UNIT I – FOUNDATIONS OF HIGH PERFORMANCE COMPUTING

Introduction to High Performance Computing

High Performance Computing (HPC) refers to the aggregation of computing resources to deliver much higher computational performance than a typical desktop or workstation — enabling the solving of large, complex problems in science, engineering, finance, and AI.

Why HPC? Problems that require HPC share three characteristics:

- They are computationally intensive (billions/trillions of operations)
- They are data-intensive (terabytes to petabytes of data)
- They have real-world urgency (climate prediction, drug discovery, crash simulation)

If a single CPU core is a single worker solving a math problem, HPC is stadium full of workers all solving different parts of the same problem at a time.

1.1 Super Computing

Supercomputing is a form of high-performance computing that determines or calculates by using a powerful computer, a supercomputer, reducing overall time to solution.

Supercomputing technology comprises supercomputers, the fastest computers in the world. Supercomputers are made up of interconnects, I/O systems, memory and processor cores.

Unlike traditional computers, supercomputers use more than one central processing unit (CPU). These CPUs are grouped into compute nodes, comprising a processor or a group of processors—symmetric multiprocessing (SMP)—and a memory block. At scale, a supercomputer can contain tens of thousands of nodes. With interconnect communication capabilities, these nodes can collaborate on solving a specific problem. Nodes also use interconnects to communicate with I/O systems, like data storage and networking.

Key characteristics of supercomputers:

- Thousands to millions of processor cores
- Extremely fast interconnects (InfiniBand)
- High-bandwidth parallel file systems (Lustre, GPFS)

- Liquid or immersion cooling

1.2 History of Super Computing

From the early vector computers of the 1960s to today's exascale supercomputers, advances in processor technology, parallel computing, memory systems, and networking have dramatically increased computational performance. Supercomputers have played a vital role in weather forecasting, climate modeling, space exploration, nuclear research, artificial intelligence, genomics, and many other fields. The evolution of supercomputing has shifted from increasing processor speed to employing thousands or even millions of processors working together in parallel to achieve unprecedented levels of performance

1964 – CDC 6600: Designed by Seymour Cray, widely regarded as the world's first supercomputer.

2002 – Earth Simulator (Japan): Became the world's fastest supercomputer, primarily used for climate and geophysical simulations.

2008 – IBM Roadrunner: First supercomputer to achieve 1 PetaFLOPS (PFLOPS).

2022 – Frontier (USA): First supercomputer to officially achieve Exascale computing (more than 1 ExaFLOPS).

Modern supercomputers integrate CPUs, GPUs, AI accelerators, and energy-efficient architectures to support scientific research, machine learning, healthcare, climate science, and national security.

1.3 Evolution from Petascale to Exascale Computing

Petascale Computing refers to computing systems capable of performing 10^{15} (one quadrillion) floating point operations per second. These systems handle massive scientific simulations that are impossible on ordinary computers, consuming megawatts of power across thousands of interconnected processors.

Example: Roadrunner — IBM introduced in the year 2008

Exascale Computing was achieved in 2022 with Frontier — a 1000× leap over 14 years, driven by GPU acceleration, high-bandwidth memory, and advanced interconnects.

Exascale Computing

Exascale computing refers to systems capable of performing 10^{18} (one quintillion) floating point operations per second — written as 1 ExaFLOP/s. It is exactly 1000 times faster than petascale. Exascale systems represent the current frontier of supercomputing,

designed to tackle problems like whole-Earth climate modeling, full protein universe mapping, and real-time nuclear simulation — all at resolutions previously considered computationally impossible.

Example: Frontier (USA) was the world's first confirmed exascale computer, achieving 1.102 ExaFLOP/s. It uses AMD EPYC CPUs + AMD Instinct GPUs connected by HPE Cray Slingshot network.

2. Performance Metrics

All the components of a CPU core can operate at some maximum speed called peak performance. Whether this limit can be reached with a specific application code depends on many. Here we introduce some basic performance metrics that can quantify the speed of a CPU.

2.1 FLOPS — Floating Point Operations Per Second

Scientific computing tends to be quite centric to floating-point data, usually with “double precision” (DP). The performance at which the FP units generate results for multiply and add operations is measured in floating-point operations per second (Flops/sec). FLOPS is the standard unit for measuring HPC performance.

$$\text{FLOPS} = (\text{Number of cores}) \times (\text{Clock frequency}) \times (\text{FP operations per cycle})$$

2.2 Throughput vs Latency

Latency is the time taken to complete a single, individual task from start to finish. It is measured in units of time — milliseconds, microseconds, or nanoseconds.

For example when you click a link and wait for a page to load, what you feel is latency. Lower latency results in faster response and better performance.

Throughput is the number of tasks completed per unit of time. Throughput tells you how much total work a system can sustain over time. In HPC, throughput matters most for embarrassingly parallel workloads where thousands of independent jobs run simultaneously — each job's latency matters less because none of them wait on each other

Higher throughput is better for parallel workloads where thousands of independent jobs run simultaneously, because the overall number of completed jobs is more important than the latency of each individual job.

For an instance, consider a 200 km highway connecting two cities. A **6**-lane highway can carry 6,000 cars per hour is its **throughput** But any single car still takes **2 hours** to travel 200 km at 100 km/h — that is its **latency**

- Adding more lanes (more parallelism) increases throughput dramatically
- But the car's travel time (latency) only improves if you increase the speed limit a fundamentally different change

2.3 Speedup

Speedup is the fundamental measure of how much benefit parallelism actually delivers. Speedup is defined as the ratio of the time taken by the best sequential algorithm (T_sequential) to the time taken by the parallel version running on p processors (T_parallel). It answers the most basic question in parallel computing: was it worth parallelizing this program?

$$\text{Speedup (S)} = T_{\text{sequential}} / T_{\text{parallel}}$$

2.4 Efficiency

Computing efficiency measures how effectively a system utilizes resources to accomplish tasks. It is defined by the ratio of useful output to total resources consumed; ensuring maximum performance is achieved without unnecessary waste. Efficiency tells you how well you are utilizing the processors you have paid for.

$$\text{Efficiency (E)} = \text{Speedup} / \text{Number of processors} = S / p$$

Memory Architecture

Memory architecture refers to the organization and interconnection of different memory components in a computer system and the way they interact with the CPU to store, access, and retrieve data efficiently. It plays a crucial role in determining the overall performance of a computing system by reducing memory access time and improving data transfer between the processor and storage.

A typical memory hierarchy consists of the CPU, registers, cache memory, main memory (RAM), and secondary memory.

- Registers are the fastest storage locations and are used to hold data and instructions currently being processed by the CPU.
- Registers are the fastest storage locations and are used to hold data and instructions currently being processed by the CPU.
- RAM serves as the primary working memory where active programs and data are stored during execution.
- Secondary memory provides permanent storage for operating systems, applications, and user data.

The hierarchical organization of these components balances speed, capacity, and cost, making efficient memory management essential for high-performance computing.

Shared Memory Architecture

Shared memory architecture is a computing model in which all processors access a common main memory. Since every processor can directly read from and write to the same memory space, communication between processors occurs through shared variables stored in memory, resulting in low communication latency and simplified programming. This architecture is commonly used in symmetric multiprocessor (SMP) systems and multicore computers, where processors share the same physical memory.

Although shared memory systems provide fast communication and are relatively easy to program, their scalability is limited because increasing the number of processors leads to memory contention and bandwidth bottlenecks. Therefore, shared memory architecture is most suitable for small- to medium-scale parallel computing systems.

Key Concepts

- All processors share a single main memory.
- Any processor can access any memory location.
- Communication is through the shared memory.
- Suitable for small to medium scale systems (e.g., SMP systems).

Distributed Memory Architecture

Distributed memory architecture is a parallel computing model in which each processor has its own private local memory, and memory is not shared among processors. Since processors cannot directly access another processor's memory, they communicate by exchanging messages through a communication network using message-passing techniques. This architecture eliminates the memory contention found in shared memory

systems and allows the number of processors to be increased efficiently, making it highly scalable.

Distributed memory systems are widely used in clusters, supercomputers, and other high-performance computing environments where large-scale parallel processing is required.

Key Concepts

- Each processor has its own private (local) memory.
- Memory is not shared.
- Processors communicate by sending messages through the network.
- Suitable for large scale systems

Although programming is more complex because explicit communication between processors is necessary, distributed memory architecture provides excellent scalability and performance for computationally intensive scientific and engineering applications.