Software solutions in High Performance Computing

Matthias Rauter

Universität Innsbruck, fall 2024

Day 2

Agenda

~~1. About me, about this.~~
~~2. Basics of HPC: Hardware, history, architecture: Distributed memory~~
~~3. Introduction to Message Passing Interface (MPI)~~
~~4. Coding with MPI~~
—
5. (Finish) exercise: Calculate Pi with MPI
6. Building and running on a cluster
7. Questions.

8. Extra: Sheard memory: Pthread, Mutex, Race conditions

Reminder: Prepare dev environment

In a PowerShell:

```
C:\Users\c0000000> wsl.exe --list --online
C:\Users\c0000000> wsl.exe --install Ubuntu
C:\Users\c0000000> wsl.exe
```

Then in the Linux shell (compiler, development library, mpi executables)

```
User@ZID-T9PCXXXX:~$ sudo apt-get update
User@ZID-T9PCXXXX:~$ sudo apt-get install g++ libopenmpi-dev openmpi-bin
```

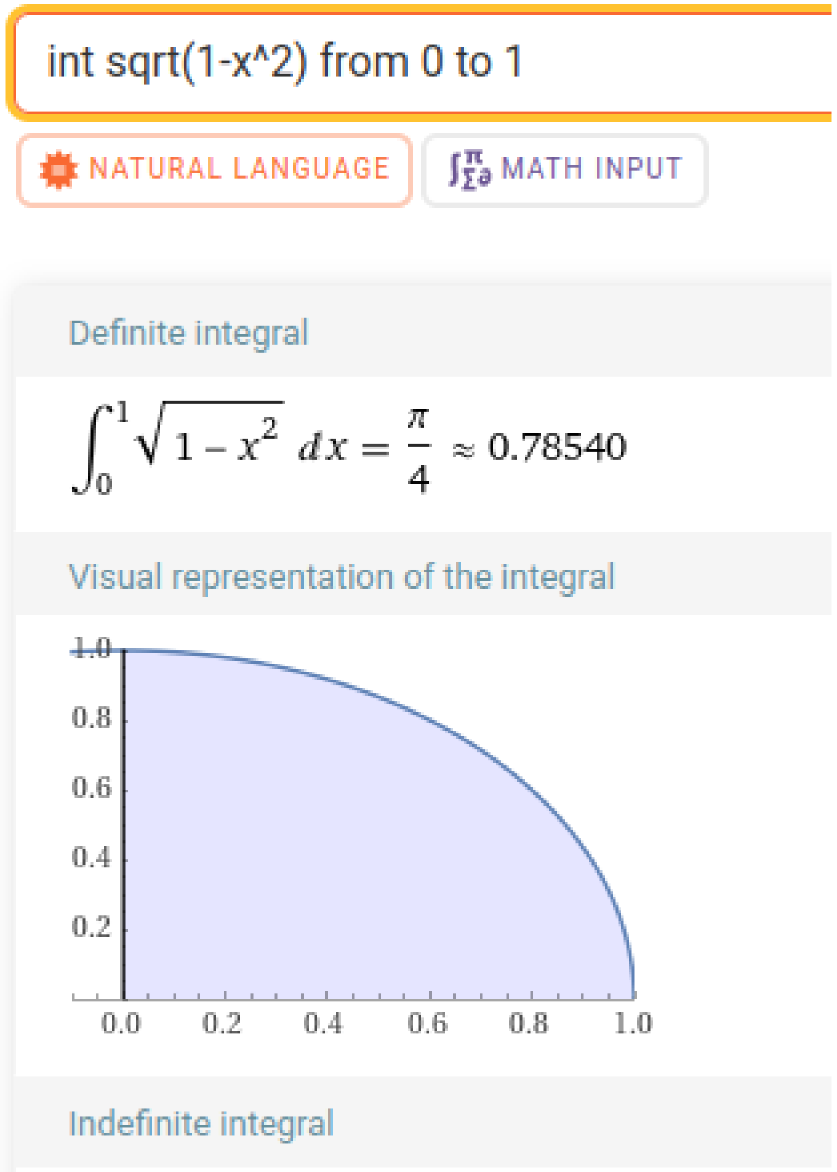
Change into your Windows home/desktop directory:

```
User@ZID-T9PCXXXX:~$ cd /mnt/c/Users/CXXXXXXX/Dokumente\ und\ Einstellungen
```

Clone the repo and test:

```
User@ZID-T9PCXXXX:~$ git clone
https://github.com/mattisnowman/mpiplayground
User@ZID-T9PCXXXX:~$ ./mpiplayground/cpp/run.sh
```

MPI Exercise: Distributed calculation of PI


int sqrt(1-x^2) from 0 to 1

Definite integral

$$\int_0^1 \sqrt{1-x^2}\, dx = \frac{\pi}{4} \approx 0.78540$$

Visual representation of the integral

Indefinite integral

- We can calculate PI with numerical integration of a quarter circle (f(x) = sqrt(1-x^2))
- Split the interval [0, 1] in N = 1e7 equal parts. Calculate their area (1/N * f(x)). Distribute the parts equally to all processes (easy option: for (int i = rank; i < N; i += size)).
- Develop 2 versions:
  - Gather/reduce results in every step of the for loop.
  - Gather/reduce results after the loop. Use a temporary variable within the loop.
- Run your examples with [1,2,4,8,16,32] nodes. How do they scale with number of nodes? Can you make sense of it? How would you describe the difference between the applications?

MPI Exercise: Tips

Add this to the beginning/end to get execution duration:

```
MPI_Barrier(MPI_COMM_WORLD);
const double begin = MPI_Wtime();

// do stuff

MPI_Barrier(MPI_COMM_WORLD);
const double end = MPI_Wtime();
if (!rank)
    std::cout << "Procs = " << size
              << ", Exec. duration = "
              << end - begin;
```

This does not include e.g. creating of processes. For that you have to use Linux bash commands, but this is good enough.

This is a very easy way to distribute over size processes:

```
for (int i = rank; i <= N; i += size)
{
    double x = from + i * d;
    double fx = f(x);
    part_sum += fx * d;
}
```

The two versions should have MPI_Reduce in:

```
for (int i = rank; i <= N; i += size){
    // do stuff
    MPI_Reduce(...);
}
```

And outside the loop (different granularity):

```
for (int i = rank; i <= N; i += size){
    // do stuff
}
MPI_Reduce(...);
```

Connecting to the cluster: LCC3

Connect with SSH:

```
User@ZID-T9PCXXXX:~$ ssh -l cbXXXXXX login.lcc3.uibk.ac.at
```

- cbXXXXXX number and password on request at Matthias
- Answer question about SSH key with yes.
- Enter password

Clone the repo, load mpi, test:

```
[CbXXXXXX@login.lcc3 ~]$ git clone
https://github.com/mattisnowman/mpiplayground
[CbXXXXXX@login.lcc3 ~]$ module load openmpi
[CbXXXXXX@login.lcc3 ~]$ ./mpiplayground/cpp/run.sh
```

Send stuff to the cluster with scp (use pwd on the cluster to know place):

```
User@ZID-T9PCXXXX:~$ scp -r mpiplayground/
cbXXXXXX@login.lcc3.uibk.ac.at:/home/cbXX/cbXXXXXX
```

Tutorials for Clusters and Slurm

- General information on the cluster LCC3: www.uibk.ac.at/zid/systeme/hpc-systeme/lcc3/

- Information about the batching system Slurm: www.uibk.ac.at/zid/systeme/hpc-systeme/common/tutorials/slurm-tutorial.html

- Other clusters, other rules (e.g. SGE baching system). Read the documentation, speak with the HPC guys (Zentrale Systeme at www.uibk.ac.at/zid/mitarbeiter/)

Some useful commands on LCC3

Print working directory (current directory): pwd

```
[CbXXXXXX@login.lcc3 ~]$ pwd
/home/cbXX/cbXXXXXX
```

Show content in directory: ls

```
[CbXXXXXX@login.lcc3 ~]$ ls
mpiplayground
```

Change directory: cd

```
[CbXXXXXX@login.lcc3 ~]$ cd mpiplayground/lcc
[CbXXXXXX@login.lcc3 ~]$ pwd
/home/cbXX/cbXXXXXX/mpiplayground/lcc
```

Read a file: less (use tail for just the end)

```
[CbXXXXXX@login.lcc3 ~]$ less slurm-140679.out
[CbXXXXXX@login.lcc3 ~]$ tail -n 100 slurm-140679.out
```

Some useful commands on LCC3: VIM

Edit a file: vim

```
[CbXXXXXX@login.lcc3 ~]$ vim run16.slurm
```

- Use arrows to navigate.
- Press "i" to enter insert mode. If it says --insert-- at the bottom you can edit code.
- Press ESC to exit enter mode.
- Write :q! to quit without saving.
- Write :x! to quit with saving.
- Cheat sheet: devhints.io/vim

This is not very user friendly (sorry), but good enough for small things.

Introduction to Slurm Workload Manager

- Distribute and organize cluster usage
- Reserve nodes for tasks and execute tasks there
- Put user tasks in queue if no nodes are free
- Tasks are described in shell scripts (like the run.sh, but *.slurm)

- Cheat sheet: slurm.schedmd.com/rosetta.pdf

Slurm: Some useful commands:

Put a task (skritp run16.slurm) in the Queue (It tells you the ID, remember it!):

```
[CbXXXXXX@login.lcc3 ~]$ sbatch run16.slurm
Submitted batch job 140680
```

See what is going on on the cluster:

```
[CbXXXXXX@login.lcc3 ~]$ sacct
140680               pi128        lva        uibk          256     RUNNING      0:0
```

Cancel job:

```
[CbXXXXXX@login.lcc3 ~]$ scancel 140680
```

Inspect a job "live", run tail –f on the log-file to get updates:

```
[CbXXXXXX@login.lcc3 ~]$ srun --jobid=140680 --overlap --pty bash
(JobID 140683)[cb761251@n001.intern.lcc3 lcc]$ tail -f slurm-140683.out
23.8382s: Step 270000000000/1000000000000
24.7195s: Step 280000000000/1000000000000
25.6007s: Step 290000000000/1000000000000
```

"Input file" for Slurm (SGE is similar)

```
#!/bin/bash

# Some options for sbatch:
#SBATCH --job-name=pi16
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1G

# Load required modules
module purge
module load openmpi

# Compile code
mpicxx -o mpi_pi mpi_pi.cpp -lm -Wall -std=c++11
# Run code
mpirun --mca btl_openib_allow_ib true -np 16 mpi_pi
```

Exercise: Calcualte MPI on the cluster:

- Change your source code to use N=1e12 integration points.
- Add `#include <iostream>` to make it compile on LCC.
- Add an update message at all 1% progress:

```
if (i % (N/100) == 0)
   std::cout << MPI_Wtime() - begin << "s: Step " << i << "/" << N << std::endl;
```

- Send your C-code to the cluster:

```
User@ZID-T9PCXXXX:~$ scp sourcefile
cbXXXXXX@login.lcc3.uibk.ac.at:destination/destinationfile
```

- Write a *.slurm script that executes it on 32 cores. Send it to the cluster.
- Put the slurm script into the Queue (sbatch).
- Find out if the script is running (sacct).