Ceng478 HW1 Report

Oğuzhan Taştan
Department of Computer Engineering
METU
Ankara, Turkey

***Abstract*— In this paper, some comparisons have been made between different sizes of data, between different numbers of processors and between two algorithms in terms of the effect on parallelism by accomplishing the task of finding the minimum element of the given array.**

***Keywords*—Parallel Computing, MPI**

I. INTRODUCTION

In the experiments three different sizes of array, 160.000, 1.600.000, 160.000.000 and five different numbers of processors, 1, 2, 4, 8, 16 and two different algorithms; called send receive and reduce algorithm; have been used.

The send receive algorithm initially divides the processes into pairs and one process from each pair sends its minimum to the other, and the other compares the received one with its own minimum and decides which is the minimum. The same procedure is applied repeatedly in several stages until the global minimum is obtained in one process.

In reduce algorithm, each processor finds its minimum and sends it to the master processor. The master processor finds the minimum of the minimums which is the global minimum.

II. PERFORMANCE OF SEND-REDUCE ALGORITHM

Since the all processors run in parallel, the cost of finding the local minimums in each processors is n/p where n is the length of the input array and p is the number of processors. After that, comparing the local minimum with the received minimum takes lg(p).

In addition to these costs, the parallelism brings in communication cost. The total cost of the algorithm is n/p+ log(p) + C where C is the communication costs. If an inefficient method that simply compares all the elements in a process sequentially is used, the complexity is O(n). Thus, the send receive algorithm reduces the cost significantly when the optimum number of processors are used. The following paragraphs demonstrate the experimental results of the algorithm.

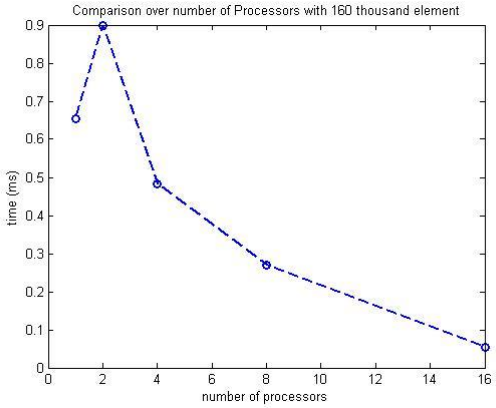

Figure 1: Time vs. Number of Processors with the array which contains 160 thousands elements.

With small number of elements, 160 000, the send receive algorithm which run on only one processor takes 0.663 milliseconds. When it runs on two processors with the same node, the expectation is that it will take less time, however in practice the time it takes have increased to 0.906 as can be seen in the figure 1. The reason behind this behavior is that the size of the data is not large enough to parallelize with two processors. In other words, the cost of communications between processors is larger than the cost of computation which the parallelism reduces. When the algorithm runs on four processors, the communication cost is less than the reduction of computation cost the parallelism brings in, therefore the efficiency is achieved.

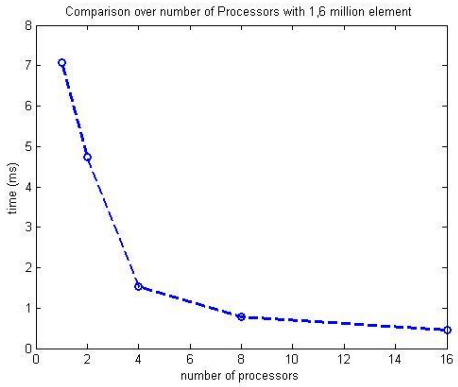

Figure 2: Time vs. Number of Processors with the array which contains 1.6 million elements.

In contrast to small amount of data, a larger number of data, 1.6 million, is appropriate to be parallelized. As shown in the figure 2, the execution time of the algorithm which runs with the larger data decreases as the number of processors increases both in one node and two nodes. The similar results are obtained when the data size is 160 million which can be seen in the figure 3.

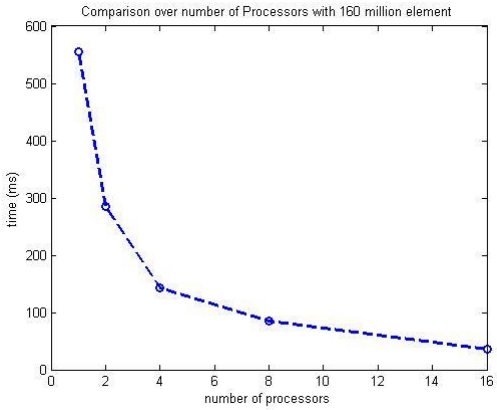

Figure 3: Time vs. Number of Processors with the array which contains 160 million elements.

In addition to all these comparisons, another eyeful point is that as the number of processors increases, the slope of line decreases meaning that the parallelism decreases. In other words, if the number of processors doubled then the execution time is not halved, instead it is more than half. The reason of that behavior is that the sequential part of the algorithm cannot be parallelized.

III. PERFORMANCE OF REDUCE ALGORITHM

The reduce algorithm gives very similar results to the send receive algorithm's results. Although the MPI_Reduce function is implemented efficiently, because of few number of communications it cannot make much difference over send receive algorithm. The figure 4 shows the comparison of two algorithms with 160 million elements.

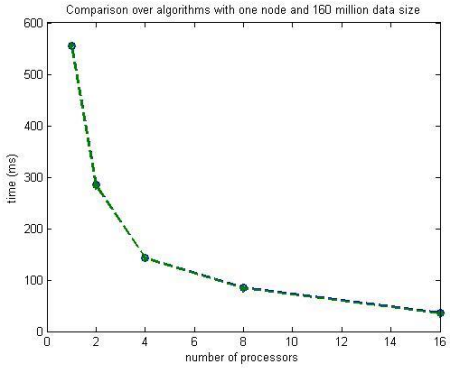

Figure 4: send receive vs. reduce. The blue dashed line represents the performance of send receive algorithm while the green dashed line represents that of reduce algorithm.

With this relatively large data we cannot differentiate the performance of algorithms. With relatively small amount of data, 160 thousands, the reduce algorithm takes very slightly less time as shown in the figure 5.

Since the send receive algorithm requires send and receive operations between all pairs of processors and in the reduce algorithm the receive operation is done by only the master processor, with very large data and large number of processors the reduce algorithm runs more efficiently. This means that the reduce algorithm has more scalability than the send receive algorithm.

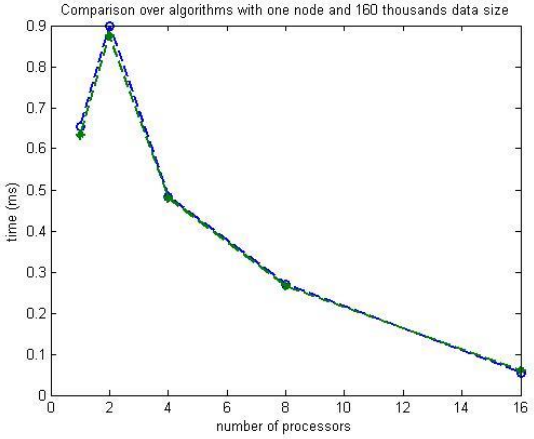

Figure 5: send receive vs. reduce. The blue dashed line represents the performance of send receive algorithm while the green dashed line represents that of reduce algorithm.

IV. COMPARISON OVER DATA SIZE

As the data size increases, the expectation is that the execution time increases with the same proportion of increase in data size. However, in practice, increase in data size result in efficiency increase. For example, with four processor and 160 000 elements the execution time is 0.185 milliseconds while with 1 600 000 and the same number of processors the execution time is 1.344 milliseconds as shown in figure 6.

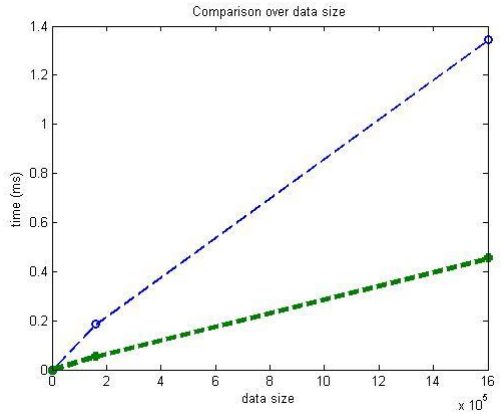

Figure 6: send receive vs. reduce. The blue dashed line represents the performance with four processors while the green dashed line represents that of sixteen processors.

As the data size goes ten times, the execution time goes less than the ten times. This is caused by the fact that the proportion of the communication cost to total cost decreases as the data size increases since communication cost is constant in the algorithms.

In addition, it can be inferred from figure 6 that the efficiency difference between different data sizes have lower values while the algorithms running on sixteen processors than running on four processors because sixteen processors need more communication than four processors do.