24th January

Workshop on 31st January 2020 on Si5
Memory Models

Symmetric Multiprocessing
Symmetric multiprocessing (SMP) involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single, shared main memory, have full access to all input and output devices, and are controlled by a single operating system instance that treats all processors equally, reserving none for special purposes. Most multiprocessor systems today use an SMP architecture. In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors.

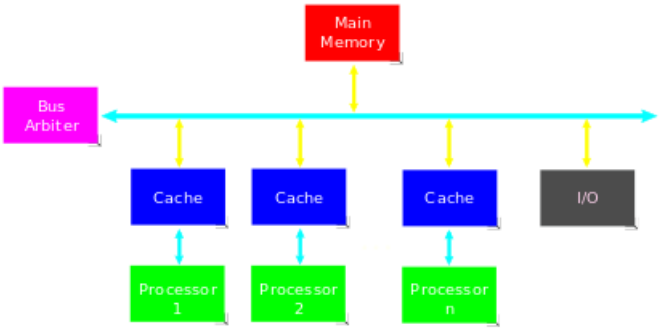

Diagram 1

Asymmetric multiprocessing

An asymmetric multiprocessing (AMP) system is a multiprocesso computer system where not all of the multiple interconnected central processing units (CPUs) are treated equally. For example, a system might allow (either at the hardware or operating system level) only one CPU to execute operating system code or might allow only one CPU to perform I/O operations. Other AMP systems might allow any CPU to execute operating system code and perform I/O operations, so that they were symmetric with regard to processor roles, but attached some or all peripherals to particular CPUs, so that they were asymmetric with respect to the peripheral attachment.

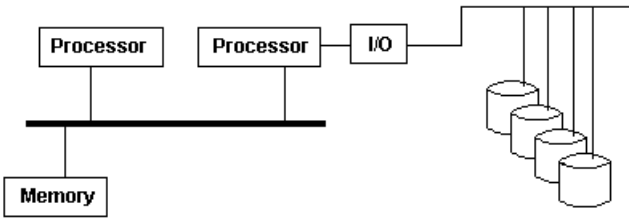

Diagram 2

- **Shared memory** – all cores see the same version of the memory across a single address space.
- Shared memory is more convenient for programmer as they dont have to worry about address space.
- In shared memory there is a single

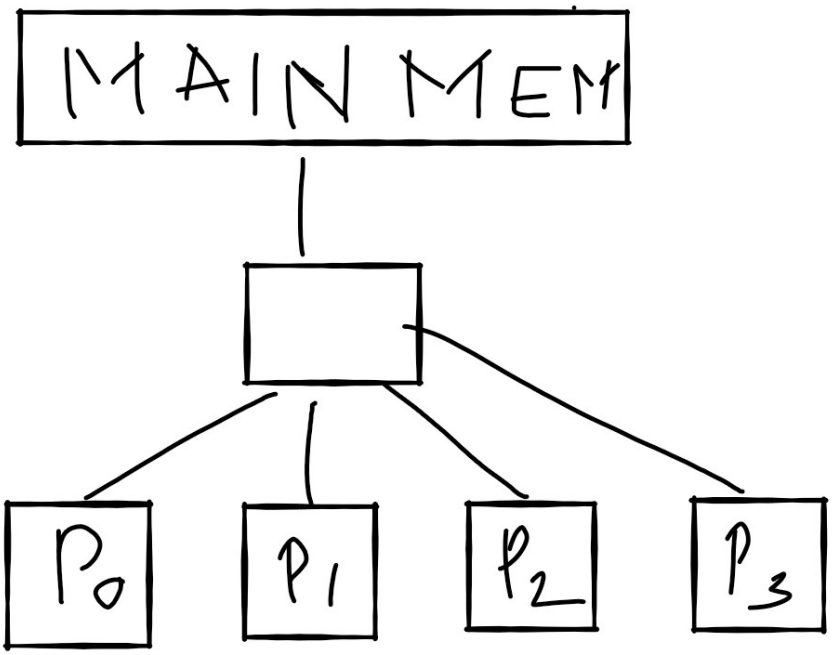

Diagram 3

- **Physically distributed shared memory** has physically distributed main memory among cores so that there is no contention for main memory as in the diagram 1. Also the core will be get more bandwidth compared to the shared memory access.

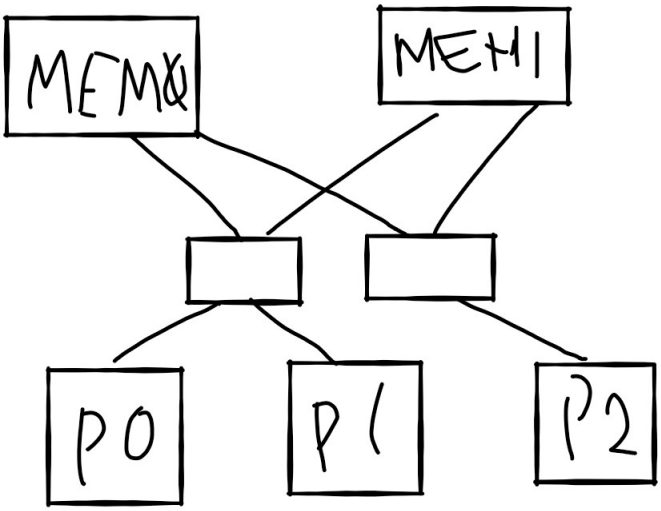

Diagram 4

- OS manages hardware resources.
- **NUMA(Non Uniform Memory access)**
  - In NUMA it has single address space

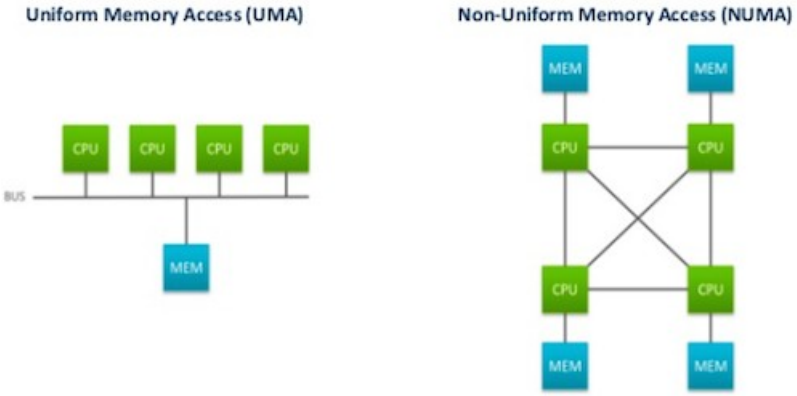

Diagram 5

- **Physically Distributed** – how the system is physically built, programmers don't need to care about the system
  - Multiple banks of memory
- **Logically distributed** – Programmers view
  - Multiple address spaces
  - Explicit communication for sharing data

Parallelism

- Exist in both operations that we have to carry out as well natural parallelism in data structure such as vectors and matrices.

Data Level parallelism

```
for(i = 0; i < N ; i++) // For N cores
        a[i] = 2 * b[i]
```

Instruction level parallelism
a = b + c
d = e + f
There is no data flow between these instructions, so they have high degree of parallelism