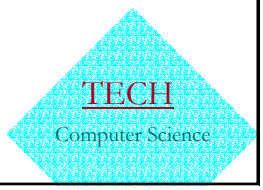
Introduction to Parallel Processing

- 3.1 Basic concepts
- 3.2 Types and levels of parallelism
- 3.3 Classification of parallel architecture
- 3.4 Basic parallel techniques
- 3.5 Relationships between languages and parallel architecture



3.1 Basic concepts

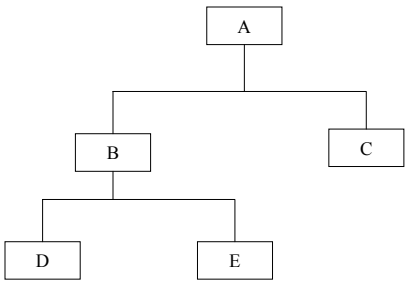
- 3.1.1 The concept of program
  - **ordered set of instructions (programmer's view)**
  - **executable file (operating system's view)**

The concept of process

- OS view, process relates to execution
- Process creation
  - **setting up the process description**
  - **allocating an address space**
  - **loading the program into the allocated address space, and**
  - **passing the process description to the scheduler**
- process states
  - **ready to run**
  - **running**
  - **wait**

Process spawning (independent processes)

- Figure 3.3 Process spawning.

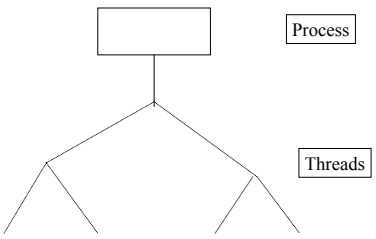


3.1.3 The concept of thread

- smaller chunks of code (lightweight)
- threads are created within and belong to process
- for parallel thread processing, scheduling is performed on a per-thread basis
- finer-grain, less overhead on switching from thread to thread

Single-thread process or multi-thread (dependent)

- **Figure 3.5 Thread tree**

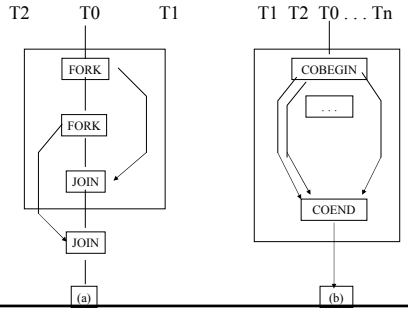


Three basic methods for creating and terminating threads

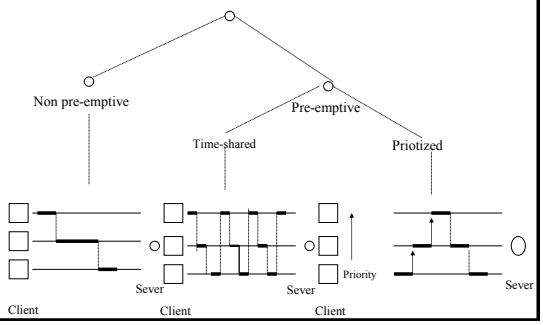
- 1. unsynchronized creation and unsynchronized termination
  - **calling library functions: CREATE_THREAD, START_THREAD**
- 2. unsynchronized creation and synchronized termination
  - **FORK and JOIN**
- 3. synchronized creation and synchronized termination
  - **COBEGIN and COEND**

3.1.4 Processes and threads in languages

- Black box view: T: thread

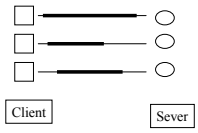


3.1.5 The concepts of concurrent execution (N-client 1-server)



Parallel execution

- N-client N-server model
- Synchronous or Asynchronous



Concurrent and parallel programming languages

- Classification

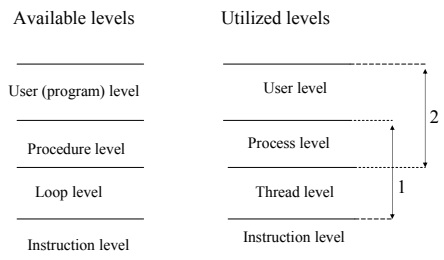
Table 3.1 Classification of programming languages.

Languages	1_client 1-server model	N_client 1-server mode	1_client N-server model	N_client N-server model
sequential	+	-	-	-
concurrent	-	+	-	-
Data-parallel	-	-	+	-
Parallel	-	+	-	+

3.2 Types and levels of parallelism

- 3.2.1 Available and utilized parallelism
  - **available: in program or in the problem solutions**
  - **utilized: during execution**
- 3.2.2 Types of available parallelism
  - **functional**
    - arises from the logic of a problem solution
  - **data**
    - arises from data structures

- Figure 3.11 Available and utilized levels of functional parallelism



1.Exploited by architectures
2.Exploited by means of operating systems

3.2.4 Utilization of functional parallelism

- Available parallelism can be utilized by
  - **architecture,**
    - instruction-level parallel architectures
  - **compilers**
    - parallel optimizing compiler
  - **operating system**
    - multitasking

3.2.5 Concurrent execution models

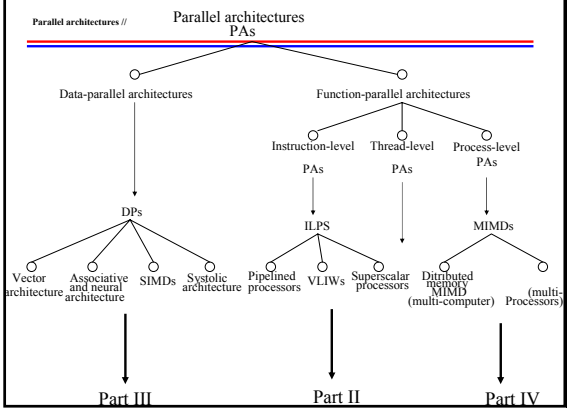
- User level --- Multiprogramming, time sharing
- Process level --- Multitasking
- Thread level --- Multi-threading

3.2.6 Utilization of data parallelism

- by using data-parallel architecture

3.3 Classification of parallel architectures

- 3.3.1 Flynn's classification
  - **SISD**
  - **SIMD**
  - **MISD (Multiple Instruction Single Date)**
  - **MIMD**



3.4 Basic parallel technique

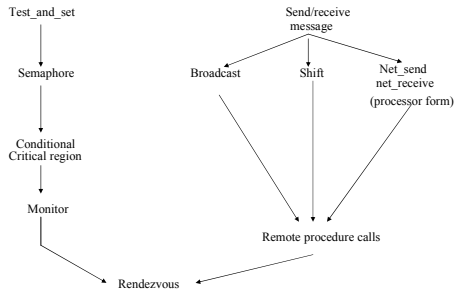
- 3.4.1 Pipelining (time)
  - **a number of functional units are employed in sequence to perform a** single **computation**
  - **a number of steps for each computation**
- 3.4.2 Replication (space)
  - **a number of functional units perform** multiply **computation simultaneously**
    - more processors
    - more memory
    - more I/O
  - **more computers**

3.5 Relationships between languages and parallel architecture

- SPMD (Single Procedure Multiple data)
  - **Loop: split into N threads that works on different invocations of the same loop**
  - **threads can execute the same code at different speeds**
  - **synchronize the parallel threads at the end of the loop**
    - barrier synchronization
  - **use MIMD**
- Data-parallel languages
  - **DAP Fortran**
    - C = A + B
  - **use SIMD**

Synchronization mechanisms

- Figure 3.17 Progress of language constructs used for synchronization



Using binary Semaphore p1 or queue semaphore p2

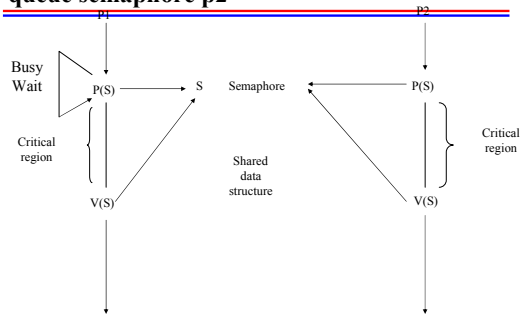


Figure 3.18 Using a semaphore to solve the mutual execution problem

Parallel distributed computing

- Ada
  - **used rendezvous concepts which combines feature of RPC and monitors**
- PVM (Parallel Virtual Machine)
  - **to support workstation clusters**
- MPI (Message-Passing Interface)
  - **programming interface for parallel computers**
- COBRA ?
- Windows NT ?

Summary of forms of parallelism

- See Table 3.3