Parallel Programming

Libraries and implementations

Partners

Funding

Horizon 2020
European Union Funding
for Research & Innovation

Reusing this material

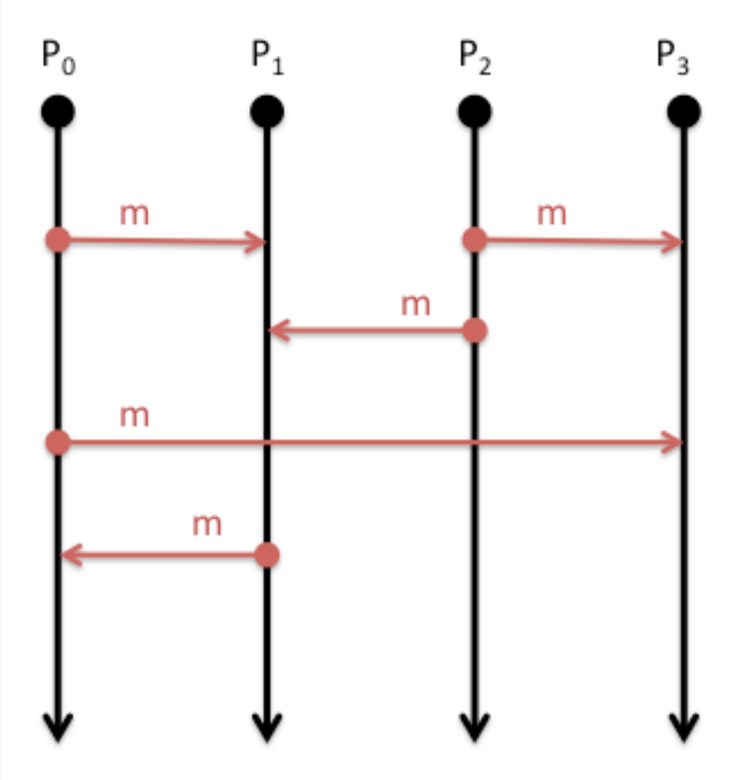
Outline

- MPI – distributed memory de-facto standard
- OpenMP – shared memory de-facto standard
- CUDA – GPGPU de-facto standard
- Other approaches
- Summary

MPI Library

Distributed, message-passing programming

Message-passing concepts



Explicit Parallelism

- In message-passing all the parallelism is explicit
  - The program includes specific instructions for each communication
  - **What** to send or receive
  - **When** to send or receive
  - Synchronisation
- It is up to the developer to design the parallel decomposition and implement it
  - How will you divide up the problem?
  - When will you need to communicate between processes?

Message Passing Interface (MPI)

- MPI is a portable **library** used for writing parallel programs using the message passing model
  - You can expect MPI to be available on any HPC platform you use
- Based on a number of **processes** running independently in parallel
  - HPC resource provides a command to launch multiple processes simultaneously (*e.g.* mpiexec, aprun)
- There are a number of different implementations but all should support the MPI-3 standard
  - As with different compilers, there will be variations between implementations but all the features specified in the standard should work
  - Examples: MPICH, Open MPI

Point-to-point communications

- A message sent by one process and received by another
- Both processes are actively involved in the communication – not necessarily at the same time
- Wide variety of semantics provided:
  - Blocking vs. non-blocking
  - Ready vs. synchronous vs. buffered
  - Tags, communicators, wild-cards
  - Built-in and custom data-types
- Can be used to implement any communication pattern
  - Collective operations, if applicable, can be more efficient

Collective communications

- A communication that involves all processes
  - “all” within a communicator, i.e. a defined sub-set of all processes
- Each collective operation implements a particular communication pattern
  - Easier to program than lots of point-to-point messages
  - Should be more efficient than lots of point-to-point messages
- Commonly used examples:
  - Broadcast
  - Gather
  - Reduce
  - AllToAll

Example: MPI HelloWorld

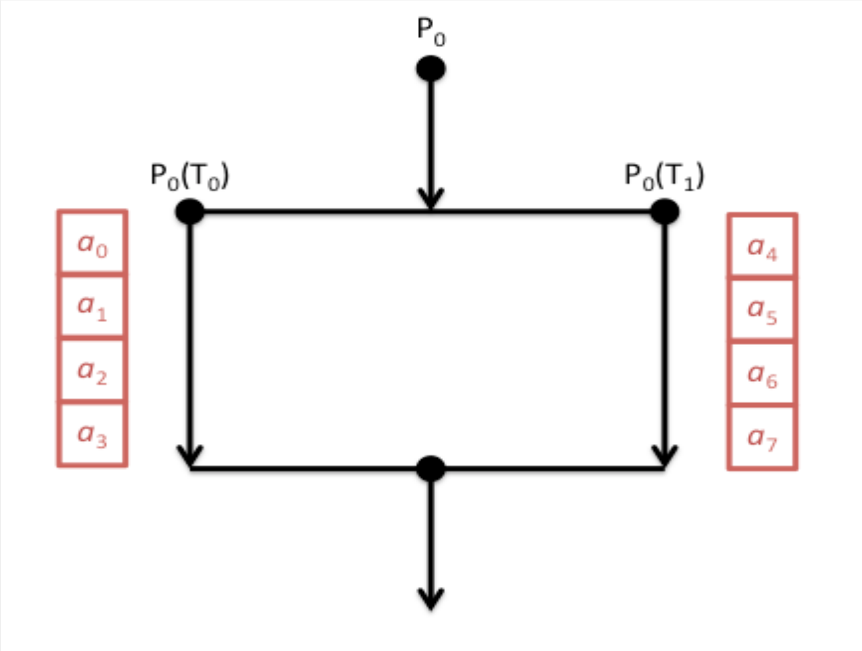
```
int main(int argc, char* argv[])
{
   int size,rank;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   printf("Hello world - I'm rank %d of %d\n", rank, size);

  MPI_Finalize();
  return 0;
}
```

OpenMP

Shared-memory parallelism using directives

Shared-memory concepts

- **Threads** “communicate” by having access to the same memory space
  - Any thread can alter any bit of data
  - No explicit communications between the parallel tasks



OpenMP

- OpenMP is an Application Program Interface (API) for shared memory programming
  - You can expect OpenMP to be supported by all compilers on all HPC platforms
- Not a library interface like MPI
  - You interact through **directives** in your program source rather than calling functions/subroutines
- Parallelism is **less explicit** than MPI
  - You specify which parts of the program you want to parallelise and the compiler produces a parallel executable
- Also used for programming Intel Xeon Phi

Loop-based parallelism

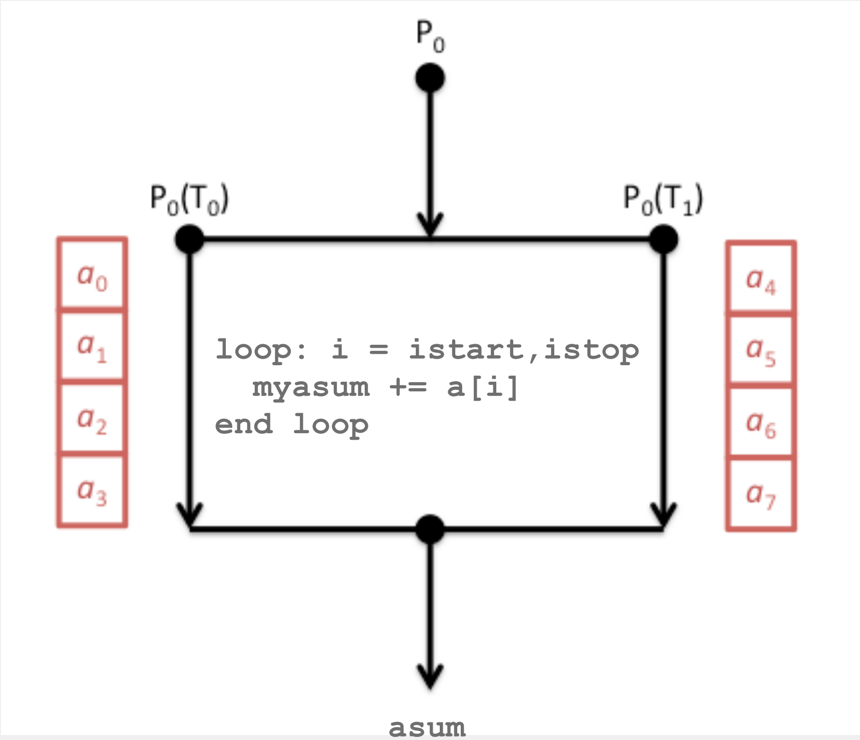
- The most common form of OpenMP parallelism is to parallelise the work in a loop
  - The OpenMP directives tell the compiler to divide the iterations of the loop between the threads

```
#pragma omp parallel shared(a,b,c,chunk) private(i)
{
   #pragma omp for schedule(dynamic,chunk) nowait
   for (i=0; i < N; i++) {
     c[i] = a[i] + b[i];
   }
}
```

Addition example

```
asum = 0.0
#pragma omp parallel \
shared(a,N) private(i) \
reduction(+:asum)
{
   #pragma omp for
   for (i=0; i < N; i++)
   {
     asum += a[i];
   }
}
printf("asum = %f\n", asum);
```

asum=0



CUDA

Programming GPGPU Accelerators

CUDA

- CUDA is an Application Program Interface (API) for programming NVIDIA GPU accelerators
  - Proprietary software provided by NVIDIA. Should be available on all systems with NVIDIA GPU accelerators
  - Write GPU specific functions called *kernels*
  - Launch kernels using syntax within standard C programs
  - Includes functions to shift data between CPU and GPU memory
- Similar to OpenMP programming in many ways in that the parallelism is implicit in the kernel design and launch
- More recent versions of CUDA include ways to communicate directly between multiple GPU accelerators (*GPUdirect*)

Example:

```
// CUDA kernel. Each thread takes care of one element of c
__global__ void vecAdd(double *a, double *b, double *c, int n)
{
    // Get our global thread ID
    int id = blockIdx.x*blockDim.x+threadIdx.x;

    // Make sure we do not go out of bounds
    if (id < n)
        c[id] = a[id] + b[id];
}

// Called with
vecAdd<<<gridSize, blockSize>>>(d_a, d_b, d_c, n);
```

OpenCL

- An open, cross-platform standard for programming accelerators
  - includes GPUs, e.g. from both NVIDIA and AMD
  - also Xeon Phi, Digital Signal Processors, ...
- Comprises a language + library
- Harder to write than CUDA if you have NVIDIA GPUs
  - but portable across multiple platforms
  - although maintaining performance is difficult

Other approaches

Niche and future implementations

Other parallel implementations

- Shared memory
  - POSIX Threads (Pthreads), Thread Building Blocks (TBB), Cilk
- Partitioned Global Address Space (PGAS)
  - Coarray Fortran, Unified Parallel C (UPC), Chapel
- Single-sided Remote Direct Memory Access (RDMA)
  - SHMEM, OpenSHMEM
- OpenACC
  - Directive-based approach for programming accelerators

Summary

Parallel Implementations

- **Distributed memory** programmed using **MPI**
- **Shared memory** programmed using **OpenMP**
- **GPU** accelerators most often programmed using **CUDA**

- Hybrid programming approaches (e.g. MPI/OpenMP) are becoming more common
  - They match the hardware layout more closely

- A number of other, more experimental approaches are available