CS231P HW6 Report

Name : Subash Chandra Kotha
UCI ID: sckotha

Name : Parth Shah
UCI ID: parths4

Shear Sort

Shear sort is a two-dimensional mesh sorting network. Each iteration of the algorithm contains two phases:

1. Row sorting in a snake like fashion
2. Column sorting

After all the iterations are completed a final row sort is performed to get elements in order

In the row sorting phase, each row is sorted so that odd numbered rows have the largest number to right, and even numbered rows have the largest number to the left.

In the column sorting phase, each column is sorted so that the smallest numbers appear at the top of columns.

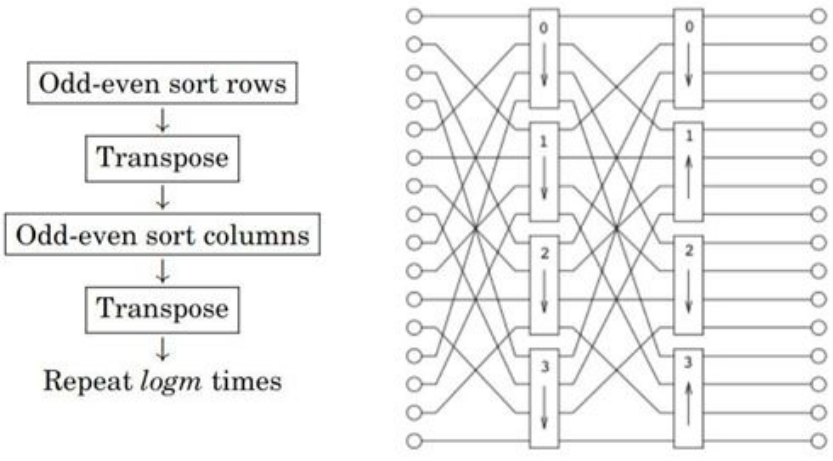

Fig 1: Shear Sort Algorithm

Procedure

We have a N element sequence and a crossbar inter connected cluster of K computers. Assuming each processor has only one core and employs a quick sort algorithm. The following are the steps:

Let us divided the N element sequence into **(M * M)** matrix where M equals square root of **N**. Here we assume M is divisible by **K.** The matrix will be stored in row-major form. Each row contains **M** elements.

$$[\, X_1 \; X_2 \; X_3 \; \ldots\ldots \; X_n \,]$$

$$[\, [\, X_1 \; X_2 \; X_3 \; \ldots\ldots\ldots\ldots \; X_M \,]$$

$$[\, X_{1+M} \; X_{2+M} \; X_{3+M} \; \ldots\ldots\ldots\ldots \; X_{2M} \,]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$[\, X_{(n-k)} \; X_{(n-k+1)} \; X_{(n-k+2)} \; \ldots \; X_n \,] \,]$$

Phase 1

Since there are M rows and K processors each processor picks **M/K** rows and rows are sorted in snake like manner i.e. odd rows(processors) are sorted from left to right and even rows(processors) are sorted from right to left.

Now we perform the transpose of the matrix using a crossbar network. Assuming we have a X-bar switch we will use a parallel transfer procedure used for taking transpose as in FFT.

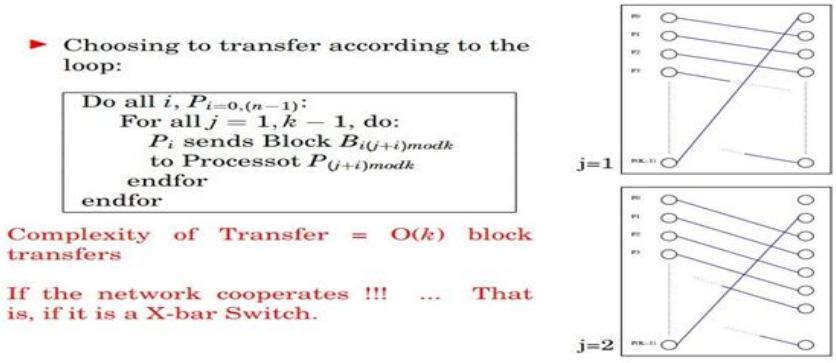

Fig 2: Crossbar network

We will be using the algorithm present in the above picture for crossbar communication.

Phase 2

We now have an **(M * M)** matrix which is the transpose of the original matrix. Similar to phase 1 each processor picks **M/K** rows but in this case we sort all the rows in ascending manner i.e. the largest element goes to the right. After sorting we perform the transpose of the matrix.

Both these phases contribute to one iteration. In each phase, we have 2 transpositions and 2 sorting. Since there are M rows after **log(M)** iterations the sequence is sorted.

Complexity

We are employing quick sort, and each row contains M elements, sorting each row takes M(log M) time. But each processor handles K rows, hence sorting in phase 1 takes M/K * M * log(M)

Similarly sorting in phase 2 takes M/K * M * log (M) time.

Time taken for each iteration = O (2 * M/K * M * log(M))

Final time complexity = Time for each iteration * total number of iterations

$$T(n) = O(2 * M/K * M * \log(M) * \log(M)) = \mathbf{2\ O(\ N/K\ (\log\ (M)^2)\ )}$$

Each processor stores M/K rows each of size M. Hence, memory required at each computer = O(M * M/K) = **O(N/K)**

Each communication for a single transpose consist of transfer of M/K * M/K blocks. Hence the number of crossbar communication cycles needed O(K). But we have 2 transpose operations in a single iteration. Therefore, total number of cross bar communications will be **2 * O(K) * log(M)**

Bitonic Sort:

Bitonic sort is a comparison based sorting algorithm that can run in parallel.

Bitonic Sequence - A bitonic sequence is a sequence of numbers which first increases then decrease or first decreases then increases. A sequence produced by rotation or cyclic shift of such a sequence is also bitonic.

A sequence of numbers that monotonically increases and then decreases or first decreases then increases.

A sequence $S = (S_1, S_2,...S_n)$ of n numbers is said to be bitonic if -

$S_1 \leq S_2 \leq . . . \leq S_k \geq . . . \geq S_n$, for some k, $1 < k < n$

OR

$S_1 \geq S_2 \geq . . . \geq S_k \leq . . . \leq S_n$, for some k, $1 < k < n$

A bitonic sequence is divided between its two halves, and the n^{th} element in each part is compared with each other. If they are out of order, they are swapped. Applying this procedure repeatedly onto the smaller lists, the result is a sorted sequence in non-decreasing order.

For a bitonic sequence of n elements, following the above procedure would result in n bitonic sequences sorted with respect to each other.

Each stage of dividing a bitonic sequence and producing two half bitonic sequences takes $\log_2(n)$ time.

However, for bitonic sort we require a bitonic sequence. If the input sequence is not bitonic, we need to convert it to a bitonic sequence. This can be done by converting consecutive pairs of elements and producing a bitonic sequence of length 4, where each pair is in ascending and descending order alternatively. This step can be repeated until we get a bitonic sequence of length 8.

Parallel Algorithm:

The above described algorithm is for doing the conversions in a serial manner, the parallel version of this algorithm is however much more efficient.

In the parallel algorithm, each computer can simultaneously perform the comparisons and handle the exchanges.

Here we are given a crossbar inter-connected network of k computers. We can divide the input sequence of n integers into k blocks, each consisting of n/k elements.

Hence each computer will be allocated memory to store this block of elements.

The parallel algorithm is quite similar to the sequential algorithm, the difference here is that we use the inter-connection network to transfer the numbers as needed by the processor.

Let us assume that we have a bitonic sequence of 8 elements and 4 processors:-

Stage 1-

- We divide the 8 elements in 2 halves of 4 increasing and 4 decreasing elements.
- Now the input to i^{th} processor is the i^{th} element in each half.
- Output is 2 bitonic sequences of length 4.

Stage 2-

- Now we again the elements in each half into two other halves and we have 4 blocks of 2 numbers
- The first element of two consecutive blocks are given as input to one processor and the second elements are given as inputs to the next processor.
- Output is 4 bitonic sequences of length 2.

We keep on doing this till we get 8 sequences of length 1. This sequence is also the sorted order.

Complexity:

It takes $\log_2(n)$ time to get two bitonic sequences of length n/2 from a bitonic sequence of length n.

Hence we can compute the time as:

$T(n) = \log_2 n * (\log_2 n + 1)/2$

Hence, Time complexity is $\mathbf{O(\log_2(n))^2}$.

Memory Requirement:

Each computer stores n/k elements as mentioned above. Hence, space complexity is **O(n/k).**

Number of crossbar communication cycles per phase is **k.**