

Programming Project 2 Parallel Implementation of the Prime Number Sieve

The goal of this project is to design and implement a parallel algorithm to generate all the prime numbers between 1 and N on the MST Cluster. You will use FORTRAN/C/C++ Language with Message-Passing Interface (MPI) for your parallel code. Observe how the **speedup** and **efficiency** change by varying N and P (no. of processors). A reasonable range for N is 1M to 10 Billion. For P, the range will be 1 to 16. Your report will include:

I. (5 pts)

Electronic copy of your program. Your program must: (i) contain a program overview/summary, your id/name etc. in the header, (ii) execute correctly, (iii) be well documented (Must include comments before every major statement, function/subroutine calls, and every MPI call explaining what the call is for. Must include specs for every function/subroutine)

II. (5 pts)

Read the file “How to submit projects?” on the class website and follow all the instructions in there. You must submit all of the files requested in the document.

III. (25 pts) Report

- (i) The total number of prime numbers between N=1 and N=1 Billion
- (ii) The first 20 prime numbers starting at 1,000,000 printed in four rows (5 numbers per row).

IV. (10 pts) Explain how the data and task partitioning is done.

V. (15 pts) As you know, the algorithm runs faster if you implement the last optimization explained in class. i.e. if current prime= p , the first number to strike out is p^2 ; then you strike out $p^2 + p$, $p^2 + 2p$, etc. Compare the optimized and non-optimized versions of the algorithm and report how much faster the code runs for P=1, and N= 32M, 64M, and 128M. You need to report 3 speedup figures. (note that in both cases, primes go upto \sqrt{N})

VI. (20 pts) Provide two tables:

- (i) A table showing *speedup* for N=4 Billion and P= 1, 2, 4, 8, and 16.
- (ii) A table showing *efficiency* for N=4 Billion and P= 1, 2, 4, 8, and 16.

Try to explain how speedup and efficiency change due to task granularity, the load balancing, and the communication overhead. Talk about the relationship between some of the following parameters: N, P, speedup, task granularity, load balancing, and the communication overhead. Use phrases like “... it increases/decreases proportional to ...” , “...gets larger/smaller...”, “.. the increase in X is more than the increase in Y for such and such reason ...”, etc.

VII. (20 pts) Provide two tables:

- (i) A table showing *speedup* for P=8 and N= varying between 100M and 4 Billion.
- (ii) A table showing *efficiency* for P=8 and N= varying between 100M and 4 Billion.

Explain how speedup and efficiency change due to task granularity, the load balancing, and the communication overhead.

(turn over)

IMPORTANT NOTES:

1) Time only the loop where primes are received from the master, and non-primes were deleted from the array. Do not time the portion where the prime numbers are collected from the slaves by the master. Master will start and stop the clock as shown below:

```
MASTER

read(input)
...
Start-Time
...
while(!done){
    Broadcast next prime
    perform markings
}

MPI_Barrier(comm)    /* to make sure all of the processes completed */
Stop-Time
...
Receive results
```

2) You'll be better off creating your arrays as

```
char *your_array; your_array = (char*)malloc(SOME_NUMBER * sizeof(char));
```

instead of

```
char your_array[ SOME_NUMBER ];
```

The first way is better because the space comes out of free space and isn't subject to the same restrictions as the second way which comes out of the stack space (general guideline: if it's subject to scoping rules then it came out of stack space). The second way works fine for small arrays, but not for large ones. Also don't forget to `#include <stdlib.h>` to use the *malloc* function.

3) To be more accurate, take the average of 2-3 runs for the same program and report it as one timing result.

BONUS:

Each program that satisfy one of the following will receive bonus points as described in the syllabus.

1) The program with the best running time for P=16 and N= 10 Billion

2) The program that finds the highest prime number

Therefore, do not forget to report your running time for P=16 and N=10 Billion as well as the highest prime number that your parallel program is able to find.