CSC 426 – PARALLEL COMPUTING

Parallel computing refers to the process of breaking down larger problems into smaller, independent, often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm.

Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time.,

There are several different forms of parallel computing:

bit-level, instruction-level, data, and **task parallelism**. Parallelism has long been employed in high-performance computing, but has gained broader interest due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.

What is High Performance Computing?

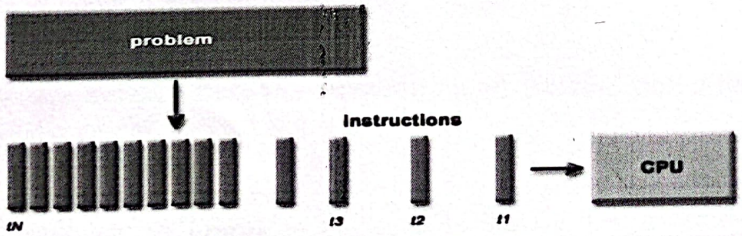
Odyssey supercomputer is the major computational resource of FAS RC:
- 2,140 nodes / 60,000 cores
- 14 petabytes of storage

Using the world's fastest and largest computers to solve large and complex problems.

Serial Computation:

Traditionally software has been written for serial computations:

- To be run on a single computer having a single Central Processing Unit (CPU)
- A problem is broken into a discrete set of instructions
- Instructions are executed one after another
- Only one instruction can be executed at any moment in time

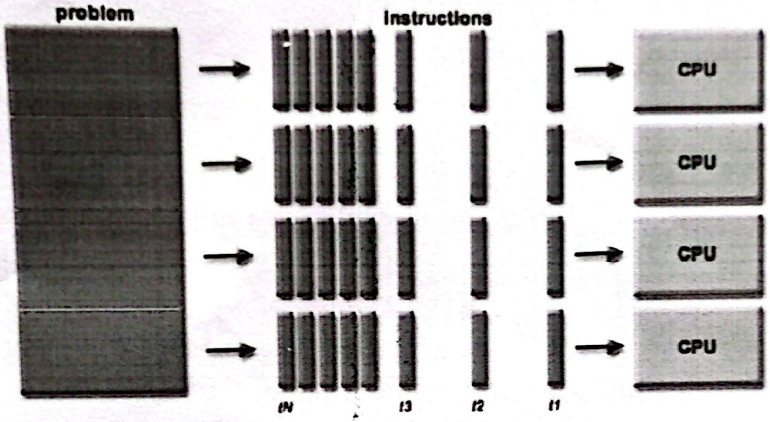


Parallel computing is a type of computing architecture in which several processors simultaneously execute multiple, smaller calculations broken down from an overall larger, complex problem.

Parallel Computing:

In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:

- To be run using multiple CPUs
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different CPUs



The primary goal of parallel computing is to increase available computation power for faster application processing and problem solving.

Parallel computing is closely related to concurrent computing—they are frequently used together though the two are distinct: it is possible to have parallelism without concurrency, and concurrency without parallelism (such as multitasking by time-sharing on a single-core CPU).

Parallel computing infrastructure is typically housed within a single Data Center where several processors are installed in a server rack; computation requests are distributed in small chunks by the application server that are then executed simultaneously on each server.

Why Parallel Computing?

There are various reasons why we need parallel computing, such are discussed below:

- Saving computation time.
- Solving larger and more challenging problems.
- access to more memory.
- Providing concurrency.
- Saving cost.

- Parallel computing deals with larger problems. In the real world, there are multiple things that run at a certain time but at numerous places simultaneously, which is difficult to manage. In this case, parallel computing helps to manage this kind of extensively huge data.
- Parallel computing is the key to make data more modeling, dynamic simulation and for achieving the same. Therefore, parallel computing is needed for the real world too.
- With the help of serial computing, parallel computing is not ideal to implement real-time systems; also, it offers concurrency and saves time and money.
- Only the concept of parallel computing can organize large datasets, complex, and their management.
- The parallel computing approach provides surety the use of resources effectively and guarantees the effective use of hardware, whereas only some parts of hardware are used in serial computation, and some parts are rendered idle.

Limitations of Parallel Computing:

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
- The algorithms must be managed in such a way that they can be handled in a parallel mechanism.
- The algorithms or programs must have low coupling and high cohesion. But it's difficult to create such programs.
- More technically skilled and expert programmers can code a parallelism-based program well.

Future of Parallel Computing

From serial computing to parallel computing, the computational graph has completely changed. Tech giants like Intel has already started to include multicore processors with systems, which is a great step towards parallel computing. For a better future, parallel computation will bring a revolution in the way of working the computer. Parallel Computing plays an important role in connecting the world with each other more than before. Moreover, parallel computing's approach becomes more necessary with multi-processor computers, faster networks, and distributed systems.

There are generally four types of parallel computing, available from both proprietary and open-source parallel computing vendors - bit-level parallelism, instruction-level parallelism, task parallelism, or superword-level parallelism:

1. **Bit-level parallelism**: The form of parallel computing in which every task is dependent on processor word size. In terms of performing a task on large-sized data, it reduces the number of instructions the processor must execute. There is a need to split the operation into series of instructions. For example, there is an 8-bit processor, and you want to do an operation on 16-bit numbers. First, it

must operate the 8 lower-order bits and then the 8 higher-order bits. Therefore, two instructions are needed to execute the operation. The operation can be performed with one instruction by a 16-bit processor. It increases processor word size, which reduces the quantity of instructions the processor must execute in order to perform an operation on variables greater than the length of the word.

2. **Instruction-level parallelism**: In a single CPU clock cycle, the processor decides in instruction-level parallelism how many instructions are implemented at the same time. For each clock cycle phase, a processor in instruction-level parallelism can have the ability to address that is less than one instruction. The software approach in instruction-level parallelism functions on static parallelism, where the computer decides which instructions to execute simultaneously. the hardware approach works upon dynamic parallelism, in which the processor decides at run-time which instructions to execute in parallel; the software approach works upon static parallelism, in which the compiler decides which instructions to execute in parallel.

3. **Task parallelism**: Task parallelism is the form of parallelism in which the tasks are decomposed into subtasks. Then, each subtask is allocated for execution. And, the execution of subtasks is performed concurrently by processors. a form of parallelization of computer code across multiple processors that runs several different tasks at the same time on the same data.
4. **Data-level parallelism (DLP)**: Instructions from a single stream operate concurrently on several data – Limited by non-regular data manipulation patterns and by memory bandwidth.

5. **Superword-level parallelism**: a vectorization technique that can exploit parallelism of inline code.

Parallel applications are typically classified as either **fine-grained parallelism**, in which subtasks will communicate several times per second; **coarse-grained parallelism**, in which subtasks do not communicate several times per second; or **embarrassing parallelism**, in which subtasks rarely or never communicate. Mapping in parallel computing is used to solve embarrassingly parallel problems by applying a simple operation to all elements of a sequence without requiring communication between the subtasks.

The popularization and evolution of parallel computing in the 21st century came in response to processor frequency scaling hitting the power wall. Increases in frequency increase the amount of power used in a processor, and scaling the