

Parallel Architectures and Systems

Programming Assignment for Fastest Sorting

Dr. Edwin Sha

Due Date: **1pm, April 15, 2010**

PUT DOWN YOUR EMAIL ADDRESS IN YOUR REPORT.

1. (100 points) Try to write the **fastest** Sorting program based on MPI. Each team submit one copy of your programs and report.

You must implement Radix Sorting, Sample Sorting and **Randomized** Sample Sorting algorithms. You may try to use collective communication or `MPI_Isend()` and `MPI_Irecv()` whenever possible to overlap computation and communication.

The Randomized Sample Sorting is easy in the splitter selection step. Let k be the number of bucket. In the splitter selection step, you will “randomly” pick sk samples from input data, sort them and pick $k - 1$ evenly spaced elements as final splitters for later bucket sorting. That’s it. This s is called over-sampling ratio. People have proved that with $s = 64$, the probability that a bucket size $\geq 2.5n/k$ is extremely small $< 10^{-6}$.

You may not see much speed-up from our clusters for solving sorting problem because sorting is not really that much computation intensive as matrix multiplications. But we will have a contest for the shortest running time for all the students’ programs and the extra points will be given to those who can run the fastest. So the grade will be based on the program’s running time and the clearness and completeness of your report. I would suggest you to use Optimization Option “-O3” in compiling. In order to make it fair, you should follow the following rules.

- There are four DATA files for you to test, DATA1, DATA2, DATA3 and DATA4 in our course directory:

`/net/core/export/home/cs/001/e/edsha/parallel/2010S`

Go ahead to copy them.

- `Starttime()` is called AFTER the Process 0 has read the data from DATA file, and before any communication is performed.
- `Endtime()` is called after the Process 0 has received all the sorted data and before any results are printed out.
- Implement Radix Sorting algorithm. It is okay if it is not fast. It is interesting to compare the timing performance with sample sorting algorithms.

Process 0 should print out the middle number, the $n/2$ -th element, in the sorted sequence. Note that if array $A[0, \dots, 9]$ (starting from index 0) store the final sorted 10 elements (when $n = 10$), the $(n/2)$ -th element is $A[4]$. If you print out a wrong one, sorry, no points.

- You should have already implemented the Sample Sorting algorithm in your previous homework. But you can revise it to make it faster.

Process 0 should print out how many items were put in each buckets, and the middle number, the $n/2$ -th element, in the whole sorted sequence.

- Implement Randomized sorting algorithms, and print out each bucket sizes. Try different s values such as 32, 64, 128.

Process 0 should print out how many items were put in each buckets, and the middle number, the $n/2$ -th element, in the whole sorted sequence.

- Which algorithm is the fastest one. Report the running times of each one.

The contest will be based on the running time obtained from your fastest program for one case: DATA4 (5M numbers) on 2-4 workstations on our paraX Linux cluster. You can also try to run it on 6 or 8 workstations. The running time reported should be the average of the two fastest runs you did. If I were you, I may run my programs in a time period with few people using the cluster and may try to avoid para1 that everyone is using.

- You should give me your codes of three algorithms and performance data, and discuss the results from your different styles of MPI programs. I will be very much interested in knowing all the experiments you have done. Try to impress me and the whole class.
- Possible Grading (subject to change): Radix Sorting (30 points), Sample Sorting (30 points), Randomized Sample Sorting (20 points), Efficiency and report (20 points).
- Be honest; otherwise, I will be very upset.

You should submit a HARD COPY of your programs, AND an ELECTRONIC COPY of your programs before the due date. How to submit your program electronically:

- (a) Go to the directory that contains all your source code, makefile, executables, README, required files, etc. You must explain how to run your programs clearly in the README file and which host computers you used to run your programs. **You must write down your name and email address in the README file.**
- (b) Make sure the directory contains ALL the files that are able to be compiled and executed for the assignment. The TA will use your Makefile to compile your program. In your README file, you should explain how to compile and run your programs. You should remove any redundant files so the tar file will not be too large! Let me say again. Please remove any unnecessary files before submission.
- (c) In that directory, issue the command `~ edsha/submit/handin.ass3` on “apache.utdallas.edu” or one of the systems in Solarium Lab. Do not use Linux machines to submit your programs because my submit executable program will not work for Linux Intel machines. We will look at the time of the submission file to know if you submit your programs before or after the due date. This command must be issued in the directory that contains the files for the assignment.
- (d) Before the due date, you can resubmit your files without any problem.