5)

in multiple-processor scheduling **multiple CPU's** are available and hence **Load Sharing** becomes possible. However multiple processor scheduling is more **complex** as compared to single processor scheduling. In multiple processor scheduling there are cases when the processors are identical i.e. HOMOGENEOUS, in terms of their functionality, we can use any processor available to run any process in the queue.

Approaches to Multiple-Processor Scheduling –

One approach is when all the scheduling decisions and I/O processing are handled by a single processor which is called the **Master Server** and the other processors executes only the **user code**. This is simple and reduces the need of data sharing. This entire scenario is called **Asymmetric Multiprocessing**.

A second approach uses **Symmetric Multiprocessing** where each processor is **self scheduling**. All processes may be in a common ready queue or each processor may have its own private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

Processor Affinity

- When a process has been running on a specific processor; The data most recently accessed by the process populates the cache for the processor. If the process migrates to another processor: The contents of cache memory must be invalidated for the current processor, and the cache for the new processor must be re-populated.
- Because of the high cost of invalidating and re-populating caches, most SMP systems try to avoid migration of processes from one processor to another and instead attempt to keep a process running on the same processor. This is known as **processor affinity.**

Load Balancing

- On SMP systems, it is important to keep the workload balanced among all processors to fully utilize the benefits of multi-processing.
- There are two general approaches to load balancing: **push migration** and **pull migration.** With push migration, a specific task periodically checks the load on each processor and if it finds an imbalance it evenly distributes the load by moving (or pushing) processes from overloaded to idle or less-busy processors.
- Pull migration occurs when an idle processor pulls a waiting task from a busy processor.
- Push and pull migration need not be mutually exclusive and are in fact often implemented in parallel on load-balancing systems.

Symmetric Multithreading

- Using SMT, a single physical processor emulates multiple processors by enabling multiple threads to issue instructions simultaneously during each cycle.
- A processor with SMT looks like multiple CPU to the OS. They are not true multi-core machine as SMT CPU shares most of the on-chip resources and contends for each other.
- Each logical processor has its own register set and instruction pipeline.
- Thread, unlike process, shares memory and page table entries which make them optimal to distribute across logical CPUs.