Introduction

In this tutorial we will explore the differences between serial and parallel computation, and look at how parallel programs work in general. We will also assess a couple of parallel program solutions that utilize the multiprocessor environment of a supercomputer.

Useful Links

https://computing.llnl.gov/tutorials/parallel_comp/#Whatis

Why Parallel?

Assume you are attempting to assemble a 10,000-piece jigsaw puzzle on a rainy weekend. The number of pieces is staggering, and instead of a weekend it takes you several weeks to finish the puzzle. Now assume you have three friends helping with the puzzle -- it goes much faster, and you are able to finish the puzzle over the weekend.

This principle is the central idea behind parallel computation. You can dramatically cut down on computation by splitting one large task into smaller tasks that multiple processors can perform all at once. With parallel processes a task that would normally take several weeks can potentially be reduced to several hours.

Serial and Parallel Processes

A serial process is simply a process that is run entirely by one core of one processor. This means tasks are run one after another as they appear in code. This is analogous to you doing the jigsaw puzzle on your own.

A parallel process is a process that is divided among multiple cores in a processor or set of processors. Each sub process can have its own set of memory as well as share memory with other processes. This is analogous to doing the puzzle with the help of friends.

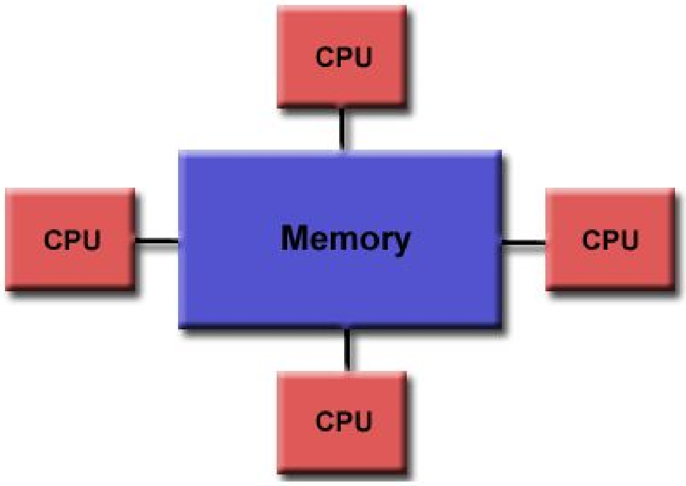
A supercomputer has a large network of nodes with many cores, and is designed for parallel computing. However, without the use of parallel software design, we cannot fully utilize supercomputing resources.

How parallel computation works

Parallel computation connects multiple processors to memory that is either pooled or connected via high speed networks. Here are three different types of parallel computation.

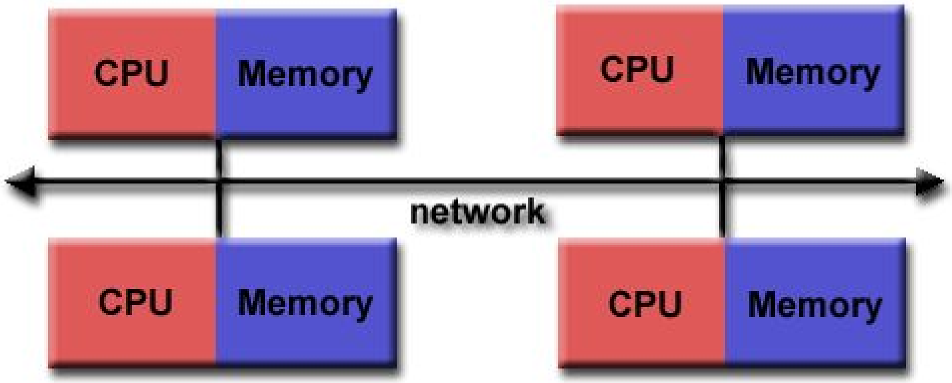
Shared Memory Model:

In a shared memory model all processors to have access to a pool of common memory that they can freely use.



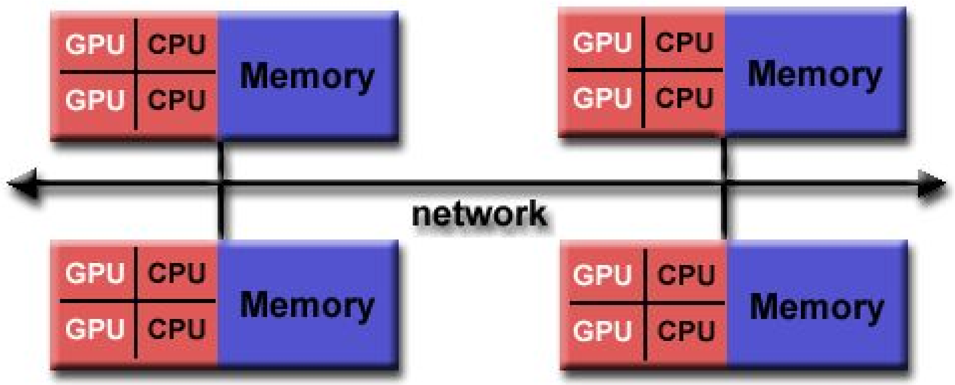
(Image courtesy of LLNL https://computing.llnl.gov/tutorials/parallel_comp/)

Distributed Memory Model:
In a distributed memory model a separate segment of memory is available to each processor. Because memory isn't shared inherently, information that must be shared between processes is sent over a network.



(Image courtesy of LLNL https://computing.llnl.gov/tutorials/parallel_comp/)

Distributed/Shared Model:
A split distributed/shared model is a hybrid between a shared and distributed model and has the properties of both. Each separate set of processors sharing a set of common memory is called a node.



(Image courtesy of LLNL https://computing.llnl.gov/tutorials/parallel_comp/)

Summit utilizes a hybrid distributed/shared model: there are 380 nodes, each having 24 cores.

Tools for Parallel Programming

Two common solutions for creating parallel code are OpenMP and MPI. Both solutions are limited to the C++ or Fortran programming languages. (Though other languages may be extended with C++ or Fortran code to utilize OpenMP or MPI.)

OpenMP

OpenMP ("Open Multi-Processing") is a compiler-side application programming interface (API) for creating code that can run on a system of threads. No external libraries are required in order to parallelize your code. OpenMP is often considered more user friendly with thread safe methods and parallel sections of code that can be set with simple scoping.

OpenMP is, however, limited to the amount of threads available on a node -- in other words, it follows a shared memory model. On Summit, this means that no more than 24 processors can be utilized with programs parallelized using OpenMP.

MPI

MPI ("Message Passing Interface") is a library standard for handling parallel processing. Unlike OpenMP, MPI has much more flexibility in how individual processes handle memory. MPI is also compatible with multi-node structures, allowing for very large, multi-node applications (i.e, distributed memory models). MPI is, however, often considered more difficult to learn. Regardless, learning the library provides a user with the ability to maximize processing ability.

MPI is a library standard, meaning there are several libraries based on MPI that you can use to develop parallel code. Two solutions available on Summit are OpenMPI and Intel MPI.