Computación Paralela

Clasificación de Arquitecturas Paralelas y Redes de Interconexión

Alain Paul Herrera Urtiaga

Facultad de Ingeniería Estadística e Informática

May 19, 2025

Contenido

Clasificaciones de Arquitecturas Paralelas

- La forma en que clasificamos las arquitecturas paralelas nos ayuda a entender sus capacidades y limitaciones
- Existen varios esquemas de clasificación, entre ellos:
  - Taxonomía de Flynn (la más popular)
  - Modelos basados en la organización de la memoria
  - Modelos basados en la coordinación
- Cada clasificación resalta diferentes aspectos de las arquitecturas paralelas y nos ayuda a entender su funcionamiento

Taxonomía de Flynn

- Es la clasificación más conocida en computación paralela
- Utiliza ideas familiares al campo de la computación convencional
- Se basa en el análisis del flujo de instrucciones y de datos
- Utiliza dos variables:
  - n_i: número de flujos de instrucciones
  - n_d: número de flujos de datos
- Esto da origen a 4 tipos de máquinas

SISD (Single Instruction Stream, Single Data Stream)

- Representa la máquina clásica de Von-Neumann
- Un único programa es ejecutado usando un conjunto específico de datos
- Contiene: memoria central, unidad de control y unidad de procesamiento
- $n_i = n_d = 1$
- Componentes principales:
  - Memoria Central: almacena datos e instrucciones
  - Procesador:
    - Unidad de Control (UC): interpreta instrucciones
    - Unidad de Procesamiento (UP): ejecuta operaciones
- Soporta un tipo de paralelismo virtual a través del paradigma de multitareas
- Ejemplo: computadoras personales (PC)

SIMD (Single Instruction Stream, Multiple Data Stream)

- Arreglo de elementos de procesamiento que ejecutan la misma instrucción al mismo tiempo
- $n_i = 1$ y $n_d > 1$
- Enfoque de paralelismo utilizado: **paralelismo de datos**
- Componentes principales:
  - Memoria Central: almacena datos globales
  - Unidad de Control Central: controla todos los procesadores
  - Procesadores (UP1, UP2, ..., UPn): ejecutan misma instrucción sobre diferentes datos
- Procesadores conectados a través de una red
- La memoria puede ser compartida o distribuida
- Todos los procesadores trabajan con perfecta sincronización
- Ejemplo: CM-2

MISD (Multiple Instruction Stream, Single Data Stream)

- Computadoras con elementos de procesamiento, cada uno ejecutando una tarea diferente
- Los datos deben pasar a través de cada elemento de procesamiento
- $n_i > 1$ y $n_d = 1$
- Componentes principales:
  - Memoria Central: almacena datos
  - Unidad de Procesamiento (UP): recibe datos
  - Unidad de Control (UC): controla procesamiento
  - Etapas de procesamiento: Etapa 1, Etapa 2, ..., Etapa n
- Implementaciones reales poco comunes
- Algunos ejemplos: mecanismos de procesamiento por encauzamiento (pipelining) en procesadores RISC o sistemas de tolerancia a fallos
- Puede considerarse como un modo de procesamiento por encauzamiento

MIMD (Multiple Instruction Stream, Multiple Data Stream)

- Modelo más general de paralelismo
- Múltiples tareas heterogéneas pueden ejecutarse al mismo tiempo
- Cada procesador opera independientemente con ocasionales sincronizaciones
- Conjunto de elementos de procesamiento donde cada uno realiza una tarea independiente o no
- $n_i > 1$ y $n_d > 1$
- Componentes principales:
  - Memoria Central: puede ser compartida o distribuida
  - Red de Interconexión: conecta procesadores y memoria
  - Múltiples procesadores, cada uno con:
    - Unidad de Control (UC): opera independientemente
    - Unidad de Procesamiento (UP): ejecuta operaciones
- Sistemas de multiprocesadores y sistemas de múltiples computadores

Tipos de MIMD

- Las MIMD se pueden distinguir entre:
  - **Sistemas con múltiples espacios de direcciones:** Comunicación explícita mediante pase de mensajes (NUMA)
  - **Sistemas con espacios de direcciones compartidos:** Modelo UMA con memoria centralizada
- Esto da lugar a:
  - MIMD a memoria distribuida
  - MIMD a memoria compartida
- En estas arquitecturas aparece el problema de gestión del acceso concurrente a la información compartida
- Esto origina cierto no determinismo en las aplicaciones

Modelo Basado en la Organización de la Memoria

- La clasificación de Flynn no permite distinguir la organización interna de la memoria
- Esta clasificación combina los modos de funcionamiento paralelos (SIMD y MIMD) con los tipos de organización de la memoria:
  - Memoria Compartida (SM)
  - Memoria Distribuida (MD)
- Esto da origen a diferentes tipos de arquitecturas con características específicas

SIMD-Memoria Compartida

- Características:
  - Control centralizado y datos centralizados
  - Máquinas vectoriales monoprocesadores con encauzamiento
- Funcionamiento:
  - Realiza una única operación sobre un conjunto de datos múltiples
  - Las operaciones se realizan de manera secuencial pero con encauzamiento
- Similar a la arquitectura SIMD básica, pero con énfasis en la memoria compartida centralizada

SIMD-Memoria Distribuida

- Características:
  - Control centralizado y datos distribuidos
  - Procesadores de bajo poder computacional
  - Gran número de procesadores
- Componentes:
  - Unidad de Control (UC): central y única
  - Múltiples procesadores (UP) cada uno con:
  - Memoria local pequeña
- Cada procesador:
  - Recibe instrucciones de una unidad de control central
  - Ejecuta la misma instrucción que los demás de manera sincronizada

MIMD-Memoria Compartida

- Características:
  - Conjunto de procesadores que comparten la misma memoria
  - Cada procesador puede ejecutar instrucciones diferentes
  - Acceso a memoria a través de red de interconexión
  - La memoria puede dividirse en varios bancos
- Componentes:
  - Memoria Central: compartida entre todos los procesadores
  - Red de Interconexión: conecta procesadores y memoria
  - Múltiples procesadores, cada uno con UC y UP propias
- Ventajas:
  - Fácil portar código secuencial para su ejecución
  - Pocos pero poderosos procesadores
- Desventajas:
  - Complejidad de la red aumenta con el número de procesadores
  - Limitado a un número reducido de procesadores

MIMD-Memoria Distribuida

- Características:
  - Control distribuido y distribución de datos
  - Cada procesador es autónomo (como máquina secuencial)
  - Cada procesador tiene su memoria local
  - Comunicación por red de interconexión mediante pase de mensajes
- Componentes:
  - Red de Interconexión: conecta computadores entre sí
  - Múltiples computadores, cada uno con:
    - Memoria propia
    - Procesador propio
- Ventajas:
  - Puede explotar ventajas de diferentes arquitecturas
  - Gran número de procesadores, pueden ser de gran poder
  - Escalabilidad

MIMD-Memoria Distribuida-Compartida

- Combinación de arquitecturas de memoria distribuida y compartida
- Objetivos:
  - Mantener acceso a memoria uniforme
  - Evitar problemas de coherencia de memoria caché
  - Facilitar la programación con espacio de direcciones globales
  - Mantener la facilidad de construcción de máquinas distribuidas
- Componentes:
  - Red de Interconexión: conecta múltiples nodos
  - Cada nodo contiene:
    - Memoria compartida local
    - Múltiples procesadores (P0, P1, ..., Pn)
- Funcionamiento:
  - Un procesador puede accesar toda la memoria como espacio uniforme
  - Pase de mensajes oculto cuando se accede a memoria remota
  - Memoria virtualmente compartida
  - Accesos remotos toman más tiempo que locales

Extensiones a la Taxonomía de Flynn

- Arquitectura MPMD (Multiple Program, Multiple Data)
  - Extensión del modelo MIMD
  - En vez de instrucciones, son múltiples programas ejecutados en diferentes sitios
  - Algunos programas pueden ser copias de otros
- Arquitectura SPMD (Single Program, Multiple Data)
  - Derivada del modelo SIMD
  - Completamente asíncrono
  - Una sola copia de un programa ejecutada sobre diferentes datos
  - Diferentes ramificaciones del programa se ejecutan en cada sitio
  - Útil cuando los datos no son regulares

Modelo Basado en la Coordinación

- Existen dos métodos generales de coordinación:
  - **Síncrono:** Todas las operaciones son ejecutadas al mismo tiempo, se trata de hacer cumplir las dependencias en órdenes de ejecución
  - **Asíncrono:** No existe tal restricción
- Esta clasificación da origen a diferentes tipos de arquitecturas

Arquitecturas Síncronas

- Tipos:
  - Computador secuencial
  - Arquitecturas encauzadas (pipelined)
  - SIMD
  - Máquinas sistólicas
- La sincronización permite garantizar dependencias entre operaciones
- Útiles para aplicaciones donde la regularidad de datos y operaciones es alta

Arquitecturas Encauzadas (Pipelining)

- Divide las operaciones en pasos ejecutados uno tras otro
- La entrada de una fase es la salida de la precedente
- Descompone operación F en conjunto de k operaciones (fases): $F = F_1, \ldots, F_k$
- Cada dato transita por las k etapas en un orden específico
- Si tenemos n instrucciones, se necesitan $n + k - 1$ ciclos de reloj
- En máquinas sin encauzamiento se necesitarían $k \times n$ ciclos

Ejemplo de Encauzamiento

Consideremos la adición de dos números de punto flotante en cuatro fases:

Reloj	**Fase Ejecutada**				**Observación**
1	F_1 de I_1				
2	F_1 de I_2	F_2 de I_1			
3	F_1 de I_3	F_2 de I_2	F_3 de I_1		
4	F_1 de I_4	F_2 de I_3	F_3 de I_2	F_4 de I_1	I_1 termina
...	...	...	...	...	...
n	F_1 de I_n	F_2 de I_{n-1}	F_3 de I_{n-2}	F_4 de I_{n-3}	I_{n-3} termina
$n+1$		F_2 de I_n	F_3 de I_{n-1}	F_4 de I_{n-2}	I_{n-2} termina
$n+2$			F_3 de I_n	F_4 de I_{n-1}	I_{n-1} termina
$n+3$				F_4 de I_n	I_n termina

Esquema de Encauzamiento

Proc./Tiempo	T_0	T_1	T_2	T_3	
P2			$F_2(F_1(F_0(d_0)))$	$F_2(F_1(F_0(d_1)))$	
P1		$F_1(F_0(d_0))$	$F_1(F_0(d_1))$	$F_1(F_0(d_2))$	
P0	$F_0(d_0)$	$F_0(d_1)$	$F_0(d_2)$	$F_0(d_3)$	

- Entrada: $d_0, d_1, d_2, d_3, d_4, ...$
- Salida: $F_2(F_1(F_0(d_0))), F_2(F_1(F_0(d_1))), ...$
- Cada procesador (P0, P1, P2) realiza una fase del procesamiento
- Los datos fluyen a través de las etapas del encauzamiento

Ejemplo de Encauzamiento

$f(x, y) = (x + y)^{3/2}$

- Subprocesos: añadir $x + y$, multiplicar por 3, hallar raíz cuadrada
- A cada subproceso se le asigna una unidad funcional
- La salida de una unidad se conecta a la entrada de la siguiente
- En el momento 1: x_1 y y_1 son sumados en la unidad 1, dando z_1
- En el momento 2:
  - z_1 se multiplica por 3 en la unidad 2, dando w_1
  - En paralelo, x_2 e y_2 son sumados en la unidad 1, dando z_2
- El proceso continúa con las tres unidades procesando diferentes variables

Ejercicio 1

Enunciado: Desarrolla un programa en Python que implemente un pipeline de procesamiento para transformar una serie de valores de entrada. El pipeline debe constar de tres etapas:

1. **Etapa 1:** Convertir el dato a entero y multiplicarlo por 2
2. **Etapa 2:** Elevar al cuadrado el resultado de la etapa anterior
3. **Etapa 3:** Convertir el resultado a string y añadir el texto "Resultado: " como prefijo

Implementa dos versiones:

- Una versión secuencial donde cada dato pasa por todas las etapas antes de procesar el siguiente dato
- Una versión de pipeline donde cada etapa procesa todos los datos antes de pasar a la siguiente etapa

Compara los resultados de ambas implementaciones para verificar que son idénticos.

Datos de entrada: ["1", "2", "3", "4"]

Salida esperada: ["Resultado: 4", "Resultado: 16", "Resultado: 36", "Resultado: 64"]

Ejercicio 2

Enunciado: Crea un sistema de encauzamiento (pipeline) en Python para procesar una lista de números a través de las siguientes etapas:

1. **Etapa 1:** Duplicar el número
2. **Etapa 2:** Sumar 10 al resultado de la etapa anterior
3. **Etapa 3:** Calcular el cuadrado del resultado de la etapa anterior
4. **Etapa 4:** Dividir el resultado por 5

Implementa el procesamiento de dos formas:

- Un enfoque secuencial tradicional donde cada número pasa por todas las etapas antes de procesar el siguiente
- Un enfoque de pipeline donde cada etapa procesa todos los números antes de pasar a la siguiente etapa

El programa debe mostrar el resultado de cada transformación y verificar que ambos métodos producen resultados idénticos.

Datos de entrada: [3, 7, 2, 9, 5]

Comparativa de Modelos de Procesamiento

Serial	Encauzado	Paralelo
1	1 2 3	1 2 3
2	1 2 3	...
3	1 2 3	...
1	1 2 3	...
2	1 2 3	...
3	...	...
...	...	...

- **Serial**: Las operaciones 1, 2, 3 se ejecutan secuencialmente para cada conjunto de datos
- **Encauzado**: Las operaciones se ejecutan en etapas, cada etapa procesa diferentes datos simultáneamente
- **Paralelo**: Múltiples procesadores ejecutan todas las operaciones sobre diferentes datos al mismo tiempo

Máquinas Sistólicas

- Multiprocesador encauzado
- Datos distribuidos desde memoria a arreglos de vectores
- Mezcla de máquina SIMD y encauzada
- Eficiencia depende de la regularidad de datos y fases
- Se adapta bien al paralelismo de grano fino
- Basado en paralelismo de datos
- Componentes:
  - Malla de procesadores
  - Conexiones en una estructura regular
  - Datos "bombeados" a través de la estructura
  - Cada procesador realiza operaciones simples
- Ejemplo: multiplicación de matrices

Arquitecturas Asíncronas

- Ejemplo clásico: máquinas MIMD
- Cada procesador opera independientemente de los otros
- Intercambios de información:
  - A través de memoria compartida (SM)
  - Enviando mensajes (DM)
- Permiten mayor flexibilidad para diferentes tipos de aplicaciones
- Mayor complejidad en la sincronización y comunicación

Comunicación en Sistemas Paralelos

- Las máquinas paralelas replican componentes: UP, UC, etc.
- Las tareas rara vez son completamente independientes
- Dependencia de tiempo: un procesador P1 debe terminar T1 antes que P2 inicie T2
- Formas de comunicación:
  - Compartiendo recursos: datos en memoria compartida
  - Transmisión de información: envío de mensajes
- Ambos casos requieren redes de comunicación
- El tamaño del sistema determina la importancia de la red de interconexión

Pase de Mensajes

- Conjunto de procesos con memoria y datos propios
- Datos intercambiados mediante mensajes
- El mensaje:
  - Se origina en un sitio (procesador emisor)
  - Se transmite a través de la red de interconexión
  - Se recibe en otro lugar (procesador receptor)
- La red actúa como servidor
- Eficiencia depende de ubicación de emisor y receptor
- Librerías: PVM, MPI, etc.

Requisitos del Sistema de Pase de Mensajes

- Cada procesador debe:
  - Conocer su dirección y la del resto de procesadores
  - Poder crear y enviar mensajes
  - Poder recibir mensajes, remover datos y manipularlos
- Operaciones esenciales:
  - **Rutina de enviar**: transmitir mensaje al nodo destino
  - **Rutina de recepción**: recibir mensajes

Operaciones de Envío y Recepción

Rutina de enviar:

Enviar(buzón, tamaño, destinatario, tipo)

- **Buzón**: dirección inicial en memoria donde está la información a enviar
- **Tamaño**: tamaño del mensaje
- **Destinatario**: a quien se enviará
- **Tipo**: tipo de mensaje (clasificación)

Rutina de recepción:

Recibir(buzón, tamaño, expedidor, tipo)

- **Buzón**: dirección inicial en memoria donde se almacenará la información
- **Tamaño**: tamaño del mensaje
- **Expedidor**: de quien se recibe
- **Tipo**: tipo de mensaje (clasificación)

Modos de Comunicación

- **Comunicación síncrona (rendez-vous)**:
  - Emisor y receptor operan al mismo tiempo
  - Similar a comunicación telefónica
  - Puede implicar tiempos de espera largos
  - Asegura al emisor que el mensaje llegará
- **Comunicación asíncrona**:
  - Operaciones realizadas en cualquier instante
  - Similar a enviar una carta
  - Emisor no debe esperar
  - Receptor quizás deba esperar si no ha llegado el mensaje
  - Menor tiempo de espera
  - Emisor nunca sabe si el receptor recibe el mensaje

Modos de Bloqueo

- **Comunicación con bloqueo**:
  - Al llamar rutinas de envío/recepción, terminan cuando se completan
  - Operación de envío termina cuando mensaje sale del emisor
  - Operación de recepción termina cuando mensaje llega y se copia
- **Comunicación sin bloqueo**:
  - Disminuye tiempo de espera
  - Capa de software subyacente maneja envío/recepción
  - Programa continúa su ejecución
  - Permite solapar comunicación y cálculo
  - Programador debe verificar que operaciones terminen a tiempo
  - Debe verificar respeto a dependencias

Comunicación por Interrupción

- Válido solo en modo asíncrono
- Basado en programación por evento
- Cuando llega un mensaje, genera interrupción en la aplicación
- Declarada con:

  IRecibir(buzón, tamaño, expedidor, tipo, procedimiento)
- Al llegar mensaje:
  - Ejecución se interrumpe
  - Mensaje se coloca en buzón
  - Control pasa al procedimiento de interrupción
  - Al terminar, se retoma ejecución normal
- Poco utilizado en la práctica

Primitivas de Comunicación

- **Transferencia**: envío de mensaje de un procesador a otro
- **Sincronización**: asegura que todos los procesos lleguen a un punto dado
- **Difusión**: envío del mismo mensaje desde un emisor a todos los demás
- **Distribución**: difusión personalizada (mensajes distintos a cada procesador)
- **Agrupamiento**: operación inversa de distribución
- **Comunicación total**: todos comparten información con todos
- **Transposición**: cada procesador realiza una distribución simultáneamente
- **Reducción**: sintetiza en un procesador un dato a partir de los datos de todos

Exclusión Mutua

- En máquinas de memoria compartida, procesadores se comunican a través de la memoria
- Problema: control de acceso para evitar condición de competencia
- **Condición de competencia**: dos procesadores leen/escriben simultáneamente la misma dirección
- Solución: exclusión mutua
- **Sección crítica**: parte de código donde se accede a memoria compartida
- Garantiza que cuando un proceso usa un espacio de memoria nadie más lo esté usando

Mecanismos de Exclusión Mutua

- Desarrollados a nivel de lenguajes y sistemas operativos:
  - Desactivación de interrupciones
  - Alternancia estricta entre procesos
  - Contadores de eventos
  - Monitores
  - Semáforos
- **Semáforos**: los más usados en máquinas de memoria compartida
  - Valor ≥ 0 cuando no hay procesos esperando
  - Valor negativo igual al número de procesos en espera
  - Operaciones atómicas: `DOWN` y `UP`

Operaciones de Semáforos

Operación DOWN:

- Verifica si el valor del semáforo es > 0
- Si es así, lo disminuye y continúa
- Si no, se bloquea (espera)

Operación UP:

- Incrementa el valor del semáforo
- Si hay procesos bloqueados, desbloquea a uno
- El proceso desbloqueado termina su operación DOWN

Con estos mecanismos, se puede controlar el acceso a secciones críticas fácilmente

Usos de la Exclusión Mutua

- Control de acceso a secciones críticas
- Sincronización de procesos
- Manejo de concurrencia en sistemas de memoria compartida
- Evitar condiciones de carrera
- Asegurar consistencia en la actualización de datos compartidos

Redes de Interconexión

- Clasificación según la conexión:
  - **Directa**: entre procesadores (típica en memoria distribuida, red estática)
  - **Indirecta**: a través de bancos de memoria compartidos (red dinámica)
- Rol diferente según arquitectura:
  - **Memoria compartida**: enlazar bancos de memoria con procesadores
  - **Memoria distribuida**: enlazar procesadores entre sí

Diferencias entre Redes

- **Ancho de Banda**:
  - Cantidad de información transmitida por unidad de tiempo
  - **Memoria compartida**: debe ser grande para alimentar procesadores
  - **Memoria distribuida**: magnitudes más pequeñas
- **Latencia de la red**:
  - Tiempo para realizar una transferencia
  - **Memoria compartida**: debe ser pequeña (caminos cortos y rápidos)
  - **Memoria distribuida**: menos crítica (pueden tener caminos largos)
- **Información a transmitir**:
  - **Memoria compartida**: datos, instrucciones, direcciones
  - **Memoria distribuida**: principalmente datos

Redes de Interconexión para Máquinas a Memoria Compartida

- Se distinguen por el número de etapas para interconectar procesadores con bancos de memoria
- Utilizan cajas de intercambios o conmutadores (switches)
- Características de los conmutadores:
  - Número b de entradas/salidas (frecuentemente $b = 2$ o $b = 4$)
  - Cada red tiene capas o etapas
  - Cada etapa tiene n/b conmutadores en paralelo
  - Número de etapas: $\log_b(n)$
- Son redes dinámicas: la topología puede variar durante la ejecución

Tipos de Redes a Memoria Compartida

- **Redes de 0 Etapa (Bus)**:
  - Una sola memoria compartida por todos los procesadores
  - Red de interconexión compartida entre unidades
  - Accesos a memoria secuenciales
  - El bus es un cuello de botella
- **Redes a 1 Etapa (Crossbar)**:
  - Todas las entradas atraviesan todas las líneas de salida
  - Conmutador en cada punto de interconexión
  - Latencia depende del número de procesadores y bancos
  - La complejidad crece rápidamente: $m \times n$ enlaces
- **Redes Multietapas**:
  - Objetivo: acercarse al comportamiento de crossbar usando menos conmutadores
  - Estructura en forma de tablas rectangulares
  - Requieren $n \times \log_2(n)$ conmutadores