Multithreading y Multiprocessing

Multithreading

- **Recursos compartidos:**
  - Heap,
  - Data Segment,
  - Code Segment (RO),
  - FDs.
- **Sincronización:**
  - Threading SO,
  - Threading runtime,
  - IPC.
- **Características:**
  - Fácil compartir información.
  - Alto **acoplamiento**.
  - **Baja estabilidad**. Thread que falla afecta a todos.
  - Escalabilidad muy limitada.

Multiprocessing

- **Recursos compartidos:** Code Segment (RO).
- **Sincronización:** IPCs.
- **Características:**
  - Complejo compartir información.
  - Componentes **simples** y **separados**.
  - +Escalable y +Estable.
  - Sin tolerancia a fallos.

Propiedades de Sistemas Distribuidos

- **Safety** *(siempre verdadera, "nada malo va a pasar")*:
  - Exclusión mutua.
  - Ausencia de deadlocks.
- **Liveness** *(eventualmente verdadera, "algo bueno va a pasar")*
  - Ausencia de starvation.
  - Fairness.

Asegurar safety: Concurrencia

- Basada en **Algoritmos**
  - Características:
    - Sin abstracciones especiales.
    - Condiciones lógicas simples p/ **Critical Sections**.
  - Técnicas:
    - **Busy-Waiting.** Problemas de performance (ej. spin-lock).
    - **Algoritmos de espera.**
- Basada en **Abstracciones**
  - Características:
    - Provistas por SOs.
    - Construir mecanismos compuestos por combinaciones.
  - Teécnicas:
    - **Operaciones atómicas.** Contadores atómicos, CAS *(Compare and Swap)*.

Mecanismos de Sincronización

- **Semáforos.**
- **Monitores.** Abstracción.
  - **Condition Variables.**
- **Barrera.**
  - Rendezvouz.

IPCs

Características

- Provistos por SO.
- **ABM excede vida del proceso.**
  - Responsabilidad del usuario.
- Identificados por nombre.
- Linux: diferentes tipos de archivo.

Modelos

- Signals.
- Shared Memory.
- File Locks.
- Pipes: unnamed pipes.
  - Jerarquía padre-hijo.
- Fifos: named pipes.
  - Dos procesos cualquiera.
  - Viven en el SO.
- Message Queues.
  - `mtype` identifica el tipo de mensaje.
- Sockets.

Problemas clásicos

- **Productor Consumidor.**
  - Situaciones de bloqueo:
    - Producir paquete con buffer lleno.
    - Consumir paquete con buffer vacío.
  - Acceso al buffer **debe ser sincronizado**.
  - Buffer acotado vs. infinito.
- **Lectores Escritores.**

Paralelización de tareas

- **Objetivos:**
  - Reducir **latencia** *(tiempo de cómputo de una tarea)*.
  - Incrementar **throughput**.
  - Reducir **potencia consumida**.
- **Camino crítico.** Máxima longitud de tareas secuenciales a computar.
- **Ley de Amdahl.** Todo computo se divide en fracciones secuenciales y paralelas.
  - `Tp = Wser + Wpar / P` con `P` unidades de cómputo.
  - Speedup máximo **acotado por fracción de tiempo no paralelizable** (`Smax <= 1/f`).
- **Ley de Gustafson.** Escalar el problema.
  - Parte serial disminuye -> +speedup.
  - Paralelismo aumenta -> +speedup.

Modelo Work-Span

- Provee cota inferior y superior para el speedup.
- Hipótesis:
  - **Paralelismo imperfecto.** No todo lo paralelizable se puede ejecutar al mismo tiempo.
  - **Greedy scheduling.** Proceso disponible == tarea ejecutada.
  - Despreciable:
    - Tiempo de **acceso a memoria**.
    - Tiempo de **comunicación entre procesos**.
- Resultados:
  - `T1`: tiempo en ejecutar operación con 1 proceso.
  - `Tinf`: tiempo en ejecutar su camino crítico (con infinitos procesos).
  - **Cota superior:** `min(P, T1 / Tinf)`
  - **Cota inferior:** `(T1 - Tinf) / P + Tinf`

Estrategias

- Descomposición Funcional.
- Particionamiento de Datos.

Patrones de procesamiento

- Fork-join.
- Pack.
- Split.
- Pipeline.
- Map.
- Reduction.