \exp(DBn(1 - \log D)/\log n). \tag{2}
\]

The bound above applies to the fixed collection \( L \). The number of ways of choosing \( L \), i.e. \( Bn/\log n \) bins out of \( n \log n \) bins, is given by the following expression where we use \( \binom{n}{k} \leq (ne/k)^k \), \( 1 \leq k \leq n \),

\[
\binom{n \log n}{Bn/\log n} \leq \left( \frac{e \log^2 n}{B} \right)^{Bn/\log n} \\
\leq \exp(2Bn \log n/\log n) \cdot \exp(Bn(1 - \log B)/\log n). \tag{3}
\]

Thus, the probability that any collection contains at least \( Bn/\log n \) balls is at most

\[
\exp(-DBn \log n/\log n) \cdot \exp(DBn(1 - \log D)/\log n) \cdot \exp(2Bn \log n/\log n) \cdot \exp(Bn(1 - \log B)/\log n) =
\]
Noting that $D \geq e$, $B \geq e$, then with probability $1 - \exp(-\Omega(n \log \log n / \log n))$, any collection of $Bn/\log n$ bins contains $O(n/\log n)$ balls.

As a result of Lemma 2, the number of keys that are being sorted in the fifth step of the algorithm in [7] is $O(n/\log n)$ and thus the deterministic comparison-based sorting [3] in that step requires $O(\log n)$ parallel time. The success probability of the algorithm to sort in time $O(\log n)$ is the smaller of the two probabilities derived by Lemma 1 and Lemma 2, and thus still $1 - \exp(-\Omega(n \log \log n / \log n))$. Theorem 1 then follows from [7].

3. Conclusion

The detailed reliability analysis summarized in Lemma 1 and Lemma 2 improves upon the reliability of the algorithm of Rajasekaran and Sen [7] as expressed by Theorem 6.1 in [7]. In particular Theorem 1 establishes a failure probability of $2^{-\Omega(n \log \log n / \log n)}$ thus improving upon the bound of $2^{-\Omega(n/(\log n \log \log n))}$ established by Theorem 6.1 of [7]. This matches the reliability of the algorithm in [4, Theorem 2.3] that establishes the same improved probabilistic bound. The analysis presented in this work somewhat improves upon the bounds obtained in [4]. Using our current notation, the work of [4, Corollary 1.1] claims that large-size subproblems with at least $2e$ keys collectively hold no more than $2e^2n/\log n$ keys with a reliability bound of $1 - 2^{-n \log \log n / \log n}$.

In this work our calculations for large-size subproblems with at least $2e$ keys ($A = 2$) determine an upper bound of $DBn/\log n$ for the number $M$ of keys in all large-size subproblems. For $B = D = e$ this implies an $e^2n/\log n$ upper bound for $M$. The corresponding reliability bound is the smaller of $1 - 2^{-B(Ae-2)n \log \log n / \log n}$ and $1 - 2^{-B(D-2)n \log \log n / \log n}$. The constant factor in the exponent for $A = 2$ and $B = D = e$ slightly improves upon the constant factor of 1 used in the reliability bound of $1 - 2^{-n \log \log n / \log n}$ cited in [4]. Another choice such as $A = 1$, $B = 2e$ and $D = e$ derives an $2e^2n/\log n$ bound for the number $M$ of keys in large-size subproblems, in line with [4, Corollary 1.1], but improving the constant factor in the reliability bound to $2e(e - 2)$ instead of the 1 in [4].

The research of the author was supported in part by the National Science Foundation (USA) under grant NSF/ITR IIS-0324816.


