Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

Performance evaluation of efficient parallel sorting algorithm for supercomputer

A.W.S. Loo, R.W.M. Yip, C.-W. Chung
Lingnan College, 15 Stubbs Road, Hong Kong

Abstract

This paper presents a parallel sorting algorithm (introduced in [LoYi91]) that makes use of the statistical properties of the elements to be sorted. Experiments are run on a MIMD computer. The results suggest that the new algorithm out-performs the parallel quicksort. Similar findings are obtained from a prior study using simulation.

Introduction

Quicksort [Hoar61] is a popular sorting algorithm. It has average complexity of O(n log n), but its performance degenerates to O(n^2) in the worst case when it divides a list of elements to be sorted into two uneven sized sublists which are to be sorted subsequently. Some proposals are raised to avoid the occurrence of the worst case [Erki84, Wain85] and improve the performance of quicksort [Sedg75, Sedg78, Fraz70], but some [Han91, NoAl85, JanLam85, YouEv84, Noga87] has observed the underlying statistical properties of the input list. In real-life applications, we can expect the input list follows certain kind of distributions. For example, data about independent events follows the Poisson distribution, and aggregated data from a set of samples approaches to the Normal distribution as the number of samples increase. Data collected from censuses, marketing researches, and so on, also exhibit certain distribution. With the knowledge of historical data, or by investigating the nature of the information, one could has some idea about the underlying distribution.

We have derived a statistical parallel sorting algorithm based on the assumption that we know the distribution of the input list. The algorithm is designed for a tightly coupled multiprocessors. Implementation is carried out on a Sequent computer with 9 processors to compare the performance of the new algorithm with that of the parallel quicksort algorithm. Dynamic scheduling method [Oste89] is used. Satisfactory results are collected which suggest that our new algorithm out performs the parallel quicksort.

The idea of the statistical parallel sorting algorithm comes from the statistical searching algorithm [Loo89, Loo90a] and statistical sorting algorithm for sequential computer [Loo90b, Loo90c].

Statistical Parallel Sorting Algorithm

Suppose we employ P processors to sort N elements that are uniformly distributed from *low_limit* to *up_limit*. The processors are named form 1 to P. The idea behind the algorithm is to partition the input list into hopefully P equal sized sublists according to the known distribution. Then, each processor takes a sublist and sorts it independently.

The algorithm consists of three phases. In the first phase, P dividers (namely X_0, X_1, ..., X_{P-1}) are calculated which are used to define P ranges, $[X_i, X_{i+1})$ for $i = 0, 1, ..., P-1$. As the data are uniformly distributed, Xi is chosen to be (*low_limit* + (*up_limit* - *low_limit* + 1) / P * i). In the second phase, the number of elements falling into each of these ranges is counted. This is done by assigning each processor a consecutive group of elements of the input list, and allowing the P processors concurrently count the numbers of elements fall into the P ranges. The results are combined to obtain the actual number of elements in each range.

In phase two, the processor i extracts all elements fall into the range $[X_{i-1}, X_i)$ from the input list, and forms a sublist. Other processors perform the similar task. This results in a new list which consists P sublists, with each element in the i^{th} sublist falls in the range $[X_{i-1}, X_i)$. In phase three, each processor sorts a sublist in the newly generated list using or Statistical Sorting method [Loo90a,Loo90b,Loo90c] or other efficient sorting methods.

Consider an example. Figure 1(a) shows a list, called A, having 24 numbers to be sorted by three processors. Assume we know in advance that the numbers follow the uniform distribution from 1 to 99. The corresponding X values are calculated as: 1, 34, 67, 100. Three ranges are then defined: [1, 34), [34, 67), [67, 100). Now, processor 1 is assigned the first 8 elements of A; processor 2 takes the next 8 elements; and processor 3 takes the rest. Each processor counts the number of elements falls into each of the 3 ranges. The counted values from the three processors are then added together to obtain the actual numbers of elements in each of the 3 ranges. For this example, the numbers counted are: 8, 7, 9.

In the second phase, the three processors concurrently scan the input list A for the elements falling in the range they are responsible for. Thus,

Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

processor 1 puts 18, 02, 21, 33, 13, 19, 17, 23 (which are all smaller then 33) into the first 8 locations of a new list called X. Processor 2 and 3 perform the similar task. Fig. 3(b) shows the list X after the phase two. In the third phase, each processor sorts a sublist by its own. Processor 1 responsible for the first 8 elements; processor 2 accounts for the next 7 elements; and processor 3 takes the rest 9 elements. Fig. 3(c) shows the final list produced by the algorithm.

A:

91	84	86	90	18	36	66	02	46	43	81	21	39	95	91	33	13	80

19	49	17	23	74	53

(a)

X:

18	02	21	33	13	19	17	23	36	66	46	43	39	49	53

↑ $0 \le X_i < 34$ ↑ $34 \le X_i < 67$ ↑

91	84	86	90	81	95	91	80	74

$67 \le X_i < 100$ ↑

(b)

X:

02	13	17	18	19	21	23	33	36	39	43	46	49	53	66

Processor 1 ↑ Processor 2 ↑

74	80	81	84	86	90	91	91	95

Processor 3 ↑

(c)

Figure 1 Example on Statistical Parallel Sorting Algorithm (a) input list; (b) the three sublist generated after phase 2; (c) each processor sorts a sublist as indicated by the arrows.

Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

The algorithm works as follows:

```
/* Statistical Parallel Sorting Algorithm           */
/* Assumptions:                                     */
/* 1) data elements are uniformly distributed       */
/*    from low_limit to up_limit                    */
/* 2) N is divisible by P                           */
/* Variables:                                       */
/* PID -- processor ID runs from 1 to P             */


/* Phase One */
for i = 0 to P
        range[i] = 1 + (up_limit - low_limit + 1) / P * i
do in parallel
        for i = (PID - 1) * N/P to (PID * N/P) do
                for j = 1 to P do
                        if (range[j-1] <= A[i]) and (A[i] < range[j]) then
                                begin
                                num[PID][j] = num[PID][j] + 1
                                exit the inner for loop
                                end
divider[1] = 0
for i = 1 to P-1 do
        begin
        temp = 0
        for j = 1 to P do
                temp = temp + num[j][i]
        divider[i+1] = divider[i] + temp
        end

/* Phase Two */
do in parallel
        for i = 1 to N do
                if (range[PID-1] <= A[i]) and (A[i] < range[PID]) then
                        begin
                        X[divider[PID]] = A[i]
                        divider[PID] = divider[PID] + 1
                        end
```

Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517


```
/* Phase Three */
do in parallel
        processor with PID use quicksort to sort sublist from X[num[PID-1]]
        to X[num[PID]]
```

The average case complexity of the algorithm is O(N + N/P * log (N/P)).

Other Distribution

This algorithm is similar to other algorithms [Noga87, YouEv84] when the data distribution is uniform. However, this algorithm does not calculate statistical information. Such information are collected by some other methods. Link list is not used in this algorithm.

However, this algorithm is quite different from others when the data distribution is normal (or other distribution). Phase one of our algorithm will be as follows:

```
/* Statistical Parallel Sorting Algorithm          */
/* Assumptions:                                    */
/* 1) data elements follow normal distribution     */
/*    with mean μ and standard deviation σ         */
/* 2) N is divisible by P                          */
/* Variables:                                      */
/* PID -- processor ID runs from 1 to P            */


/* Phase One */
for i = 0 to P
        range[i] = z_i * σ + μ,   where Prob[ X<= z_i ] = i/P
1 + (up_limit - low_limit + 1) / P * i
```

Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517


```
do in parallel
    for i = (PID - 1) * N/P to (PID * N/P) do
        for j = 1 to P do
            if (range[j-1] <= A[i]) and (A[i] < range[j]) then
                begin
                num[PID][j] = num[PID][j] + 1
                exit the inner for loop
                end
divider[1] = 0
for i = 1 to P-1 do
    begin
    temp = 0
    for j = 1 to P do
        temp = temp + num[j][i]
    divider[i+1] = divider[i] + temp
    end
```

Implementation

The statistical parallel sorting algorithm and the parallel quicksort are implemented on a Sequent computer having 9 processors. A series of uniform distributed data is generated by a random number generator to test the two algorithms. For the parallel quicksort, a queue is maintained to store sublist produced after a call to the quicksort procedure. Any idle processor can retrieve one sublist from the queue and works on it. It is found that a lot of time is wasted in executing the locking mechanism that is used to allow a process to get one sublist. While in the statistical parallel sorting algorithm, such problem is avoided. Figure 2 and 3 show the results.

Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

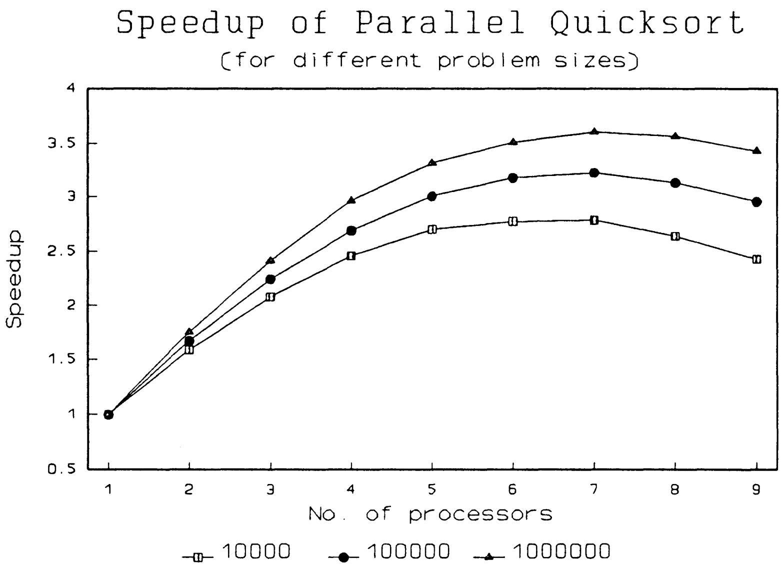

Figure 2 Speedup of Parallel Quicksort

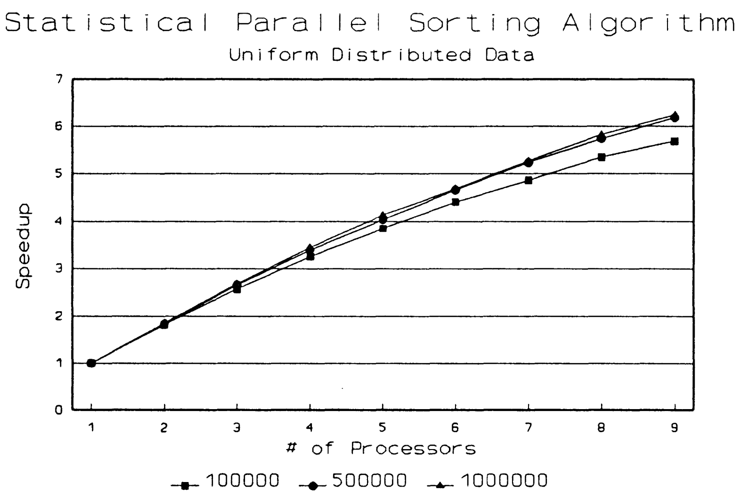

Figure 3 Speedup of Statistical Parallel Sorting Algorithm

Conclusion

We have presented a parallel sorting algorithm that makes use of the statistical properties of the input list. Experiments run on a Sequent multiprocessor with 9 processors shows satisfactory results which suggest that the new algorithm out-performs the parallel quicksort because the parallel quicksort has low parallelism at the beginning of its execution [Quin88, Akl85, Lori75]. The findings match with the results obtained from simulations carried out on an IBM-AT [LoYi91]. The main characteristics of the new algorithm is that it attains high parallelism during the whole process, and does not rely on the use of any locking mechanism.

8. References

[Akl85] Akl S. A., *Parallel Sorting Algorithm*, Academic Press, 1985.

[Erki84] Erkio H., "The Worst Case Permutation for Median-of-Three Quicksort", *The Computer Journal*, Vol. 27, No. 3, 1984.

[Fraz70] Frazer W. D. "Samplesort: A Sampling Approach to Minimal Storage Tree Sorting", *Journal of ACM*, Vol. 17, No. 3, July 1970.

[Han91] Handley C. C. , A Space Efficient Distributive Sort, Information Processing Letter, 37, 1991, p75-78.

[Hoar61] Hoare C. A. R., "Quicksort", *Computer Journal*, Vol. 5, No. 1, 1961.

[JanLam85] Janus P. & Lamagna A., An Adaptive Method for Unknown Distributions in Distributive Partitioned Sorting, IEEE Transactions on Computer, vol. c34, no4,1985.

[Loo89] Loo A. W., "Application of 'Key Distribution' in Very Large Database", *Proceeding of International ASME Conference*, Brighton, U.K., July, 1989.

[Loo90a] Loo A. W., "Application of Statistical Properties in Record

Searching", *Proceeding of 1990 Modeling and Simulation Conference*, Pennsylvania, USA, May, 1990.

[Loo90b] Loo A. W., "Statistical Sorting Algorithms", *Proceeding of ASME Conference*, N.C., USA, October, 1990.

[Loo90c] Loo A. W., "Pivot Selection in Sorting Algorithms", *Proceeding of ASME Conference*, China, 1990.

[Lori75] Lorin H., *Sorting and Sort System*, Addison Wesley, 1975.

[LoYi91] Loo A. W. & Yip R. W., "Statistical Parallel Sorting Algorithm", *International Conference on Concurrent Engineering & Electronic Design Automation*, Bournemouth, U.K., 1991.

[Noga87] Sorting in Parallel by Double Distributive Partitioning, BIT 37, 1987, p340-348.

[NoAl85] Noga M. T. & Allision D. , Sorting in Linear Expected Time, BIT, 1985, p451-465.

[Oste89] Osterhaug A., *Guide to Parallel Programming*, Prentice-Hall, 1989.

[Quin88] Quinn M. J., "Parallel Sorting Algorithms for Tightly Coupled Multiprocessors", *Parallel Computing*, 1988.

[Sedg75] Sedgewick R., "Quicksort". Ph.D. Thesis, Standford Computer Science Rep., Standford University, Standford, California, 1975.

[Sedg78] Sedgewick R., "Implementing Quicksort Programs", *Comm. of ACM*, Vol. 21, No. 10, pp. 847-857, 1978.

[Wain85] Wainwright R. L., "A Class of Sorting Algorithm Based on Quicksort", *Comm. of ACM*, 1985.

[YouEv84] Yousif N. Y. & Evans D. J. , Parallel Distributive Partitioned Sorting Methods, Intern. J. Computer Math., 1984, p.231-254.