Parallel Machine Learning and Artificial Intelligence

Dr. Handan Liu

h.liu@northeastern.edu

Northeastern University

Content

- High Performance Parallel Computing
  - Overview (done)
  - Concepts and Terminology (done)
  - Parallel Memory Architectures
  - Parallel Programming Model
  - Parallel Examples and Exercises

Parallel Computer Memory Architectures

Architecture in High Performance Computing

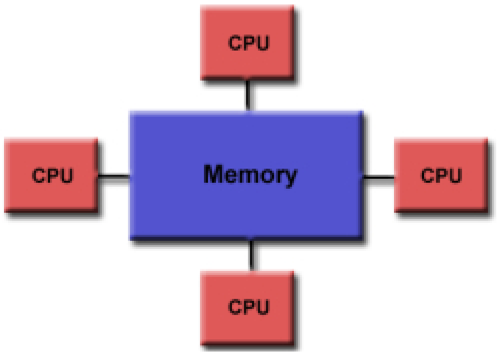
- Memory Access
  - Shared memory
  - Distributed memory
  - Hybrid Distributed-Shared memory

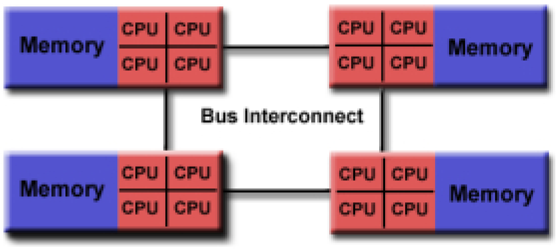
- Processor Type
  - Single core CPU
  - Multi-core CPU (since 2005)
  - Accelerators
    - NVidia GPGPU
    - Intel Xeon Phi (MIC – Intel's Many Integrated Cores)

Shared Memory Architecture

- Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.
- Historically, shared memory machines have been classified as **UMA** and **NUMA**, based upon memory access times.
  - UMA: Uniform Memory Access
  - NUMA: Non-Uniform Memory Access

Shared Memory Architecture





Uniform Memory Access (UMA):

- SMP: Symmetric Multiprocessor (**SMP**) machines
- Sometimes called <u>CC-UMA - Cache Coherent UMA.</u>

Non-Uniform Memory Access (NUMA):

- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then it may also be called <u>*cc-NUMA - cache coherent NUMA*</u>

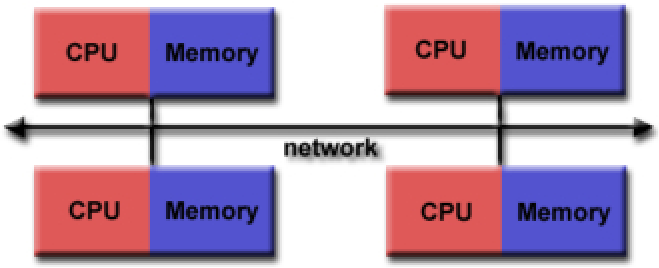
Shared Memory Architecture

- Advantages:
  - Global address space provides a user-friendly programming perspective to memory
  - Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs

- Disadvantages:
  - Lack of scalability between memory and CPUs.
  - Programmer responsibility for synchronization constructs that ensure "correct" access of global memory.

Distributed Memory Architecture

- Distributed memory systems require a communication network to connect inter-processor memory.

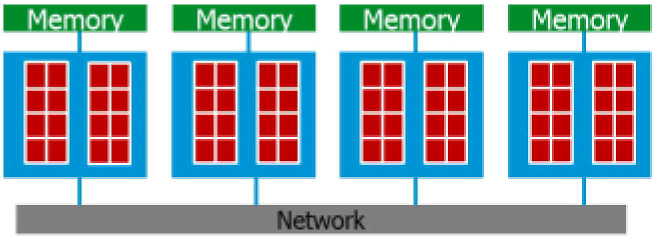


- Each processor has its own local memory.
- Data exchange by message passing over a network between the processors.
- Synchronization between tasks is likewise the programmer's responsibility.

Distributed Memory Architecture

- Advantages:
  - Memory is scalable with the number of processors.
  - Access its own memory without the overhead
  - Cost effectiveness
- Disadvantages:
  - High requirements for programmer skills.
  - Harder to convert serial code to distributed memory architecture.
  - Non-uniform memory access time.

Hybrid Distributed-Shared Memory Architecture

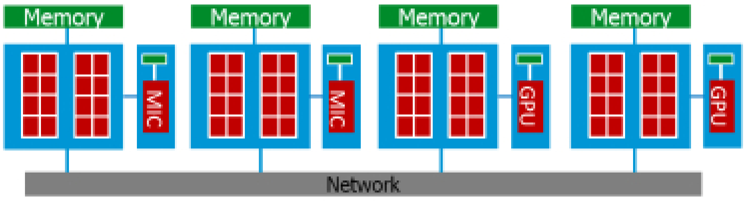


Advantages and Disadvantages:

Whatever is common to both shared and distributed memory architectures:

- Increased scalability is an important advantage.
- Increased programming complexity is a major disadvantage.

Co-Processor Accelerating Architecture



co-processor

- Calculations made in both CPUs and accelerators
- No longer limited to single precision calculations
- Load balancing critical for performance
- Typically communicate over PCI-e bus

Parallel Programming Models

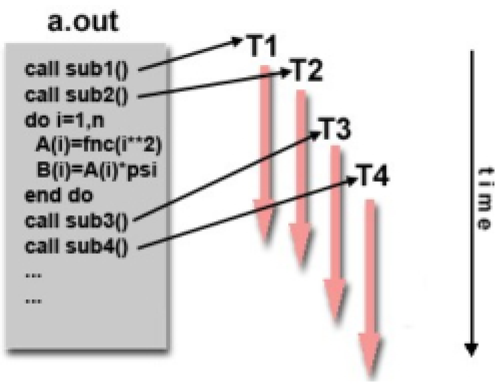
Overview

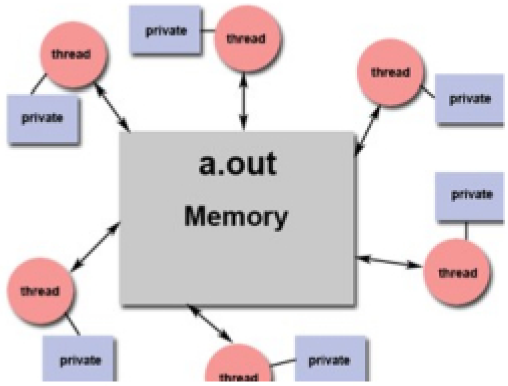
- Parallel programming models exist as an abstraction above hardware and memory architectures

- There are several parallel programming models in common use:
  - Shared Memory Model
  - Distributed Memory Model
  - Data Parallel Model
  - Hybrid Model
  - Single Program Multiple Data (SPMD)
  - Multiple Program Multiple Data (MPMD)

Shared Memory Programming – Threads Model

In the threads model of parallel programming, a single "heavy weight" process can have multiple "light weight", concurrent execution paths.





What is a Thread?

- Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system. But what does this mean?
- To the software developer, the concept of a "procedure" that runs independently from its main program may best describe a thread.
- To imagine a main program (a.out)
- From the perspective of the hardware layer

Shared Memory Programming – Threads Model

<u>Implementations:</u>

- From a programming perspective, threads implementations commonly comprise:
  - A library of subroutines that are called from within parallel source code
    - POSIX Threads
  - A set of compiler directives imbedded in either serial or parallel source code
    - OpenMP

Northeastern University
College of Engineering

Threads Models

- POSIX Threads
    - Commonly referred to as Pthreads
    - Library based; requires parallel coding
    - C Language only; Interfaces for Perl, Python and others exist
    - Part of Unix/Linux operating systems
    - Very explicit parallelism; requires significant programmer attention to detail.

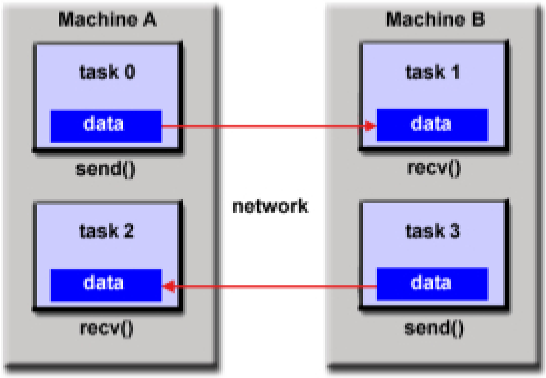
Threads Models

- OpenMP -- Open Multi Processing
  - Industry standard for shared memory programming
  - Compiler directive based
  - Portable / multi-platform, including Unix and Windows platforms
  - Available in C/C++ and Fortran implementations
  - Can be very easy and simple to use - provides for "incremental parallelism". Can begin with serial code.
  - Other threaded implementations are common:
    - ✔ Microsoft threads
    - ✔ Java, Python threads
    - ✔ CUDA threads for GPUs

OpenMP -- Tutorials & Articles
https://www.openmp.org/resources/tutorials-articles/

Distributed Memory / Message Passing Model

- A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines.
- Tasks exchange data through communications by sending and receiving messages.
- Data transfer usually requires cooperative operations to be performed by each process.



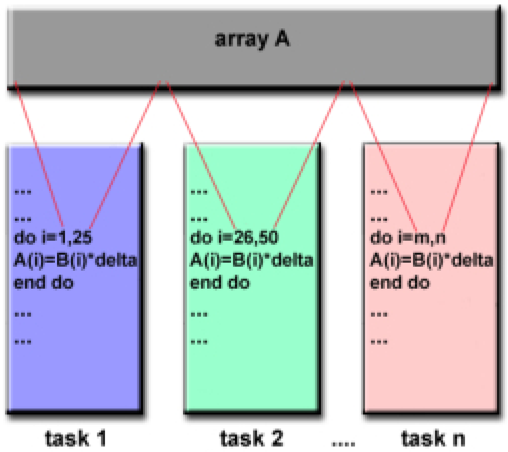
Implementations:

- Message passing implementations usually comprise a library of subroutines
- The programmer is responsible for determining all parallelism.

Data Parallel Model

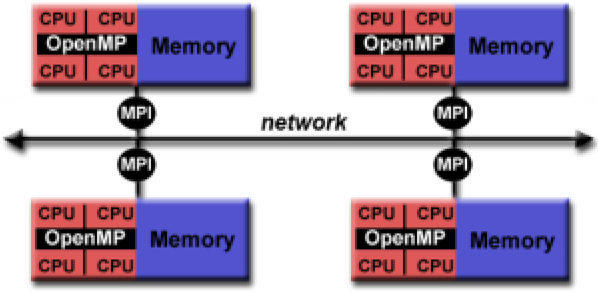
May also be referred to as the Partitioned Global Address Space (PGAS) model.

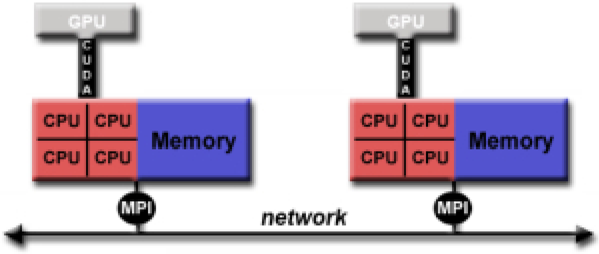
- Address space is treated globally
- Most of the parallel work focuses on performing operations on a data set.
- A set of tasks work collectively on the same data structure, however, each task works on a different partition of the same data structure.
- Tasks perform the same operation on their partition of work.



Implementations:

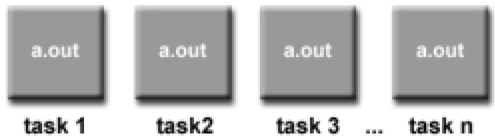
Hybrid Model





Single Program Multiple Data (SPMD) – SPMD

- SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- Single Program: All tasks execute their copy of the same program simultaneously. This program can be threads, message passing, data parallel or hybrid.
- Multiple Data: All tasks may use different data



Multiple Program Multiple Data – MPMD

- Like SPMD, MPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- Multiple Program: Tasks may execute different programs simultaneously. The programs can be threads, message passing, data parallel or hybrid.
- Multiple Data: All tasks may use different data

- Stay safe!
- See you next class!

<u>Next Lecture will Continue:</u>

High Performance Parallel Computing

- Overview
- Concepts and Terminology
- Parallel Memory Architectures
- Parallel Programming Model
- Parallel Examples and Exercises