Fast Parallel Sorting Algorithm Using Subsets and Quick Sort

A. Bosakova-Ardenska, N. Vasilev, L. Kostadinova-Georgieva

***Abstract*— This paper presents fast parallel sorting algorithm which uses division in subsets and quick sort algorithm. The proposed algorithm is evaluated by analytic way and results shows that it will be faster than algorithm in [3] when time for division in subsets of all set is bigger than time for sending/receiving of partial subsets.**

***Index Terms* — parallel sorting algorithm, quick sort, subsets**

I. INTRODUCTION

NEARLY 25% of the time during which the computer works he sorts data [1]. The sorting is operation which is a part of many algorithms. There are many sorting algorithms- for common or special cases. In the group of common (universal) sorting algorithms are: bubble sort, selection sort, insertion sort, merge sort, quick sort and etc. In group of private sorting algorithms are lexicographical sort, bucket sort and other. One of most popular methods for acceleration when it is necessary to sort big data arrays is using multiprocessors system. In this case is necessary to think about parallel architecture and memory distribution to choose appropriate parallel algorithm.

II. PARALLEL SORTING WITH QUICK SORT

On figure 1 is shown processes allocation for "classic" parallel quick sort algorithm [2]. For this algorithm number of parallel processes depends of given row. In other words for rows with same number of elements but different character (number of inversions) the number of parallel processes will be different.

In [3] is presented parallel sorting algorithm which is faster than "classic" parallel quick sort. In this algorithm number of parallel processes doesn't depend of rows character. On figure 2 is shown principle scheme of this algorithm.

Atanaska Dimitrova Bosakova-Ardenska works in University of Food Technologies, Plovdiv in Bulgaria. She is associated professor in department of "Computer Systems and Technologies"(a_bosakova@uft-plovdiv.bg).

Naiden Borisov Vasilev was head of department of "Computer Systems and Technologies" in Technical University Sofia, branch in Plovdiv, Bulgaria. (e-mail: mnvasilev@yahoo.com).

Lena Filipova Kostadinova-Georgieva is head of department of "Computer Systems and Technologies" in University of Food Technologies, Plovdiv, Bulgaria (e-mail: lenakostadinova@yahoo.com).

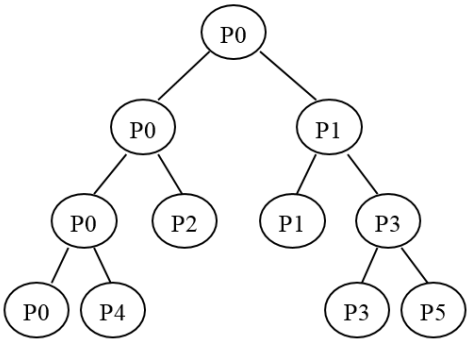


Fig.1 Process allocation

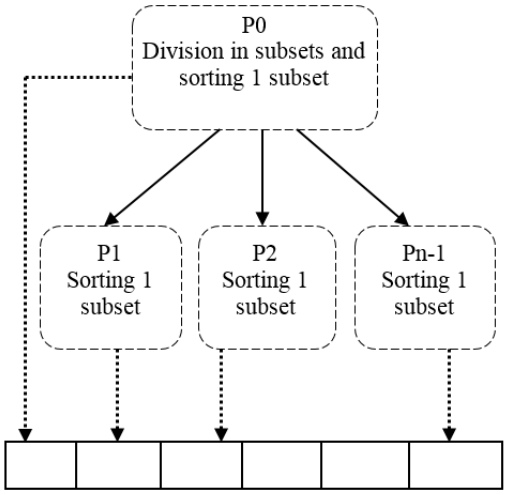


Fig.2 Principle scheme of algorithm

III. FAST PARALLEL SORTING ALGORITHM

The main idea is to divide the given big set of numbers in small subsets. And then parallel sort these subsets with fastest known universal sorting algorithm- quick sort. In other words is used the technique "divide and conquer". In this paper will be discussed one modification of parallel algorithm presented in [3] which will "parallelize" division in subsets (sub-rows).

Let we use MPMD (Multiple Program Multiple Data) model in version "Master/Slave" [4]. The "Master" process will execute the next steps as follow:

1) Open selected by user file with numbers for sorting;
2) Read numbers in one array;
3) Calculate count of elements in initial subsets

considering number of parallel processes (n). Every process must receive nearly equal piece of data;

4) Send to other processes initial subsets (from 1 to n-1). The first subset will stand in master process;

5) Divide initial subset in n partial subsets. The steps are:

- calculate bounds for subsets. The number of bounds is equal to n+1. The first bound is 0 and last is max (the maximal value for given numbers). The other bounds are calculated by formula:

$$Bi = i*]max/np[, \ i=1,2 \text{ to } n-1 \qquad (1)$$

where with Bi is noted i-th bound. The expression "] max/np [" indicates that the np divides max as integer;

- allocate memory for partial subsets;
- divide numbers in partial subsets and store in other array count of elements for every partial subset;
- send to other processes count of elements for their partial subset;
- send to other processes their partial subset;
- receiving from other processes first count of elements for own partial subset and second – partial subset;

6) Sort own subset (which is built from n partial subsets) with quick sort algorithm;

7) Receive from slave processes their sorted subsets;

8) Save sorted row in file.

Every "Slave" process will execute the next steps as follow:

1) Receive initial subset;

2) Divide initial subset in n partial subsets. The steps are:

- calculate bounds for subsets. The number of bounds is equal to n+1. The first bound is 0 and last is max (the maximal value for given numbers). The other bounds are calculated by formula (1).
- allocate memory for partial subsets;
- divide numbers in partial subsets and store in other array count of elements for every partial subset;
- send to other processes count of elements for their partial subset;
- send to other processes their partial subset;
- receiving from other processes first count of elements for own partial subset and second – partial subset;

3) Sort own subset (which is build from n partial subsets) with quick sort algorithm;

4) Send to master process sorted subset.

On Figure 3 is shown consolidated flow chart of parallel algorithm.

Let show one example for sorting with our fast parallel sorting algorithm. The initial conditions are:

- given row has 20 integers;
- number of parallel processes is 4 (n=4);
- range of numbers is: 1 to 100.

In this case bounds will be 5 and they will have the values: 1, 25, 50, 75 and 100. The given row is: 90, 12, 55, 80, 3, 76, 92, 45, 33, 20, 13, 48, 55, 60, 100, 1, 10, 29, 77, and 81. In table 1 are shown initial subsets (parallel step 1).

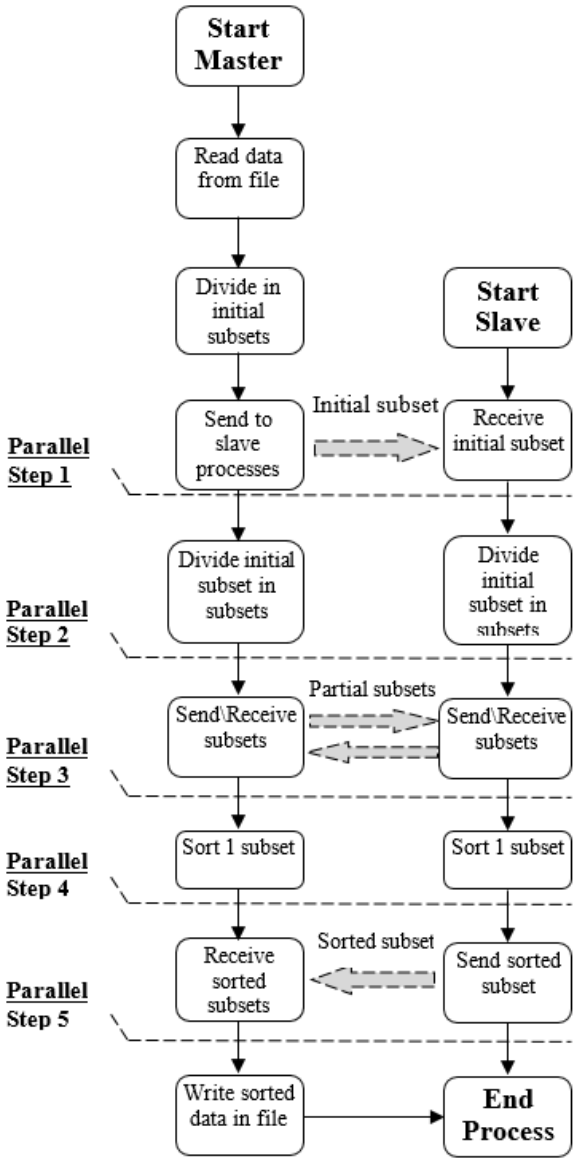


Fig. 3 Consolidated flow chart of parallel algorithm

TABLE 1
PARALLEL STEP 1 (INITIAL SUBSETS) FOR 4 PROCESSORS

Process 1	Process 2	Process 3	Process 4
90, 12, 55, 80, 3	76, 92, 45, 33, 20	13, 48, 55, 60, 100	1, 10, 29, 77, 81

After division of initial subset in partial subsets every process will have 4 partial subsets as shown in Table 2 (parallel step 2).

TABLE 2
PARALLEL STEP 2 (PARTIAL SUBSETS) FOR 4 PROCESSORS

Process / Subset	Process 1	Process 2	Process 3	Process 4
1 (1-25)	12, 3	20	13	1, 10
2 (25-50)		45, 33		29
3 (50-75)	55		48, 55, 60	
4 (75-100)	90, 80	76, 92	100	77, 81

After sending/receiving of partial subsets every process will have own subset as shown in Table 3 (parallel step 3).

TABLE 3
PARALLEL STEP 3 (SUBSETS FOR SORTING) FOR 4 PROCESSORS

Process 1	Process 2	Process 3	Process 4
12, 3, 20, 13, 1, 10	45, 33, 29	55, 48, 55, 60	90, 80, 76, 92, 100, 77, 81

The next step is sorting and after this every process will have sorted subset (parallel step 4) which is shown in Table 4.

TABLE 4
PARALLEL STEP 3 (SORTED SUBSETS) FOR 4 PROCESSORS

Process 1	Process 2	Process 3	Process 4
1, 3, 10, 12, 13, 20	29, 33, 45	48, 55, 55, 60	76, 77, 80, 81 90, 92, 100

It is obviously that sequential writing of subsets of process 1 to process 4 will give all sorted row (set).

IV. EVALUATION OF ALGORITHM

Time for sorting with algorithm presented in [3] depends of n and is noted with t1.

$$t1 = tds1 + tc1 + ts1 + tc1 = tds1 + 2tc1 + ts1 \quad (2)$$

In formula 2 tds1 is time for division in subsets, tc1 is time for sending/receiving of subsets and ts1 is time for sorting of one subset with quick sort algorithm. Time for sorting with our fast parallel sorting algorithm is presented with formula 3.

$$t2 = tcis2 + tdps2 + tcps2 + ts2 + tcs2 \quad (3)$$

In formula 3 tcis2 is time for sending/receiving of initial subsets, tdps2 is time for division in partial subsets, tcps2 is time for sending/receiving of partial subsets, ts2 is time for sorting of subset with quick sort algorithm and tcs2 is time for sending/receiving of subsets.

To be our algorithm faster than algorithm in [3] it is necessary to satisfied inequality:

$$t2 < t1 \quad (4)$$

Because initial partial subset is n times smaller than all set we can think that tdps2 is n times smaller than tds1. The time for sorting subsets ts1 and ts2 are equal because the subsets are the same. Let us assume that tcis2 is equal to tc1 - this means that subsets are almost equal by number. In this case inequality 4 will be:

$$t_{ds1} > \frac{n}{n-1} t_{cps2} \quad (5)$$

where *n* is number of subsets (parallel processes).

V. CONCLUSION

In this paper was presented fast parallel sorting algorithm which modify algorithm in [3]. The proposed algorithm "parallelizes" division in subsets (sub-rows) and use MPMD (Master/Slave) model. The algorithm is appropriate for performance on supercomputers and probably will be faster than "classic" parallel quick sort and algorithm with division in subsets [3] in many cases.

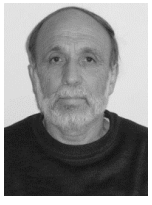
In future will be interesting to evaluate experimentally proposed algorithm and compare them with other known parallel sorting algorithms.

REFERENCES

[1] Knuth D., The art of computer programming, V3. Sorting and Searching, Addison Wesley Publishing Company, 1973.
[2] Wilkinson B. and Allen M., Sorting Algorithms, Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice-Hall, 1999.
[3] Bosakova-Ardenska A., N. Vasilev, I. Fillipov, Fast parallel sorting based on quick sort, TechSys 2013, Journal of the Technical University Sofia, branch Plovdiv, "Fundamental Sciences and Applications", Vol. 19, 2013, ISSN 1310-8271, pp 35-40.
[4] Joseph JaJa, An Introduction to Parallel Algorithms, Addison-Wesley publishing company, 1992.

BIOGRAPHIES

ATANASKA D. BOSAKOVA-ARDENSKA was born in 1980. She received the M.Sc. degree of Computer Systems and Technologies at Technical University of Sofia, Plovdiv branch 2004. She receives Ph.D. in 2009 with thesis "Parallel information processing in image processing systems". From 2010 she is assistant in department of Computer Systems and Technologies in University of Food Technologies. From 2014 she is associated professor by "Synthesis and Analysis of Algorithms" in department of Computer Systems and Technologies in University of Food Technologies in Plovdiv, Bulgaria. She is member of USB (Union of Scientist in Bulgaria). Her research interests include: parallel algorithms, sorting algorithms, image processing, MPI (Message Passing Interface), C++ programming.
Contacts: aardenska@abv.bg, a_bosakova@uft-plovdiv.bg



NAIDEN B. VASILEV is associate professor in department of "Computer Systems and Technologies" at Technical University of Plovdiv. He is receives Ph.D. in 1976. His research interests include: parallel algorithms, discrete mathematics and music.
Contacts: mnvasilev@yahoo.com

LENA Ph. KOSTADINOVA-GEORGIEVA receives M.Sc. degree of Automation in technical University Sofia 1975. She receives Ph.D. in 2003 with thesis "Automation of grading and sorting of certain fruit and vegetables". From 2005 she is associated professor by "Automation, computing and control systems". She is head of department of "Computer Systems and Technologies" in University of Food technologies, Plovdiv, Bulgaria. Her research interests include: application of cybernetic methods (management, pattern recognition, diagnosis, machine vision) for objective analysis and classification of foods.
Contacts: lenakostadinova@yahoo.com