MPI-based parallelism

CTBP Tech Talk

11/15/2023

Paul Whitford

Northeastern University

https://github.com/Whitford/ctbp-techtalks/

What is MPI?

- MPI=Message Passing Interface
- Unlike OpenMP, MPI is not built into the compiler
- MPI code must be build with MPI compilers
- While OpenMP launches multiple threads for a single process, MPI launches multiple processes
- MPI processes are organized and indexed by ”rank”
- In OpenMP:
  - All threads can access the same addresses in memory
- Each rank :
  - Has it’s own memory
    - i.e. ranks will never confuse variables: variables with identical names are local to each rank
  - Can work on completely independent calculations
  - Can be on its own node, or the same node
  - Any communication between ranks must be explicit **You need to say where and when data moves between ranks. Nothing is automatic!**
- Multi-level parallelism: MPI ranks can each launch their own OpenMP threads – threads on a rank are local to the rank
- Global variables don’t really exist… and updated global variable must be communicated to the other ranks
- For more comprehensive introductions:
  - https://warwick.ac.uk/research/rtp/sc/rse/training/intrompi/
  - http://www.cs.cmu.edu/afs/cs/academic/class/15418-s19/www/lectures/rec_05.pdf

https://github.com/Whitford/ctbp-techtalks/

MPI Concepts

- Processes are collected into groups
- Messages may be sent between processes in a group via a "communicator"
- If not specified, all processes will share a single communicator called MPI_COMM_WORLD
- When one wants to send/receive messages, you must use the communicator. Think of it like a telephone service. You can use it to communicate with some distant node (i.e. a person).

https://github.com/Whitford/ctbp-techtalks/

MPI Datatypes

- When sending a message, you need to specify the:
  - Address
  - Type of data
  - Number of data elements to be transferred
- Data types can be:
  - Default definitions, such as MPI_INT, MPI_DOUBLE_PRECISION
  - Custom definitions, such as user-defined data structures

https://github.com/Whitford/ctbp-techtalks/

MPI Tags

- When sending a message, one need to “tag” it. This way, the receiver knows what message it is expecting to receive.

https://github.com/Whitford/ctbp-techtalks/

MPI Basic Calls

- MPI_INIT – initialize the MPI environment in your program
- MPI_FINALIZE – take everything down
- MPI_COMM_SIZE – the number of ranks available
- MPI_COMM_RANK – the index of each rank
- MPI_SEND – send a message to another rank
- MPI_RECV – receive a message from another rank

https://github.com/Whitford/ctbp-techtalks/

Example of the idea behind MPI

- Imagine you have an array with N elements and you want to square each one
- With OpenMP, you simply parallelize the loop

```
#pragma omp parallel for
    for (int j = 0; j < N; j++) {
        array[j]=array[j]*array[j]
```

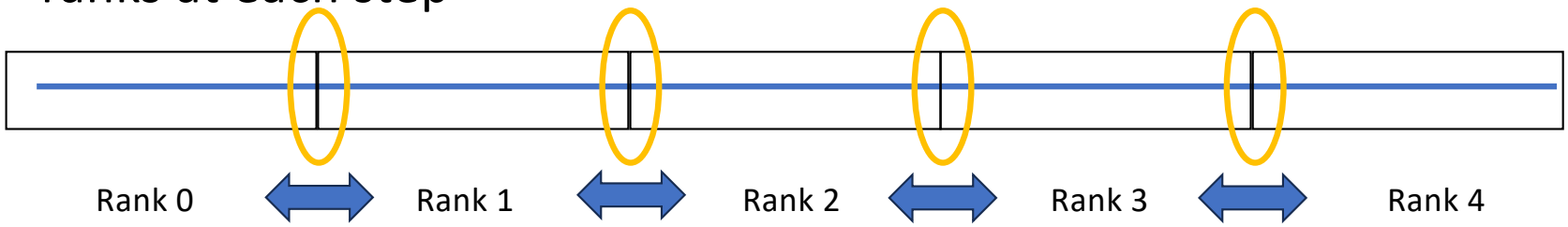
- With MPI, you must communicate the relevant array elements to each rank, so that they are local to that rank. Then, you work on those local element

```
    for (int j = 0; j < Nlocal; j++) {
        arraylocal[j]=arraylocal[j]*arraylocal[j]
```

- Every rank has its own version of arraylocal, which is a fragment of aray

What are we going to do?

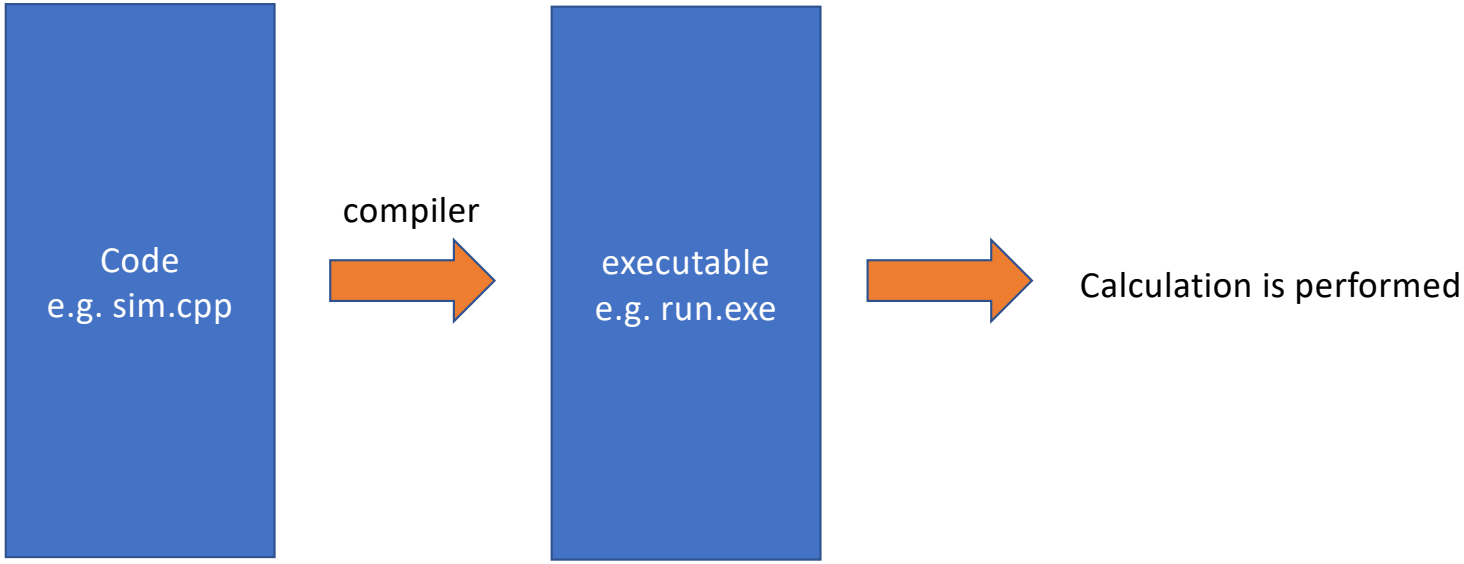
- Look at our string system.
- “cut” it into pieces and have a piece on each rank
- The rank will work on its piece
- The end particles of each piece will be communicated to neighboring ranks at each step



Particles at the boundary are transferred between ranks

https://github.com/Whitford/ctbp-techtalks/

C++ is still a compiled language, but we need a different compiler



https://github.com/Whitford/ctbp-techtalks/

Our standard system

- N particles connected by harmonic springs with fixed boundaries
- Initialize the system with some initial y displacement for all particles
- Release the system and watch the dynamics using VMD.

- We are going to use C++ and OpenMP parallelization

https://github.com/Whitford/ctbp-techtalks/

View string.mpi.cpp

https://github.com/Whitford/ctbp-techtalks/

Running and compiling, today

- On a compute cluster
  - Can probably just load a module for mpi/openmpi/mpich
  - You will want to get an interactive node:
    - srun -N 1 -n 4 -c 6 -t 02:00:00 --pty /bin/bash
    - -N 1 means all ranks are on the same node (but they don't have to be)
    - -n 4 means we are asking for 4 ranks
    - -c 6 means each rank can access 6 cores
  - If all is working, you can call the compiler with:
    - mpicxx -O3 string.mpi.cpp -o run.exe
    - run.exe is the executable
  - Run the program with:
    - mpirun –n 4 run.exe
    - mpirun enables the MPI environment, which then launches the executable
- Using a prepared container (if docker is installed on your machine)
  - Download and launch container
    - `docker pull smogserver/techtalk:latest`
    - `docker run –it ––rm –v $(pwd):/workdir smogserver/techtalk:latest`
  - Compile
    - mpiCC –O3 string.mpi.cpp -o run.exe
  - Run the program:
    - mpirun –n 4 run.exe

https://github.com/Whitford/ctbp-techtalks/

Our standard string system: string.mpi.cpp

- 100k particles for 1M steps
- turn on compiler optimization with the –O3 flag during the compile step
- run the job in “serial” with: mpirun –n 1 run.exe
  - i.e. launch 1 rank
- Now try to run it with 2, 4, 8 ranks
- We will discuss a bit about what is going on…

https://github.com/Whitford/ctbp-techtalks/

Add OpenMP parallelization (look familiar?)

- Before any loop that you want to parallelize, add:
  - #pragma omp parallel for
- Add parallelization and re-run your system with optimization
  - Don’t forget the –fopenmp flag, to enable OpenMP
- Change the number of cores by issuing the following command on the command line:
  - export OMP_NUM_THREADS=1
  - You can set to 1, to give a single thread per rank.
  - Try re-running with 2, 4, 8, 16 threads (per rank)
- Re-check your performance

https://github.com/Whitford/ctbp-techtalks/