Transactions on Information and Communications Technologies vol 9, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Efficiency measurement of distributed statistical sorting algorithms

A.W.S. Loo, C.W. Chung, R.S.W. Fu, M.S.K. Chan, J. Lo
Department of Computer Studies, Lingnan College,
15 Stubbs Road, Hong Kong

Abstract

This paper presents the evaluation results of a distributed sorting algorithm which was presented by the authors[10]. The evaluation is done by simulation. The algorithm makes use of the statistical properties of the data file. The objective of the algorithm is to minimize the number of messages required for the whole sorting process. The algorithm is designed to sort a very large file which is physically distributed over many sites (work-stations). The file size is so large that it is not feasible to transfer all data to a single node as no node has sufficient memory space for internal sorting. The sorting work will be shared by all sites involved and data will be sent along the lines communicating in bulk.

The Distributed System Model

The model was presented in [10] and is described again for easy reference as follows :

- a large file is physically distributed among P sites (workstations).
- N records are approximately uniformly distributed among P sites (i.e. N/P records for each site).
- no computer in any site has enough resource to collect all records and sorts the file locally (or does it efficiently).
- a single channel is available for broadcast communication.
- each record consists of a single element (note: in order to simplify the illustration of the algorithm only and it is a constraint of the model).
- we denote the elements in the file as X(i,j) where i is an integer (1 to N/P) and j is site number

- the objective of sorting is to rearrange X(i,j) elements so that the elements will be in the following order :

$$X(1,1) < X(2,1) < .... < X(N/P,1)$$
$$X(1,2) < X(2,2) < .... < X(N/P,2)$$
$$X(1,3) < X(2,3) < .... < X(N/P,3)$$
$$\vdots$$
$$X(1,P) < X(2,P) < .... < X(N/P,P)$$
$$\text{and } X(N/P,k) < X(1,k+1)$$

  where k is an integer from 1 to P-1
- the elements follow normal distribution (the algorithm will also be suitable for other distribution, but the normal distribution will be used as an example in this paper).

Statistical Sorting Algorithms

The statistical methods were applied successfully in sorting files for both conventional serial computers[3,4] and parallel computers[5]. Experimental and simulation results in earlier research works suggested significant improvements in statistical sorting algorithms in terms of number of exchanges and number of comparisons[3,4]. Improved "speedup" was obtained in the multi-processors environment[5].

However, the efficiency of distributed algorithms are usually measured in a different way. The communication time is usually much longer than the computation time. The communication time is thus a major criteria for measuring the performance of a distributed algorithm.

Furthermore, a major factor which affects the overall communication time is the number of communication messages. Therefore, the number of communication messages becomes the focus of discussion in most research works of this area[6,7,8]. The objective of the new distributed statistical algorithm in this paper is to reduce the number of communication message.

Distributed Statistical Sorting Algorithm

Phase 1 :
Every processor sorts its elements using some efficient sequential algorithm (such as quicksort[9], statistical sort[3,4]).
Phase 2 :
Find out the divisor D(k) using statistical search method (Details of this phase will be explained in next section of this paper) where k is an integer from 1 to P-1. After the sorting, D(k) will satisfy the following conditions:

$$X(N/P,1) <= D(1) < X(1,2)$$
$$X(N/P,2) <= D(2) < X(1,3)$$
$$\vdots$$
$$X(N/P,P-1) <= D(P-1) < X(1,P)$$

Phase 3 :

Each processor sequentially broadcasts and deletes all elements which do not belong to it's site. Other processors listen to the broadcast and then capture their elements by comparing the D(k) values. The X(i,j) elements will satisfy the following conditions:

$$X(i,1) <= D(1)$$
$$X(i,2) <= D(2)$$
$$\vdots$$
$$X(i,P-1) <= D(P-1)$$
$$\text{and } D(P-1) < X(i,P)$$

Phase 4 :

Each processor merges all elements to form a sorted subfile.

Statistical Search Algorithms

The distributed statistical sorting algorithm is quite similar to the distributed selectsort algorithm[8] except the statistical search method is used instead of binary search method in phase 3. In earlier research works[1,2], the number of searches required for the statistical search method is less than the binary search.

Steps in Phase 2

Step 1:

One processor will be selected as coordinator. It will calculate the initial D(k) values. The calculation will be as follows :

M : mean of key values
SD : standard deviation of key values
Prob : cumulative probability
F : cumulative probability function of the standard normal distribution

1. calculate Prob(k) = (N/P) * k
2. look up the corresponding Z(k) value from the standard normal cumulative probability table.
3. calculate D(k) = Z(k) * SD + M

Step 2:

The coordinator broadcasts the D(k) values to other processors.

Step 3:

Other processors broadcast the rank R(i,k) of D(k) in it's local elements (i.e.

the numbers of local elements which is smaller than D(k). The coordinator will also calculate it's own R(i,k) value.

Step 4:

The coordinator recalculates D(k) if $\sum R(i,k)$ does not equal (N/P) * k. Go back to step 2 until all D(k) satisfy this condition. The new D(k) will be calculated as follows:

1. calculate Prob(k) = Prob(k) + ((N/P)*k) - $\sum R(i,k)$) / N
2. look up the corresponding Z(k) value from the standard normal cumulative probability table.
3. calculate D(k) = Z(k) * SD + M

Examples

Consider an example consisting of a list called **A** of 24 numbers to be sorted by three processors. Assuming we know in advance that the numbers follow the uniform distribution from 10 - 100. Now, *processor 0* is assigned the first 8 elements of **A**, *processor 1* takes the next 8 elements and *processor 2* takes the rest as shown in figure 1.

We first calculate the initial **Target_Value** (they are 40, 70 and 100) and **Target_Rank** (they are 8, 16 and 24) as shown in figure 2. Moreover, each processor contains the values of **max_R** (= 24), **min_R** (= 1), **max_V** (= 100) and **min_V** (= 10). In the second phase, we count the numbers less than or equal to the **Target_Value** and store in each processor's **Total_Rank**:

1. for **Target_Value**=40, **Total_Rank**=10 in processor 0. It means that there are ten numbers <= 40 as marked with * in figure 1.
2. for **Target_Value**=70, **Total_Rank**=15 in processor 1.
3. for **Target_Value**=100, **Total_Rank**=24 in processor 2.

In the final phase, we re-calculate the **Target_Value** since **Target_Rank** ≠ **Total_Rank** as shown in (I) and (II) of figure 3.

For case (I), we have **min_R** = 1, **max_R** = 10, **min_V** = 10 and **max_V** = 40. Since **Total_Rank** > **Target_Rank**, we need a smaller **Target_Value** in order to make **Total_Rank** equal to **Target_Rank.**

$$\text{Target\_Value} = \frac{\text{Target\_Rank} - \text{min\_R}}{\text{max\_R} - \text{min\_R}} \times (\text{max\_V} - \text{min\_V}) + \text{min\_V}$$

$$= \frac{8 - 1}{10 - 1} \times (40 - 10) + 10$$

$$= 33\frac{1}{3}$$

For case (II), we have **min_R** = 15, **max_R** = 24, **min_V** = 70 and **max_V** = 100. Similarly, we require a larger Target_Value to make **Total_Rank** matching **Target_Rank**.

$$\text{Target\_Value} = \frac{16 - 15}{24 - 15} \times (100 - 70) + 70$$
$$= 73\frac{1}{3}$$

Then, we repeat the second and final phases until all the **Target_Rank** = **Total_Rank**.

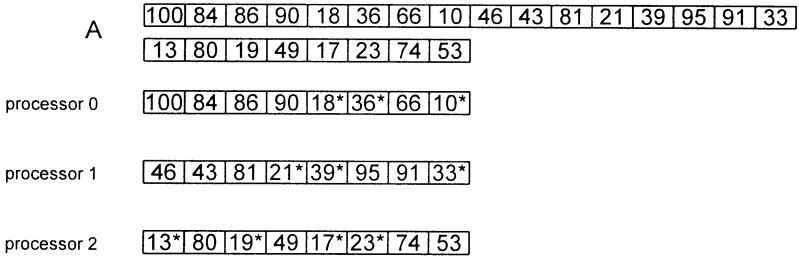

Figure 1

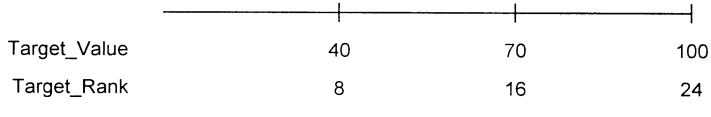

Figure 2

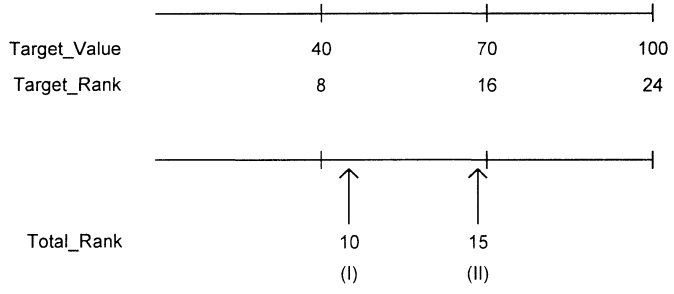

Figure 3

Evaluation Results

Programs were written in C++ to simulate the sorting process in the distributed system. Data files which followed normal distribution were generated. Numbers of communication messages which were required in the sorting process were recorded. The simulation results were presented in Figure 4 and 5.

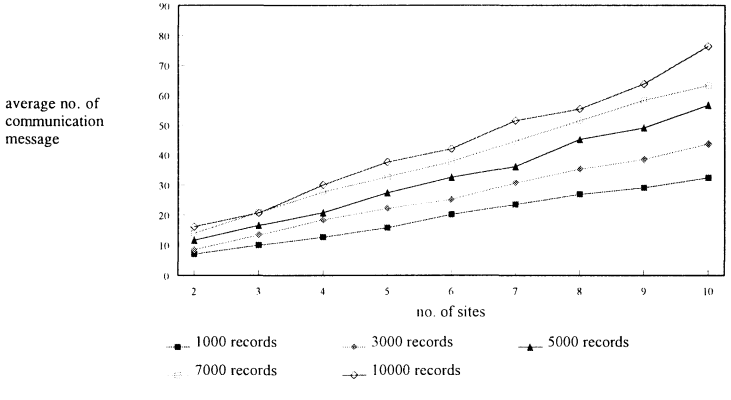

Figure 4

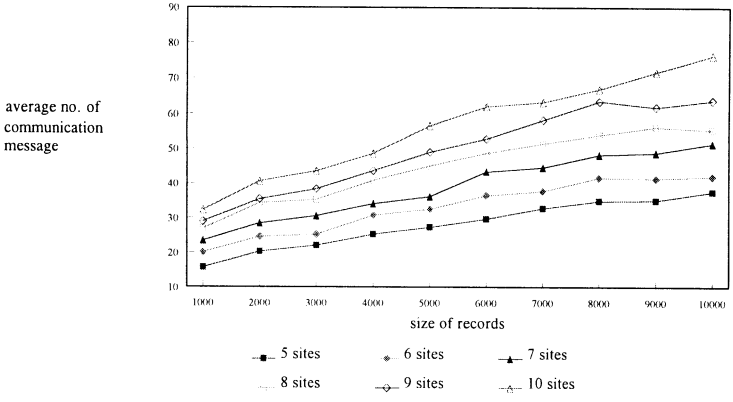

FIGURE 5

Conclusion

The performance of the sorting algorithm is very steady when number of records and number of sites increases.

Reference

1. Loo, A.W.S., "Application of 'Key Distribution' in Very Large Database", *Proc. of Intern. ASME Conf.*, Brighton (U.K.), July 1989.
2. Loo, A.W.S., "Application of Statistical Properties in Record Searching", *Proc. of 1990 Modelling and Simulation Conference*, Pennsylvania, U.S.A., May 1990.
3. Loo, A.W.S., "Pivot Selection in Sorting Algorithms", *Proc. of ASME Conference*, China, 1990.
4. Loo, A.W.S., "Statistical Sorting Algorithms", *Proc. of Signal and Systems*, U.S.A., 1990.
5. Loo, A.W.S., "Parallel Statistical Sorting Algorithm", *Proc. of Intern. Conf. Concurrent Engineering*, U.K., 3/1991.
6. Dechter, R. & Kleinrock, L., "Broad Communications and Distributed Algorithms", *IEEE Transactions Computers*, vC-35, n3, 1986.
7. Huang, J.H. & Kleinrock, L., "Distributed Selectsort Sorting Algorithms on Broadcast Communication Networks", *Parallel Computing*, 1990.
8. Wegner, L.M., "Sorting a Distributed file in a Network", *Proc. of Conf. Information Sci. Systems*, Princeton, New Jersey, 1982.
9. Hoare, C.A.R., "Quicksort", *Computer Journal*, 1961.
10. Loo, A.W.S. & Ng, Jim M., "Distributed Statistical Sorting Algorithms", *Proceeding of Sicon*, 1991.