



LAB EXPERIMENT 4

PARALLEL AND DISTRIBUTED COMPUTING

Exercise - 4 :

1. The gap between consecutive prime numbers 2 and 3 is only one , while the gap between consecutive primes 7 and 11 is 4. Write a parallel program using MPI to determine, for all integers less than 1,00,000, the largest gap between a pair of consecutive prime numbers.

CODE:

```
/*
```

By jack_1806

```
*/
```

```
#include "mpi.h"
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
#define LIMIT    2500000    /* Increase this to find more primes */
```

APURV TODKAR
16BCB0048

```
#define FIRST    0        /* Rank of first task */
```

```
int isprime(int n) {
```

```
    int i,squareroot;
```

```
    if (n>10) {
```

```
        squareroot = (int)sqrt(n);
```

```
        for (i=3; i<=squareroot; i=i+2)
```

```
            if ((n%i)==0)
```

```
                return 0;
```

```
        return 1;
```

```
    }
```

```
/* Assume first four primes are counted elsewhere. Forget everything else */
```

```
else
```

```
    return 0;
```

```
}
```

```
int main (int argc, char *argv[])
```

```
{
```

```
    int  ntasks,          /* total number of tasks in partition */
```

```
        rank,            /* task identifier */
```

```
        n,               /* loop variable */
```

```
        pc,             /* prime counter */
```

```
        pcsum,          /* number of primes found by all tasks */
```

```
        foundone,       /* most recent prime found */
```

```
        maxprime,       /* largest prime found */
```

```
        mystart,        /* where to start calculating */
```

```
        stride;         /* calculate every nth number */
```

```
double start_time,end_time;
```

```
MPI_Init(&argc,&argv);
```

APURV TODKAR
16BCB0048

```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&ntasks);
if (((ntasks%2) !=0) || ((LIMIT%ntasks) !=0)) {
    printf("Sorry - this exercise requires an even number of tasks.\n");
    printf("evenly divisible into %d. Try 4 or 8.\n",LIMIT);
    MPI_Finalize();
    exit(0);
}

start_time = MPI_Wtime(); /* Initialize start time */
mystart = (rank*2)+1;     /* Find my starting point - must be odd number */
stride = ntasks*2;       /* Determine stride, skipping even numbers */
pc=0;                    /* Initialize prime counter */
foundone = 0;            /* Initialize */

/***** task with rank 0 does this part *****/
if (rank == FIRST) {
    printf("Using %d tasks to scan %d numbers\n",ntasks,LIMIT);
    pc = 4;               /* Assume first four primes are counted here */
    for (n=mystart; n<=LIMIT; n=n+stride) {
        if (isprime(n)) {
            pc++;
            foundone = n;
            /***** Optional: print each prime as it is found
            printf("%d\n",foundone);
            *****/
        }
    }
}

MPI_Reduce(&pc,&pcsum,1,MPI_INT,MPI_SUM,FIRST,MPI_COMM_WORLD);

MPI_Reduce(&foundone,&maxprime,1,MPI_INT,MPI_MAX,FIRST,MPI_COMM_WORLD);

end_time=MPI_Wtime();
```

APURV TODKAR
16BCB0048

```
printf("Done. Largest prime is %d Total primes %d\n",maxprime,pcsum);  
printf("Wallclock time elapsed: %.2lf seconds\n",end_time-start_time);  
}
```

```
/****** all other tasks do this part *****/
```

```
if (rank > FIRST) {  
    for (n=mystart; n<=LIMIT; n=n+stride) {  
        if (isprime(n)) {  
            pc++;  
            foundone = n;  
            /***** Optional: print each prime as it is found  
            printf("%d\n",foundone);  
            *****/  
        }  
    }  
    MPI_Reduce(&pc,&pcsum,1,MPI_INT,MPI_SUM,FIRST,MPI_COMM_WORLD);  
  
    MPI_Reduce(&foundone,&maxprime,1,MPI_INT,MPI_MAX,FIRST,MPI_COMM_WORLD);  
}  
  
MPI_Finalize();  
}
```

APURV TODKAR

16BCB0048

```
Applications ▾ Places ▾ Terminal ▾ Fri Oct 27, 22:26:58 1
root@jackie: ~/pdclPR/u/4
File Edit View Search Terminal Help
root@jackie:~/pdclPR/u/4# mpirun -np 4 ~/pdclPR/u/4/1
Using 4 tasks to scan 2500000 numbers
Done. Largest prime is 2499997 Total primes 183072
Wallclock time elapsed: 0.24 seconds
root@jackie:~/pdclPR/u/4#
```

bash

bash

.bashrc if it exists

5. What if I can't modify my PATH and/or LD_LIBRARY_PATH?

There are some situations where you cannot modify the PATH or LD_LIBRARY_PATH -- e.g., some ISV application prefer to hide all parallelism from the user, and therefore do not want to make the user modify their shell startup files. Another case is where you want a single user to be able to launch multiple MPI jobs simultaneously, each with a different MPI implementation. Hence, setting shell startup files to point to one MPI implementation would be problematic.

In such cases, you have two options:

1. Use mpirun's --prefix command line option (described below).
2. Modify the wrapper compilers to include directives to include run-time search locations for the Open MPI libraries ([see this FAQ entry](#)).

mpirun's --prefix command line option takes as an argument the top-level directory where Open MPI was installed. While relative directory names are possible, they can become ambiguous depending on the job launcher used; using absolute directory names are strongly recommended.

For example, say that Open MPI was installed into /opt/openmpi-3.0.0. You would use the --prefix option like this:

```
1 shell$ mpirun --prefix /opt/openmpi-3.0.0 -np 4 a.out
```

This will prefix the PATH and LD_LIBRARY_PATH on both the local and remote hosts with /opt/openmpi-3.0.0/bin and /opt/openmpi-3.0.0/lib, respectively. This is *usually* unnecessary when using resource managers to launch jobs (e.g., SLURM, Torque, etc.) because they tend to copy the entire local environment -- to include the PATH and LD_LIBRARY_PATH -- to remote nodes before execution. As such, if PATH and LD_LIBRARY_PATH are set properly on the local node, the resource manager will automatically propagate those values out to remote nodes. The --prefix option is therefore usually most useful in rsh or ssh-based environments (or similar).

Beginning with the 1.2 series, it is possible to make this the default behavior by passing to configure the flag --enable-mpirun-prefix-by-default. This will make mpirun behave exactly the same as "mpirun --prefix \$prefix ...", where \$prefix is the value given to --prefix in configure.

Finally, note that specifying the absolute pathname to mpirun is equivalent to using the --prefix argument. For example, the following is equivalent to the above command line that uses --prefix:

```
1 shell$ /opt/openmpi-3.0.0/bin/mpirun -np 4 a.out
```

6. How do I launch Open MPI parallel jobs?

Similar to many MPI implementations, Open MPI provides the commands mpirun and mptexec to launch MPI jobs. Several of the questions in this FAQ category deal with using these commands.

Note, however, that these commands are exactly identical. Specifically, they are symbolic links to a common back-end launcher command named orterun (Open MPI's run-time environment interaction layer is named the Open Run-Time Environment, or ORTE -- hence orterun).

APURV TODKAR
16BCB0048

2. Write a parallel program using MPI to sort a given set of integers using merge sort.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>

oid merge(int *, int *, int, int, int);
void mergeSort(int *, int *, int, int);

int main(int argc, char** argv) {

    /***** Create and populate the array *****/
    int n = atoi(argv[1]);
    int *original_array = malloc(n * sizeof(int));

    int c;
    srand(time(NULL));
    printf("This is the unsorted array: ");
    for(c = 0; c < n; c++) {

        original_array[c] = rand() % n;
        printf("%d ", original_array[c]);

    }

    printf("\n");
    printf("\n");

    /***** Initialize MPI *****/
```

APURV TODKAR
16BCB0048

```
int world_rank;
int world_size;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
MPI_Comm_size(MPI_COMM_WORLD, &world_size);

/***** Divide the array in equal-sized chunks *****/
int size = n/world_size;

/***** Send each subarray to each process *****/
int *sub_array = malloc(size * sizeof(int));
MPI_Scatter(original_array, size, MPI_INT, sub_array, size, MPI_INT, 0,
MPI_COMM_WORLD);

/***** Perform the mergesort on each process *****/
int *tmp_array = malloc(size * sizeof(int));
mergeSort(sub_array, tmp_array, 0, (size - 1));

/***** Gather the sorted subarrays into one *****/
int *sorted = NULL;
if(world_rank == 0) {

    sorted = malloc(n * sizeof(int));

}

MPI_Gather(sub_array, size, MPI_INT, sorted, size, MPI_INT, 0,
MPI_COMM_WORLD);

/***** Make the final mergeSort call *****/
if(world_rank == 0) {
```

APURV TODKAR
16BCB0048

```
int *other_array = malloc(n * sizeof(int));
mergeSort(sorted, other_array, 0, (n - 1));

/***** Display the sorted array *****/
printf("This is the sorted array: ");
for(c = 0; c < n; c++) {

    printf("%d ", sorted[c]);

}

printf("\n");
printf("\n");

/***** Clean up root *****/
free(sorted);
free(other_array);

}

/***** Clean up rest *****/
free(original_array);
free(sub_array);
free(tmp_array);

/***** Finalize MPI *****/
MPI_Barrier(MPI_COMM_WORLD);
MPI_Finalize();
return 0;
}

/***** Merge Function *****/
```


APURV TODKAR
16BCB0048

```
void merge(int *a, int *b, int l, int m, int r) {
```

```
    int h, i, j, k;
```

```
    h = l;
```

```
    i = l;
```

```
    j = m + 1;
```

```
    while((h <= m) && (j <= r)) {
```

```
        if(a[h] <= a[j]) {
```

```
            b[i] = a[h];
```

```
            h++;
```

```
        }
```

```
    else {
```

```
        b[i] = a[j];
```

```
        j++;
```

```
    }
```

```
    i++;
```

```
    }
```

```
if(m < h) {
```

```
    for(k = j; k <= r; k++) {
```

```
        b[i] = a[k];
```

APURV TODKAR
16BCB0048

```
                i++;

            }

        }

    else {

        for(k = h; k <= m; k++) {

            b[i] = a[k];
            i++;

        }

    }

    for(k = l; k <= r; k++) {

        a[k] = b[k];

    }

}

/***** Recursive Merge Function *****/
void mergeSort(int *a, int *b, int l, int r) {

    int m;

    if(l < r) {
```

APURV TODKAR
16BCB0048

```
m = (l + r)/2;
```

```
mergeSort(a, b, l, m);
```

```
mergeSort(a, b, (m + 1), r);
```

```
merge(a, b, l, m, r);
```

```
}
```

```
}
```

APURV TODKAR
16BCB0048

