NAME : ORIMOGUNJE AYOOLA GIDEON

LEVEL : 400 LEVEL

DEPT : COMPUTER SCIENCE

MATRIC NO : DU0511

PARALLEL PROGRAMMING MODELS

Parallel Programming Model is a technique used to enhance the performance of software by dividing tasks into smaller sub-tasks that can be executed simultaneously. This approach leverages multiple processing elements to solve problems more efficiently and quickly

Parallel computing, also known as parallel programming, is a process where large compute problems are broken down into smaller problems that can be solved simultaneously by multiple processors.

The processors communicate using shared memory and their solutions are combined using an algorithm. Parallel computing is significantly faster than serial computing (also known as serial computation), its predecessor that uses a single processor to solve problems in sequence.

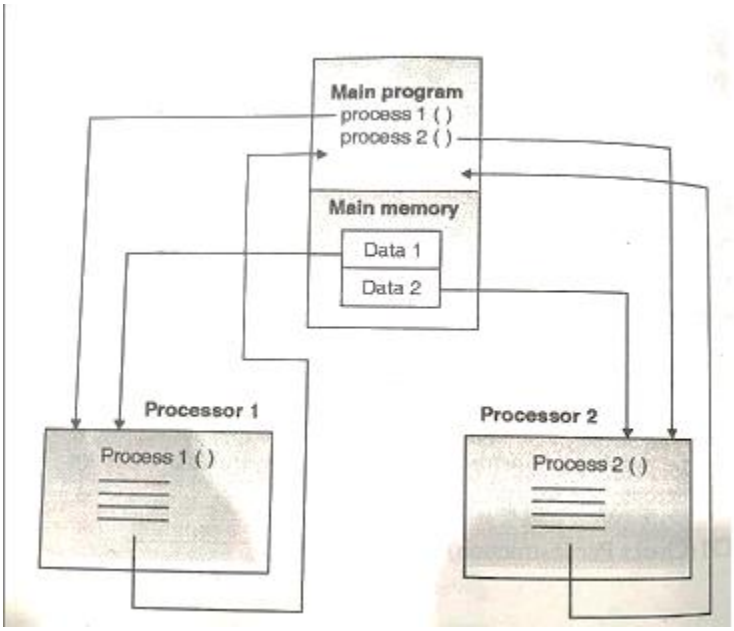
When computers were first invented in the late 1940s and 1950s, software was programmed to solve problems in sequence, which restricted processing speed. To solve problems faster, algorithms had to be built and implemented following a set of instructions on a central processing unit (CPU). Only after one instruction had been run might another one be solved.

Today, parallel systems have evolved to the point where they are used in various computers, making everyday tasks like checking email or sending a text message hundreds of times faster than if they were performed using serial computing. In addition to powering personal devices like laptops and smartphones, parallel systems also power the most advanced supercomputers and cutting-edge technologies like Artificial intelligence (AI) and the Internet of Things (IoT).

TYPES OF PARALLEL PROGRAMMING MODELS

1. Shared memory model
2. Message passing model
3. Threads model
4. Data parallel model

1. **Shared Memory Model:** In this type, the programmer views his program as collection of processes which use common or shared variables. The processor may not have a private program or data memory. A common program and data are stored in the main memory. This is accessed by all processors. Each processor is assigned a different part of the program and the data. The main program creates separate process for each processor. The process is allocated to the processor along with the required data. These process are executed indecently on different processors. After the execution, all the processors have to rejoin the main program.



Advantages

Program development becomes simple.

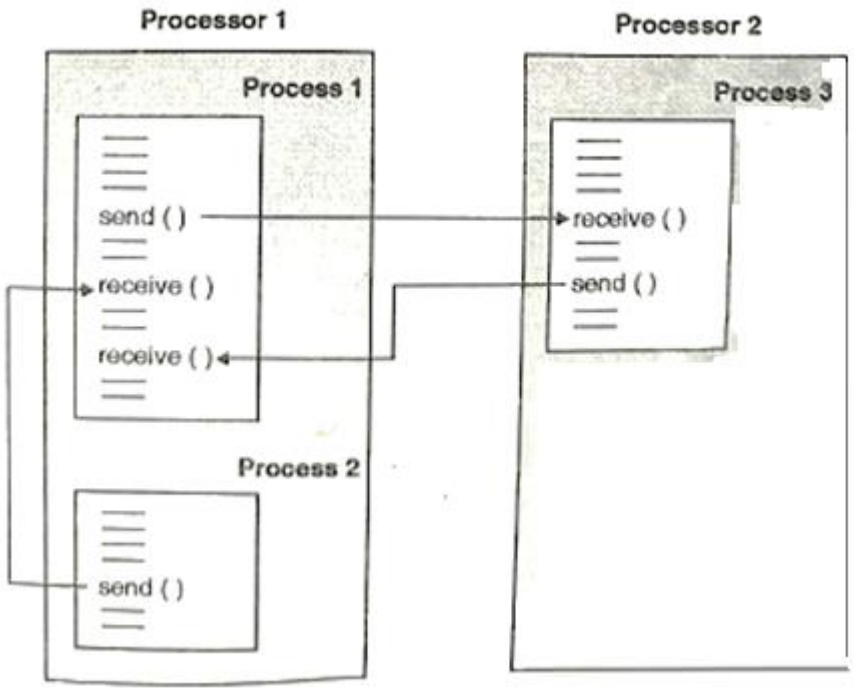
There is no need to explicitly specify the communication between data and process.

Disadvantages

Users will not understand where his variables use stored.

The native compiler present at each processor translates user variables into act addresses in main memory.

2. **Message Passing Model :**



In this type, different processes may be present on single multiple processors. Everyprocess has its own set of data

The data transfer between the processes is achieved by send and receive message requires co-operation between every process.

There must be a receive operation for every send operation.

Advantages

The data can be stored anywhere.

Communication between processes is simple.

Many Manage Passing Interfaces (MPS) are available.

Disadvantage

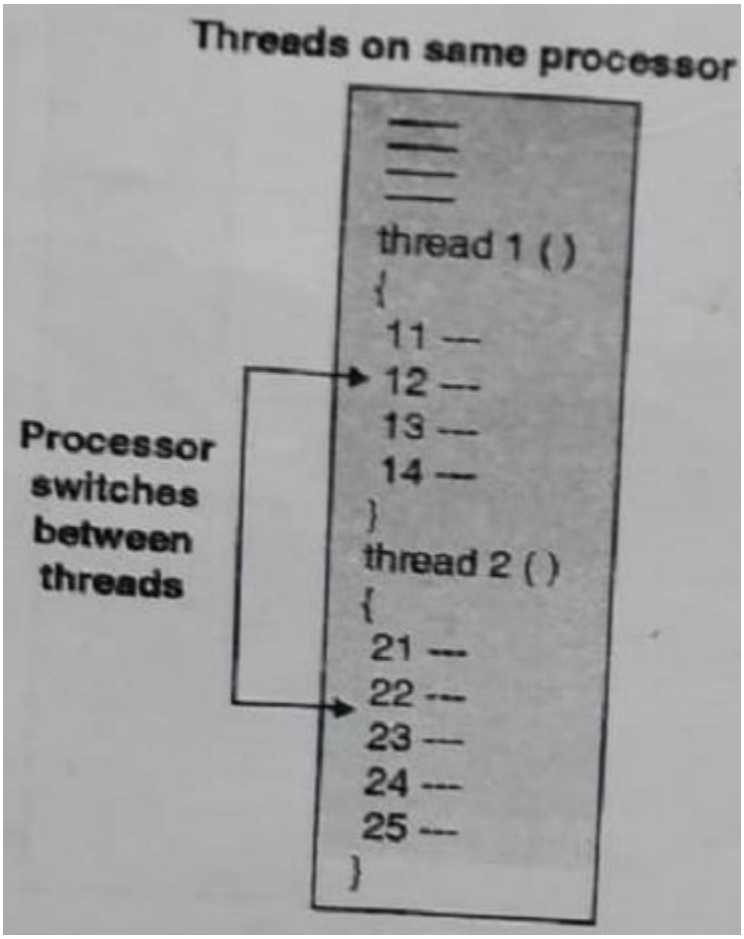
Programmers should take care that there is a receive function for every send function.

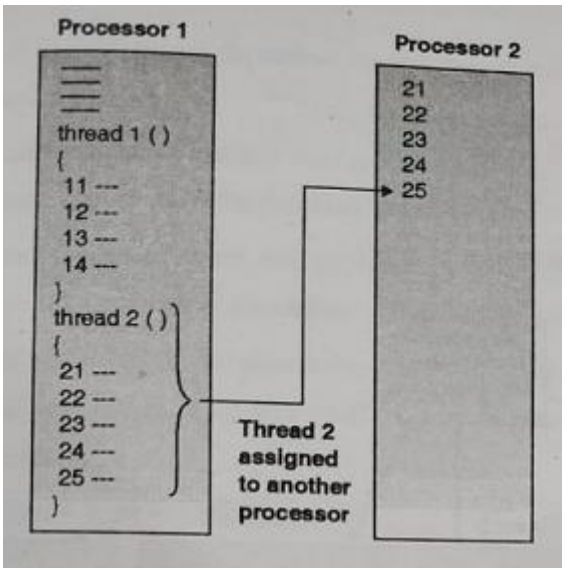
If any of the process, quite, others also will stop working, an they are dependent.

3. **Thread Models:** A thread is defined as a short sequence of instructions, within a process. Different threads can be executed on same processor or on different process.

If the threads are executed on same processor, then the processor switches between the threads in a random fashion.

If the threads are executed on different processors, they are executed simultaneously.





The threads communicate through global memory.

Advantages

Programmer need not have to worry about parallel processing.

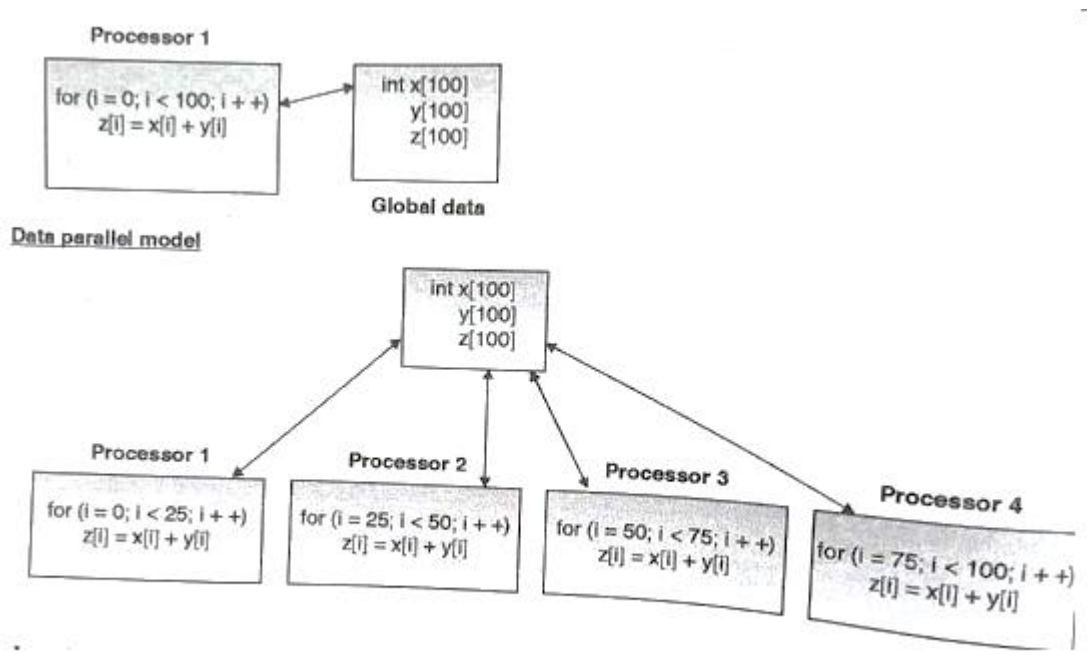
Disadvantages

Care must be taken that no two threads update the shared resources simultaneously.

4. **Data Parallel Model** :

   Data parallelism is one of the simplest form of parallelism. Here data set is organized into a common structure. It could be an array.Many programs apply the same operation on different parts of the common structure.

Suppose the task is to add two arrays of 100 elements store the result in another array. If there are four processor then each processor can do 25 additions.



Key Parallel Programming Models

1. **Message Passing Interface (MPI)**
    - MPI is a standardized and portable message-passing system designed for distributed computing.
    - It allows multiple processes running on different nodes in a cluster to communicate via message exchange.
    - Common functions include point-to-point communication, collective communication, and process synchronization.
    - MPI is widely used in scientific computing, simulations, and large-scale applications.

2. **OpenMP (Open Multi-Processing)**
    - OpenMP is an API that supports multi-platform shared-memory parallel programming in C, C++, and Fortran.
    - It uses compiler directives (e.g., #pragma omp parallel) to manage parallel regions in code.
    - OpenMP enables thread-level parallelism, where work is shared among multiple CPU cores dynamically.
    - It simplifies the parallelization of loops and sections within a program.

3. **MapReduce**
    - MapReduce is a programming model for processing large-scale data sets across distributed clusters.
    - It consists of two main phases:
        - **Map Phase**: Processes input data and generates key-value pairs.
        - **Reduce Phase**: Aggregates and processes key-value pairs to produce the final output.
    - Used in big data applications, including Hadoop-based ecosystems.
4. **OpenCL (Open Computing Language)**
    - OpenCL is an open standard for parallel computing across heterogeneous platforms, including CPUs, GPUs, and FPGAs.
    - It provides a uniform programming model to write portable code that executes across different hardware architectures.
    - OpenCL enables fine-grained control over memory management and computation for optimized performance.

5. **CUDA (Compute Unified Device Architecture)**
    - CUDA is a parallel computing framework developed by NVIDIA for GPU acceleration.

- It enables developers to write parallel applications using C, C++, and Fortran while leveraging the massive parallelism of GPUs.
- CUDA programming involves defining **kernels**, which are functions executed in parallel by multiple GPU threads.
- Used extensively in AI, deep learning, simulations, and high-performance computing.