International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-2, December 2012

Introducing Direct Mapping Sorters For Parallel Sorting Algorithms

Pushpendra Kumar, Priyanka Tyagi, Smriti Joshi

Abstract*— Sorting is one of the most basic problems of computer science and has been discussed continuously since the evolution of computer science. Several algorithms have been devised and applied and the work is still unfinished. For the parallel computing sorting is of same relevance as for sequential and very primitive problem domain too. Grain size is very important aspect of any parallel algorithm and is decisive in term of complexity. For the sorting problems minimum unit for sorting is two elements, since we apply a swap operation if required, and the two elements are sorted. This is considered to be the single step operation. In this paper we will increase primitive unit to four elements and four elements will be sorted in a single step. By applying this technique we can improve the performance of many parallel algorithms.

***Keywords*—Parallel sorting; Bitonic; shear sort; Direct mapping.**

I. INTRODUCTION

A. A swap operation is

Swap (Element A, Element B)

The values of A and B are interchanged in this operation. This is very basic operation of most of the sorting algorithms whether they are sequential or parallel.

B. Designing direct mapping sorters

For parallel algorithms we are proposing a new kind of basic sorting operation which does not use swapping. Instead it uses a direct mapping technique which sorts some fixed number of elements in a single step. This technique is faster than swapping because internally it itself uses parallelism and directly writing the results to memory in O(1) enhances the performance.

The basic idea to implement this theory comes from DNA analysis. For finding the parent of X, the DNA sample of X can be matched from a pool of DNA samples. Each DNA sample in pool has some identification. There are two possibilities now, either there is someone with same DNA pattern as DNA pattern of X or no parent of X is in the pool. If any pattern is matched we can map the parent of X in a single step if all DNA patterns have unique ID. Second case is not considerable for us because we will create a complete DNA pool and nothing should stand out of this pool.

C. Implementing basic sorters in different algorithms

After designing a fast direct mapping sorter we can use it as basic unit in different well recognized algorithms. We have used DMS_4(which is single step 4 integer sorter) in bitonic sort and Shear sort algorithm later in this paper.

Manuscript published on 30 December 2012.
* Correspondence Author (s)
Pushpendra Kumar, Meerut Institute of Engineering and Technology, Meerut.
Priyanka Tyagi, Meerut Institute of Engineering and Technology, Meerut.
Smriti Joshi, Meerut Institute of Engineering and Technology, Meerut.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

D. Terms used for direct mapping

First thing is a Direct mapping sorter which is a small, efficient function.

Second thing is pattern.

In any sequence of elements (with respect to sorting), there are only two possible filaments, A<B and A>=B.

We can represent these two only by 1 and 0.

For example the possible pattern for sequence 1, 7, 3, 5, 5, 6 can be 101010 (last 0 represents comparison between 6 and 1). We can identify a unique sequence which can be used to identify each possible combination of a given set of elements (All the elements should be comparable). Chosen pattern may vary from designer to designer. It is more or less an assumption rather than fix technique.

Pattern is used to find unique index to find the solution sequence in DNA pool.

Third thing is calculating the pattern correction sequence (parent) for creating the DNA pool. DNA pool is a static pool and several copies of DNA pool can be distributed. It is created statically and is a read only data structure.

Fourth thing is a **direct mapping function** (optional), which maps pattern to DNA pool. Sometimes it is possible to directly find index from the patterns to identify correct sequence from DNA pool.

Fifth term is **DNA pool**. As in our basic assumption we create all the correct sequence indices in DNA pool and no solution should stand out of this pool.

We can write the steps to create a DMS as following.

In a Direct mapping sorter

- To sort k elements k processors will be required.
- Each processor has a copy of DNA pool to find solution pattern.
- Each processor can calculate DNA pool index itself using pattern.
- After calculating the index it will get the value to write in array from copy of this array using the index stored in DNA pool.
- All the processors can run in parallel.

II. DESIGNING A DMS

A. Designing the primitive direct mapping sorter for two integers written in an array two[2]={A,B},(there is copy of array two[] available with name copy_of _two[2])

Possible cases	correct Pattern	index
A<B	1,2	0
A>=B	2,1	1

int DNA[2][2]={2,1,1,2}.
int Two[2]=copy_of_two={x,y}.
//x,y are two integers
DMS_2 (int two[0], int two[1])
{

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

two[0]=copy_of_two[DNA[B<A][0]].
// for Processor 1
two[1]=copy_of_two[DNA[B<A][1]].
//for Processor 2
//Both processor can execute in parallel
}
This is definitely much faster code than simple swap since it can be executed in parallel. In normal Swap function we need to execute three steps sequentially.

B. Designing a direct mapping sorter for three integers

There are six combinations possible for 3 elements.
abc, acb, bac, bca, cab, cba.
For these combinations patterns are 110,100,010,101,011,001 respectively.
For these patterns the correct sequenced can be written in DNA table.

```
Int DNA[6][3]=
{2,1,0,1,0,2,1,2,0,0,2,1,2,0,1,0,1,2}.
Int three[3]=copy_of_three[3]={x,y,z}.
//x,y and z are integers
DMS_3(int three[0],int three[1],int three[2])
{
Index=(A>B)*4+(C>B)*2+(A>C).
//mapping index
three[1]=copy_of_three[DNA[index][1]].
//for processor 1
three[2]=copy_of_three[DNA[index][2]].
//for processor 2
three[3]=copy_of_three[DNA[index][3]].
//for processor 3
//all three processors can execute in
//Parallel
}
```

C. Designing a direct mapping sorter for 4 integers

Four elements can be sorted using 24 different patterns. To represent a pattern of four elements we need 6 bits, so we need an array containing 2^6 =64 patterns.

```
int DNA[64][4]=
        {
        3,2,1,0,2,3,1,0,0,0,0,0,2,1,3,0,
        3,1,2,0,0,0,0,0,1,3,2,0,1,2,3,0,
        0,0,0,0,0,0,0,0,0,0,0,0,2,1,0,3,
        0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,3,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        3,1,0,2,0,0,0,0,1,3,0,2,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,1,0,3,2,1,0,2,3,
        3,2,0,1,2,3,0,1,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,2,0,3,1,0,0,0,0,2,0,1,3,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        3,0,2,1,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,3,2,1,0,2,3,1,0,0,0,0,0,2,1,3,
        0,3,1,2,0,0,0,0,0,1,3,2,0,1,2,3}.
int four[4]=copy_of_four[4]={a,b,c,d}.
//a,b,c and d are integers
DMS_4(int four[0],int four[1],int four[2],int four[3])
{
Index=(four[1]>four[0])*32+(four[2]>four[0])*16+(four[3]>four[0])*8+(four[2]>four[1])*4+(four[3]>four[1])*2+(four[3]>four[2]).
//the mapping index
four[0]=copy_of_four[DNA[index][0]].
//for processor 1
four[1]=copy_of_four[DNA[index][1]].
//for processor 2
four[2]=copy_of_four[DNA[index][2]].
//for processor 3
four[3]=copy_of_four[DNA[index][4]].
//for processor 3
//all four processors can execute in parallel
}
```

All the three DMS discussed here can be implemented easily as subroutines and can be called in any parallel algorithm to sort a smaller grain of sequence. We can increase the parallelization using DMS. The mapping technique can be made even faster using bitwise operations.

III. WORKING OF A DMS

A direct mapping sorter works as shown in Figure 1. A copy of an input array is kept in shared memory of all the processors. A copy of DNA pool is kept in each processor's local memory. Processor can map sorted sequence of its part directly from the copy of original array to the original array and overwrite the sorted values in parallel with other processors. The copy of the array then can be discarded and latest copy of this array can be generated. Since there is no swapping required in direct mapping sorter and all the execution is kept parallel as much as possible, the number of processors required will increase considerably.

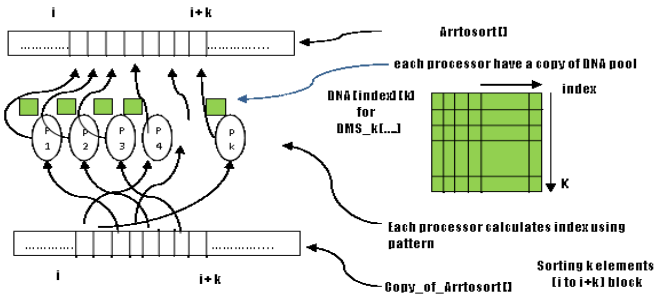

Figure 1. Arrangement of a direct mapping sorter

IV. IMPLEMENTING DIRECT MAPPING SORTERS

A. Bitonic sorting

Bitonic sorting is a sorting network algorithm developed by Batcher [1].Bitonic sorting uses the property of being bitonic of any sequence. Bitonic merge sort [2]is based on repeatedly merging two bitonic sequences, to form a larger bitonic sequence. This is a very classic algorithm and is studied thoroughly. Bitonic algorithm is implemented on several machines and architectures. We can reduce the number of steps in this algorithm considerably using DMS_4 (direct mapping sorter to sort 4 elements in single step). For example the implementation of bitonic sorts on mesh networks. To sort 16 integers we need a mesh having 16 processors. Each processer contains a value for its index.
Let the index arrangement of processors of a mesh network be in the following fashion.

p0	p1	p2	p3
p4	p5	p6	p7
p8	p9	p10	p11
p12	p13	p14	p15

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

The steps are as following in the sorting process using DMS_4.

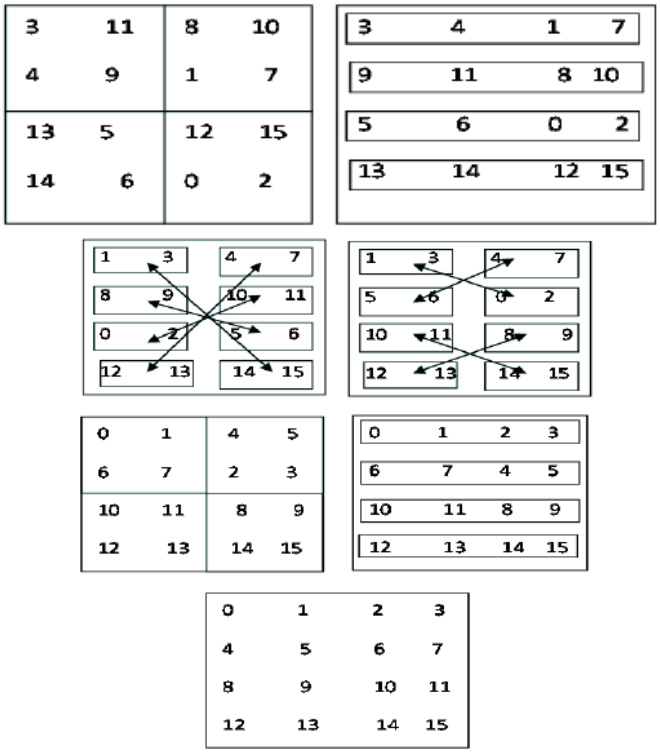

Figure 2. Sorting 16 integers in 7 steps using DMS_4

Using pure bitonic sort we need 10 steps to sort any 16 element sequence but using DMS_4 we can sort it just 7 steps. There can be several possible index arrangements of a mesh network. We are using left to right snake like arrangement of a mesh network. The steps are particular for this arrangement but we can easily adjust these steps according to type of m

TABLE 1.Comparison of bitonic sort with and without DMS_4

Input(number of input elements)	Steps Required(With DMS_4)	Steps required(Without DMS_4)
1	1	1
2	1	1
4	1	3
8	3	6
16	7	10
32	12	15
64	18	21
128	25	28

Figure 2 shows the steps involved in sorting 16 integers in 7 steps using bitonic sorting with the help of DMS_4. The time complexity [4] of bitonic sort algorithm is $O(\log^2 n)$. It requires log n(log n+1)/2 steps to sort any arbitrary sequence. So the bitonic sequence takes 10 steps to sort the given sequence and it only requires 7 steps.After using DMS_4 the sorting will always require 3 steps lesser than normal bitonic sort. The graph shows the difference. The graph is plotted on logarithmic scale.

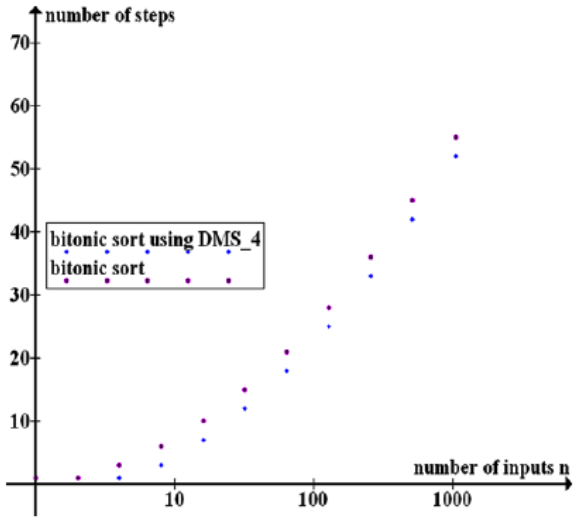

Figure 3. Comparison of Bitonic sort with and without DMS 4

B. Shear Sort

Shear sort [3] is used to sort k^2 elements where we uses k× k mesh. There are two phases in shear sort as following.

Phase A:

Do k times

Sort even numbered lows right to left and even numbered rows left to right.

Sort columns top to bottom.

Phase B: Sort rows in alternating sequence as above.

To sort any column or row of size k we need k steps or $n^{1/2}$ steps if $n=k^2$, where n is the number of input elements. The overall complexity of shear sort is $O(n^{1/2} \log n)$.To sort 16 element mesh we need 8 steps in shear sort.

Shear sort provides extremely great results but on the cost of large number of processors required, but a cost optimal result can be found for this arrangement. Figure 5 elaborates the steps involved in shear sort for 4×4 mesh. The arrangement of nodes is same like bitonic sort. This is worst case arrangement when the entire array is arranged in decreasing order, hence number of rearrangements will be maximum.

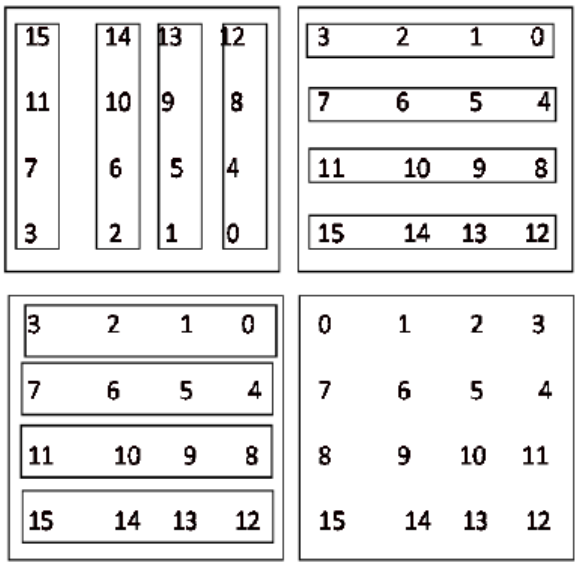

Figure 4. Three steps required to sort 16 integers using DMS_4 with shear sorting.

We need only three steps to sort 16 elements using DMS_4 with shear sort.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

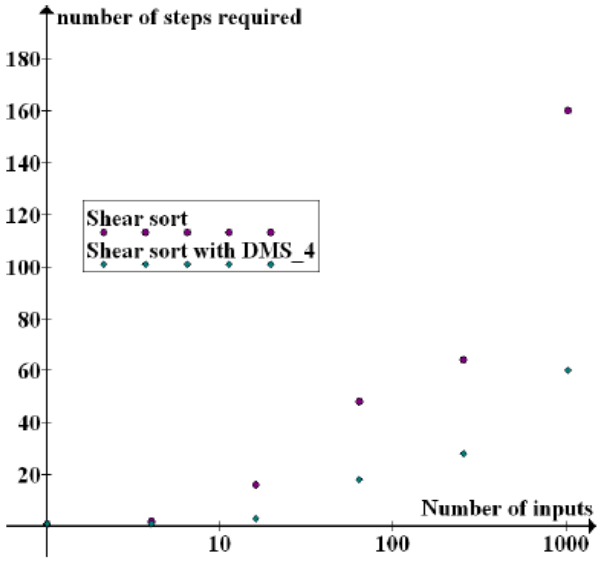

Figure 5. Comparison of shear sort with and without DMS_4

We can analyze the results more deeply using following table.

TABLE 2. Comparison of shear sort with and without DMS_4

Input(number of input elements)	Steps Required(Without DMS_4)	Steps required(With DMS_4)
1	1	1
4	2	1
16	16	3
64	48	18
256	64	28
1024	160	60

V. ANALYSING WITH RESULTS OF DMS ON GIVEN EXAMPLES

Parallel sorting like bitonic sort have been more recent [5] subject of research. Bitonic sorting has two major phases in itself. In first phase any arbitrary sequence is converted into bitonic sequence and in second phase bitonic sequence is sorted in log n steps. From the figure 3 we can see that three steps are reduced for each number of inputs. Although it is not much significant value for large sequences but produce far better results for smaller sequences.

Shear sort is a two dimensional sort. According to figure 5 and Table 1, the great impact of DMS_4 is evident. Even for large inputs we can produce very effective results with the use of DMS_4. Since DMS_4 can sort four elements in single step for a small problem like 4*4 mesh we can get results in just few(three) steps as in figure 4.

Effectiveness of DMS depends on the granularity of problem and distribution of algorithm.

We created DMS up to DMS_4 which sorts 4 elements in single step using 4 processors. We can create larger DMS to solve larger grains of sequence to sort in a single step. Larger will be the size of DMS it will become easier to get to the output in lesser steps.

In Terms of cost optimality the DMS implementation may prove weaker since the number of processor required gets equal to the number of input elements and cost may increase considerably, but in terms of time complexity DMS may perform outstanding. In our examples of bitonic sort we can easily understand that the number of processors required get double (for a simple Bitonic sort we need only p=n/2 processors) and only 3 steps in every respective sorting are reduced. For bitonic sort the results in terms of cost are not that much impressive.

On the other hand for shear sort the results are encouraging even in the terms of cost optimality. Even the number of processors gets double, but the numbers of steps are reduced by 1/3 of original results. DMS can prove very effective in multicore [6] and Reconfigurable [7] architectures.

VI. CONCLUSION

The achievement of DMS is to reducing the number of steps in sorting using a parallel algorithm. Well designed and larger size DMS may prove even faster, but to design a larger DMS is difficult job each and every time as the size of input increases because the size of DNA pool may become unmanageable. In addition to identify correct sequence to return unique index is difficult.

The space complexity increases as each processor will keep a copy of DNA pool and a copy of Input sequence is always kept in advance for future use. Due to technological advancements and cheaper hardware the processor and memory requirements are not of that much significance.

REFERENCES

[1] Kenneth E. Batcher. Sorting Networks and their Applications.volume 32 of AFIPS '68 (Spring), pages 307–314, New York, NY, USA, 1967. ACM.

[2] Z. Hong and R. Sedgewick. Notes on merging networks. In Proc. 14th ACM Symp. on Theory of Computing(STOC).

[3] D. E. Knuth. The Art of Computer Programming, volume 3. Addison Wesley, Reading Massachusetts, 1973.

[4] M. S. Paterson. Improved sorting networks with O(logn) depth. Algorithmica,

[5] D.A. Bader, D.R. Helman, and J. J´aJ´a. Practical Parallel Algorithms for Personalized Communication an d Integer Sorting. ACM Journal of Experimental Algorithmics, 1(3), 1996.

[6] Hagen Peters, Ole Schulz-Hildebrandt, Norbert Luttenberger "A novel sorting algorithm for many-core architectures based on adaptive bitonic sort".

[7] J. Angermeier, E. Sibirko, R. Wanka, and J. Teich Bitonic Sorting on Dynamically Reconfigurable Architectures Technical Report CS-2011-01,December 2011

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.