TEMA 1:

ARQUITECTURAS PARALELAS: CLASIFICACIÓN Y PRESTACIONES

Lección 2: CLASIFICACIÓN DE ARQUITECTURAS PARALELAS.

Computación paralela y computación distribuida.

Computación paralela: Estudia los aspectos hardware y software relacionados con el desarrollo y ejecución de aplicaciones en un sistema de cómputo compuesto por múltiples cores/procesadores/computadores que es visto externamente como una unidad autónoma (multicores, multiprocesadores, multicomputadores, cluster).

Computación distribuida: Estudia los aspectos hardware y software relacionados con el desarrollo y ejecución de aplicaciones en un sistema distribuido; es decir, en una colección de recursos autónomos situados en distintas localizaciones físicas.

- **Computación distribuida a baja escala:** Estudia los aspectos relacionados con el desarrollo y ejecución de aplicaciones en una colección de recursos autónomos de un dominio administrativo situados en distintas localizaciones físicas conectados a través de infraestructura de red local.
- **Computación distribuida a gran escala:**
    - **Computación grid:** Estudia los aspectos relacionados con el desarrollo y ejecución de aplicaciones en una colección de recursos autónomos de múltiples dominios administrativos geográficamente distribuidos conectados con infraestructura de telecomunicaciones.
    - **Computación nube o cloud:** Comprende los aspectos relacionados con el desarrollo y ejecución de aplicaciones en un sistema cloud.
        - Sistema cloud: Ofrece servicios de infraestructura, plataforma y/o software, por los que se paga cuando se necesitar (pay-per.use) y a los que se accede típicamente a través de una interfaz (web) de auto-servicio.
            - Consta de recursos virtuales que:
                - Son una abstracción de los recursos físicos.
                - Parecen ilimitados en número y capacidad y son reclutados/liberados de forma inmediata sin interacción con el proveedor.
                - Soportan el acceso de múltiples clientes.
                - Están conectados con métodos estándar independientes de la plataforma de acceso.

Clasificaciones de arquitecturas y sistemas paralelos.

Criterios de clasificación de computadores.

- Comercial.
    - Segmento del mercado.
- Educación, investigación (también usados por fabricantes y vendedores).
    - Flujos de control y flujos de datos.
    - Sistema de memoria.
    - Flujos de control (propuesta de clasificación de arquitecturas con múltiples flujos de control).
    - Nivel de paralelismo aprovechado.

Flujos de control y datos.

Clasificación de Flynn: divide el universo de computadores en cuatro clases según el número de secuencias o flujos de instrucciones y secuencias o flujos de datos que pueden procesarse simultáneamente en el computador. Está relacionada con el aprovechamiento del paralelismo a partir de la replicación de elementos, por lo que existen otros aspectos del paralelismo que no se reflejan bien.

- **SISD:** un único flujo de instrucciones procesa operandos y genera resultados, definiendo un único flujo de datos. Usa paralelismo funcional.
- **MISD:** se ejecutan varios flujos distintos de instrucciones, aunque todos actúan sobre el mismo flujo de datos.
- **SIMD:** un único flujo de instrucciones procesa operandos y genera resultados, definiendo varios flujos de datos, dado que cada instrucción codifica realmente varias operaciones iguales, cada una actuando sobre operadores distintos. Usa paralelismo de datos.
- **MIMD:** el computador ejecuta varias secuencias o flujos distintos de instrucciones, y cada uno de ellos procesa operandos y genera resultados definiendo un único flujo de instrucciones, de forma que existen también varios flujos de datos uno por cada flujo de instrucciones. Usa paralelismo funcional.

La taxonomía de Flynn pone en manifiesto dos tipos de paralelismo que puede aprovecharse según la plataforma de computo: paralelismo de datos y paralelismo funcional.

El paralelismo funcional se aprovecha cuando las funciones, bloques, instrucciones etc, que intervienen en la aplicación se ejecutan en paralelo. Se dispone el paralelismo funcional en:

- **Nivel de instrucciones u operaciones:** cuando se ejecutan en paralelo las instrucciones de un programa. Nivel de granularidad más fina.
- **Nivel de bucle:** cuando se ejecutan en paralelo distintas iteraciones de un bucle o secuencias de instrucciones de un programa. En este caso la granularidad es más gruesa que la del paralelismo entre instrucciones, se suele dar la granularidad fina-media.
- **Nivel de funciones:** los distintos procedimientos que constituyen un programa se ejecutan simultáneamente. Nivel de granularidad media.
- **Nivel de programas:** cuando la plataforma ejecuta en paralelo programas diferentes que pueden corresponder, o no, a la misma aplicación. Nivel de granularidad gruesa.

Flujos de control.

- Propuesta clasificación arquitecturas con múltiples flujos de control o threads.
    - TLP:
        - Implícito: flujos de control creados y gestionados por la arquitectura.
        - Explícito: flujos de control creados y gestionados por el SO.
            - Multiprocesadores, multicores, cores multithread con una instancia SO.
            - Multicomputadores con múltiples instancias SO.

Sistemas de memoria.

Se ha distinguido entre sistemas en los que todos los procesadores comparten el mismo espacio de direcciones y sistemas en los que cada procesador tiene su propio espacio de direcciones.

- **Multiprocesadores o sistemas con memoria compartida (SMP):** Todos los procesadores comparten el mismo espacio de direcciones, es inmediato pues, implementar una estructura física para multiprocesadores en la que los módulos de memoria se encuentren ubicados en la misma zona del sistema, separados de los procesadores por una red de interconexión que arbitre el acceso a estos módulos. El programador NO necesita conocer donde están almacenados los datos.
  Los multiprocesadores, también pueden tener centralizados los dispositivos con E/S. Con esta estructura, el tiempo de acceso de los procesadores a memoria, será igual sea cual sea la posición de memoria a la que acceden, es una estructura simétrica. Los multiprocesadores con esta característica se llaman SMP o multiprocesadores simétricos.
  El tamaño de las transferencias con memoria depende del hardware.

- **Multicomputadores o sistemas con memoria distribuida:** Cada procesador tiene su espacio de direcciones propio, es lógico que, en su estructura física, cerca de cada procesador haya un módulo de memoria local, en el que se encuentre su espacio de direcciones. El programador necesita conocer dónde están almacenados los datos.
  En esta configuración la red de interconexión se utiliza para la transferencia de datos compartidos entre nodos de la red. El tamaño de los mensajes a transferir depende del programador.

Comparativa de los multicomputadores y multiprocesadores:

- **Latencia en el acceso a memoria:** El tiempo de acceso a memoria es mayor en los multiprocesadores que en los multicomputadores. La mayor latencia por parte de los multiprocesadores se debe a la no localidad de los módulos de memoria, a la necesidad de acceder a memoria atravesando la red de interconexión y al incremento en la latencia media debido a conflictos en la red entre accesos de diferentes procesadores. Este problema influye en que el multiprocesador es poco escalable, al contrario que el Multicomputador que es un sistema que podría escalar incluso hasta miles de procesadores. (Escalable: si al añadir más recursos al sistema este incrementa sus prestaciones de forma proporcional al número de recursos utilizados).
- **Mecanismos de comunicación:** En un multiprocesador, ya que los procesadores comparten el espacio de direcciones, pueden acceder a contenidos de direcciones de memoria que otros han producido. En un Multicomputador un procesador no puede acceder al módulo de memoria local de otro procesador mediante un acceso a memoria de carga o almacenamiento. Se necesita implementar un mecanismo software que permita copiar datos entre módulos de memoria de diferentes procesadores a través de la red de interconexión. En este caso, el mismo dato quedará duplicado en el sistema tras la transferencia. Para hacer esta transferencia se usan send/receive.
- **Herramientas de programación:** Antes de ejecutar una aplicación en un multicomputador, hay que ubicar en la memoria local de cada procesador el código a ejecutar y los datos que este código utiliza (hay que distribuir la carga de trabajo entre los procesadores). En multiprocesadores, esta distribución no es necesaria, ya que todos los procesadores pueden acceder a cualquiera de los módulos de memoria del sistema.
- **Programación:** Se considera que para el programador la programación de un multiprocesador es más sencilla que la de un multicomputador, ya que no debe pensar en copiar datos entre nodos, ni tampoco en la asignación de trabajo a los procesadores. La programación es más sencilla en multiprocesadores para aquellos casos en los que los esquemas de comunicación son complejos o varían dinámicamente.

Comunicación uno-a-uno en un multiprocesador: Se debe garantizar que el flujo de control consumidor del dato lea la variable compartida cuando el productor haya escrito en la variable el dato.

Incremento de escalabilidad en multiprocesadores y red de interconexión:

- Aumentar caché del procesador.
- Usar redes de menor latencia y mayor ancho de banda en un bus (jerarquía de buses, barras cruzadas, multietapa).
- Distribuir físicamente los módulos de memoria entre los procesadores (pero se sigue compartiendo espacio de direcciones).

Clasificación completa de computadores según el sistema de memoria:

- **Multi-computadores (memoria no compartida):** NORMA.
- **Multi-procesadores (memoria compartida, único espacio de direcciones):**
    - **UMA** (multiprocesadores con acceso a memoria uniforme): El tiempo de acceso a los procesadores a una determinada posición de memoria principal es igual sea cual sea el procesador. Igualmente, el acceso a una posición de memoria en caché es igual para todos los procesadores.
    - **NUMA** (multiprocesadores con acceso a memoria no uniforme): El tiempo de acceso a una posición de memoria depende del procesador, ya que un procesador tardará menos tiempo en acceder al bloque de memoria local que al situado junto a otro procesador.
        - **NUMA:** Estas arquitecturas no incorporan hardware para evitar problemas por incoherencias entre cachés de distintos nodos. Esto hace que los datos modificables compartidos no se puedan trasladar de caché, sino que hay que acceder a ellos individualmente a través de red.
        - **CC – NUMA:** Estas arquitecturas tienen hardware para mantener coherencia entre cachés de distintos nodos, que se encarga de las transferencias de datos compartidos entre nodos.
        - **COMA:** En estas arquitecturas la memoria local de los procesadores se gestiona como caché. Permite recopilación y migración de bloques de memoria en función de su frecuencia de uso en los nodos. La migración de un bloque se prefiere cuando éste se usa en mayor medida por otro nodo diferente al nodo donde se encuentra. La réplica se prefiere cuando el bloque se usa en varios nodos.

Nivel de paralelismo aprovechado.

Arquitecturas con DLP (Data Level Parallelism): Ejecutan las operaciones de instrucción concurrente o en paralelo. Unidades funcionales vectoriales o SIMD.

Arquitecturas con ILP: Ejecutan múltiples instrucciones concurrentemente o en paralelo. Cores escalares segmentados, superescalares o VLIW/EPIC.

Arquitecturas con TLP explícito y una instancia de SO: Ejecutan múltiples flujos de control concurrente o en paralelo. Cores que modifican la arquitectura escalar segmentada, superescalar o VLIW/EPIC para ejecutar threads concurrentemente o en paralelo.

- Multiprocesadores: ejecutan threads en paralelo en un computador con múltiples cores.

Arquitecturas con TLP explícito y múltiples instancias SO: Ejecución múltiples flujos de control en paralelo.

- Multicomputadores: ejecutan threads un paralelo en un sistema con múltiples computadores.