**4.1 : Overview**

**4.1.1 Motivation:**

Most software application that run on modern computer are multithreaded. Many similar task may be required to perform by server. It makes server heavier and slow to response with one thread.

Multithreaded is used in kernel of operating system and many application that require algorithm, graphic or artificial intelligence.

**4.1.2 Benefits:**

Responsiveness : The lighter service can be responsive immediately without waiting for executing.

Resource sharing : Thread shared memory by default

Economy: Thread share the resource of the process. Creating and context switching thread is more economical

Scalability : application can be developed broader and bigger.

**4.2 : Multicore Programming:**

Concurrency systems: deal with many task at once (the system can request, read book,…). If the computer have one CPU, then tasks must split time to deal each task at once in one CPU.

Parallelism systems: do many task at once.

**4.2.1 Programming Challenges:**

The designer of operating systems must write scheduling algorithm to allow parallel execution. 5 challenges:

Identify tasks: Examining task to find the independence task of another to run in parallel

Balance: Task must contribute as much value to overall process to get rid of unworthy.

Data splitting

Data dependency

Testing and debugging

**4.2.2 Types of parallelism:**

Data parallelism: distributing subset of the same data and performing the same operation on each core.

Task parallelism: Each of task perform the unique operation.

**4.3 Multithread Models :**

User Thread : provided at the user level, doesn’t be managed kernel support.

Kernel Thread : supported and managed directly by the operating system

**4.3.1 Many to one model :**

Map many user thread to one kernel thread( eg: Green thread )

**4.3.2 One to one model:**

Each user thread map to one kernel thread. The drawback is creating the user thread corresponding the new kernel thread. It make burden the performance of a system. ( Eg : Linux, windows )

**4.3.3 Many to many model:**

The developer need not to careful that creating many thread for application. However it hard to implement many to many model. Moreover, the increasing number of processing core that make the limiting kernel thread become less important.

**4.4 Thread Libraries :**

There are 2 types of approach for thread library : user space with no kernel support (Java thread), kernel-level library (POSIX thread and Windows thread).

There are 2 types of strategies for thread library:

**Asynchronous threading** : Parent create child thread, parent can do concurrently with child thread.

**Synchronous threading :** Parent create child thread and wait for child terminate before parent resume. Child threads can execute concurrently but the parent cannot.

**4.5 Implicit Threading :**

Nowadays, application contains hundred or even thousands of thread which make hard to the developer. Implicit Threading is transferring the creation and management of threading from application to compiler and run time libraries.

**Thread pools :** limit the thread that runs actively in system to get rid of exhausting system resources. The number of thread in pool can be based on number of CPUs, physical memory, and expected number of concurrent client requests.

**Fork join**: Fork join model, synchronous model: The main parent create child threads and then wait for the children terminate and join with its parent (combine the result).