Parallel Programming with MPI

Science and Technology Support
Ohio Supercomputer Center
1224 Kinnear Road.
Columbus, OH 43212
(614) 292-1800
oschelp@osc.edu
http://www.osc.edu/supercomputing/

Functions - Scope of Activity

Supercomputing. Computation, software, storage, and support services empower Ohio's scientists, engineers, faculty, students, businesses and other clients.

Networking. Ohio's universities, colleges, K-12 and state government connect to the network. OSC also provides engineering services, video conferencing, and support through a 24x7 service desk.

Research. Lead science and engineering projects, assist researchers with custom needs, partner with regional, national, and international researchers in groundbreaking initiatives, and develop new tools.

Education. The Ralph Regula School of Computational Science delivers computational science training to students and companies across Ohio.

Table of Contents

Ohio Supercomputer Center

Acknowledgments

- Edinburgh Parallel Computing Centre at the University of Edinburgh, for material on which this course is based
- Dr. David Ennis (formerly of the Ohio Supercomputer Center), who initially developed this course

Setting the Stage

- Overview of parallel computing
- Parallel architectures
- Parallel programming models
- Hardware
- Software

Overview of Parallel Computing

- In **parallel computing**, a program uses concurrency to either
  - decrease the runtime needed to solve a problem
  - increase the size of problem that can be solved
- Mainly this is a price/performance issue
  - Vector machines (e.g. Cray X1e, NEC SX6) are very expensive to engineer and run.
  - Massively parallel systems went out of vogue for a few years but are making a bit of a comeback at the very large scale (e.g. Cray XT-3/4, IBM BlueGene L/P).
  - Parallel clusters with commodity hardware/software are the lion's share of HPC hardware today.

Writing a parallel application

- Decompose the problem into tasks
  - Ideally, each task can be worked on independently of others
- Map tasks onto “threads of execution”
- Threads have shared and local data
  - Shared: used by more than one thread
  - Local: private to each thread
- Develop source code using some parallel programming environment
- Choices may depend on (among many things)
  - hardware platform
  - level of performance needed
  - nature of the problem

Ohio Supercomputer Center

Parallel architectures

- **Distributed memory**
  - Each processor has local memory
  - Cannot directly access the memory of other processors
- **Shared memory**
  - Processors can directly reference memory attached to other processors
  - Shared memory may be *physically* distributed
    - Non-uniform memory architecture (NUMA)
    - The cost to access remote memory may be high
  - Several processors may sit on one memory bus (SMP)
- **Hybrids** are now very common, e.g. IBM e1350 Opteron Cluster:
  - 965 compute nodes, each with 4-8 processor cores sharing 8-16 GB of memory
  - High-speed Infiniband interconnect between nodes

Parallel programming models

- Distributed memory systems
  - For processors to share data, the programmer must explicitly arrange for communication - "**Message Passing**"
  - Message passing libraries:
    - MPI ("Message Passing Interface")
    - PVM ("Parallel Virtual Machine")
    - Shmem, MPT (both Cray only)
- Shared memory systems
  - "Thread" based programming
  - Compiler directives (OpenMP; various proprietary systems)
  - Can also do explicit message passing, of course

Parallel computing: Hardware

- In very good shape!
- Processors are cheap and powerful
  - EM64T, Itanium, Opteron, POWER, PowerPC, Cell…
  - Theoretical performance approaching 10 GFLOP/sec or more
- SMP nodes with 8-32 processor cores are common
  - Multicore chips becoming ubiquitous
- Clusters with hundreds of nodes are common.
- Affordable, high-performance interconnect technology is available.
- Systems with a few hundreds of processors and good inter-processor communication are not hard to build.

Parallel computing: Software

- Not as mature as the hardware
- The main obstacle to making use of all this power
  - Perceived difficulties with writing parallel codes outweigh the benefits
- Emergence of standards is helping enormously
  - MPI
  - OpenMP
- Programming in a shared memory environment generally easier
- Often better performance using message passing
  - Much like assembly language vs. C/Fortran

Brief History of MPI

- What is MPI?
- MPI forum
- Goals and scope of MPI
- MPI on OSC parallel platforms

What is MPI?

- **M**essage **P**assing **I**nterface
- What are the messages? **DATA**
- Allows data to be passed between processes in a distributed memory environment

MPI Forum

- Sixty people from forty different organizations
- International representation
- MPI 1.1 Standard developed from 1992-1994
- MPI 2.0 Standard developed from 1995-1997
- Recently resumed activity in 2008
- Standards documents
  - http://www.mcs.anl.gov/mpi
  - http://www.mpi-forum.org/docs/docs.html

Goals and scope of MPI

- MPI's prime goals are:
  - To provide source-code portability
  - To allow efficient implementation
- It also offers:
  - A great deal of functionality
  - Support for heterogeneous parallel architectures

MPI on OSC platforms

- Itanium 2 cluster
  - MPICH/ch_gm for Myrinet from Myricom (default)
  - MPICH/ch_p4 for Gigabit Ethernet from Argonne Nat'l Lab
- SGI Altix
  - MPICH/ch_p4 for shared memory from Argonne Nat'l Lab (default)
  - SGI MPT
- Pentium 4 cluster
  - MVAPICH for InfiniBand from OSU CSE (default)
  - MPICH/ch_p4 for Gigabit Ethernet from Argonne Nat'l Lab
- BALE Opteron cluster
  - MVAPICH for InfiniBand from OSU CSE (default)
- IBM e1350 Opteron cluster
  - MVAPICH for InfiniBand from OSU CSE (default)
  - MVAPICH2 for InfiniBand from OSU CSE (in progress)

Using MPI on the Systems at OSC

- Compile with the MPI wrapper scripts (`mpicc`, `mpiCC`, `mpif77`, `mpif90`)
- Examples:

```
$ mpicc myprog.c
$ mpif90 myprog.f
```

- To run:

In batch: **`mpiexec ./a.out`**