LabX-8, BY 08 December 2021

How to do the exercises in this set

We are now increasing the usage of the bash scripts and pick up additional utility commands. Please consult the material in the Files section of canvas for this course, it may help with the syntax in the examples of the exercises. Use the C book of Kernighan & Ritchie (see Pages/Textbooks on canvas[1]). In canvas Files you have a summary of C language in slides file 3.1_Linux-C[2] . Also use the summary slides in file 3.1_Linux_Bash[3]

In this assignment we consider the MPI parallelization. This can be done traditionally by distributing the workload and also by increasing the computational volume as new resources (threads) are added.

On the delivery of your results: When you are ready with your program assignment, please create a directory with your name and copy the files there:

```
mkdir /trinity/home/LINUX_SUPERCOMPUTERS/yourname/   #it should exist already from exe1
mkdir /trinity/home/LINUX_SUPERCOMPUTERS/yourname/labx7
cp yourprogram /trinity/home/LINUX_SUPERCOMPUTERS/yourname/labx7/yourprogram
```

After you have finished copying submit also to canvas, so we can grade the arrivals. You do not have to send us e-mails when you complete this exercise.

Of course, if there are any problems or questions, please email me and/or Andrey:

"Igor Zacharov" I.Zacharov@skoltech.ru

"Andrey Kardashin" Andrey.Kardashin@skoltech.ru

[1] http://cslabcms.nju.edu.cn/problem_solving/images/c/cc/The_C_Programming_Language_(2nd_Edition_Ritchie_Kernighan).pdf

[2] https://skoltech.instructure.com/files/257760/download?download_frd=1

[3] https://skoltech.instructure.com/files/262799/download?download_frd=1

Further commands and Bash structure

Login to your sandbox account: ssh –p 2200 yourname@sandbox.zhores.net

tar

Some additional commands

a) The command to archive hierarchies: tar

from your home directory, try:

`tar –cvzf mydirs.tar.gz .` (note the dot – it stands for the current directory)

This will collect all files in all directories starting from local (specified by the dot "." on the command) the flags are: -c for create, this will create, -f with parameter the file name: mydirs.tar.gz
-z is for the compression option, -v is verbose. Verify the result with `ls –l`

Note, the sequence of flags is important, -f must have parameter, which is the file name.

To extract you have the –x flag. Try this: `cd /tmp; tar –xvzf ~/mydirs.tar.gz`

[Keep the tar-file of all your directories just in case you make mistake and remove something]

Now you just copied all of your files to /tmp (remove after the test). This can also be done in a single command:

```
export user=`whoami`; tar –cvf -  . | (mkdir /tmp/$user; cd /tmp/user;  tar –xf - )
```

Note the usage of – as file name with the tar command, it will stream the result to /from stdin/out. Thus –c will create a stream to stdout and –x will collect the stream from stdin.

b) Use the tar command to copy your directories across the machine boundary. Make a tar of a directory with a name mydir.tar.gz in your home direcoty, then on your laptop you can copy:

`scp –P 2200 yourname@sandbox.zhores.net:mydir.tar.gz .`

(the dot at the end stands for current directory)

`tar –xzf mydir.tar.gz` (if your laptop has tar installed or use an appropriate windows tool).

Writing C programs

1) Implement the following MPI program that computes pi from a random sequence:

```c
#include <mpi.h>

#define NPOINTS         8000000000      // number trials
#define GENSEED         1235791         // prime random seed

#define NANOSEC_in_SEC 1000000000LL
#define dsecond(tx,ty) ((double)((ty.tv_sec)-(tx.tv_sec))+(double)((ty.tv_nsec)-(tx.tv_nsec))/NANOSEC_in_SEC)


int main(int argc, char *argv[])
{
    long ncirc = 0;
    double pi, dpi;
    int numthrd;
    int mythrid;
    unsigned long long num_trials = NPOINTS;
    int rank, size;
    struct timespec tstart, tend;

    MPI_Init(&argc, &argv);
    MPI_Comm_size( MPI_COMM_WORLD, &numthrd);
    MPI_Comm_rank( MPI_COMM_WORLD, &mythrid);

    clock_gettime(CLOCK_REALTIME,&tstart);

//#pragma omp parallel default(none) firstprivate(num_trials) shared(ncirc)
{
    double x, y, t, dres1, dres2;
    struct drand48_data rbuf;
    long rseed = (mythrid+1) * GENSEED;
    unsigned long long local_ncirc = 0;
    unsigned long long i;

    srand48_r(rseed, &rbuf);

    for( i = 0; i < num_trials; i++ ) { // do not split work !

        drand48_r(&rbuf, &dres1);       // re-entrant random num gen
        drand48_r(&rbuf, &dres2);       // [0..1)

        x = 2.0 * dres1 - 1;            // place the circle around 0
        y = 2.0 * dres2 - 1;
        t = x*x + y*y;

        if( t <= 1.0 ) local_ncirc++;
    }
/*
    #pragma omp atomic
        ncirc += local_ncirc;
*/
    MPI_Reduce(&local_ncirc, &ncirc, 1, MPI_LONG_LONG_INT, MPI_SUM, 0, MPI_COMM_WORLD);
} // end parallel region

    clock_gettime(CLOCK_REALTIME,&tend);
    double tlaps = dsecond(tstart,tend);

    num_trials *= numthrd;              // all threads same work
    pi = 4.0 * (double) ncirc/ (double) num_trials;
    dpi = pi*sqrt(2.0/num_trials);
    if(mythrid == 0)
    fprintf(stdout,"Trials: %ld Ncirc: %ld Threads: %d Elapsed: %.2f  PI: %.8lf dpi: %.1g\n",
                    num_trials, ncirc, numthrd, tlaps, pi, dpi);

    MPI_Finalize();
    return 0;
}
```

This is similar to the pi calculation with OpenMP, especially the “Gustafson” variant. Note that here we use the function MPI_Reduce(…,MPI_SUM,…) to get the global sum of all points that hit the disk.

a) compare to the OpenMP version of this program (see LabX-7 exercises piprog_G.c) and annotate all differences in the source with comments explaining the need or reason for the change. You can get the differences between the programs with sdiff command. This will guide your comparison. (Comment only on substantial changes, if your source is much different, use –b –B flags if needed).

b) compile and run the program using the following sequence:

```
module load mpi/openmpi-3.1.2
mpicc -o pim -O3 piprog_MPI.c -lm
mpirun -np 4 -mca btl_base_warn_component_unused 0 ./pim
```

Turn this sequence into a bash script that you can invoke as ./run.sh #P where P is number of parallel threads. Note that the –mca flag in the mpirun command just disables some comments we do not want to consider at this time. Same can be done by setting the environment variable as follows:

```
NP=${1:-4}
export OMPI_MCA_btl_base_warn_component_unused=0
mpirun -np $NP ./pim
```

When running - choose the NPOINTS setting inside the program such that the run takes only a ~minute with 1 process.

[When you do the experiments, put into the program comments the time you found for different NPOINTS, so you can use it later]

c) For correct timing under load with several processes the program has to reserve resources. This will be done as follows:

```
salloc -N 1 -n 4 -t 10:00
```

This will allocate the resources for 10 minutes, which you can verify with the command: `env | grep SLURM` →

```
SLURM_NODELIST=cn01
SLURM_JOB_NAME=bash
SLURM_NODE_ALIASES=(null)
SLURM_NNODES=1
SLURM_JOBID=40905
SLURM_NTASKS=4
SLURM_TASKS_PER_NODE=4
SLURM_JOB_ID=40905
SLURM_CPUS_PER_TASK=4
SLURM_SUBMIT_DIR=/trinity/home
/i.zacharov/LabX_8
SLURM_NPROCS=4
SLURM_JOB_NODELIST=cn01
SLURM_CLUSTER_NAME=sandbox
SLURM_JOB_CPUS_PER_NODE=4
SLURM_SUBMIT_HOST=an
SLURM_JOB_PARTITION=cpu
SLURM_JOB_NUM_NODES=1
```

Running the MPI program: `mpirun -np 4 pim`

The –np specifies the number of parallel processes that MPI system will run, try –np 4 and –np 2 and put the time into a table with result of the calculation (pi and the statistical error in pi).

Cancel your reservation with `scancel $SLURM_JOB_ID`

2) Advanced.

Modify the MPI program for pi calculation for additional OpenMP parallelism:

a) to reduce time spent in the loop when adding the OpenMP threads

b) to increase the accuracy of calculation using same run time when adding the OpenMP threads.

Verify with the compiler that your program has no error and try to run. Most likely it will be difficult in sandbox, since the system is too small and has certain limitations.

No worry, submit your proposed variant of the program(s) as you think it should be done.

We will come back to this assignment and run it when we move on to Zhores proper.