Arquitectura de Computadores II
Clase #17

Facultad de Ingeniería
Universidad de la República

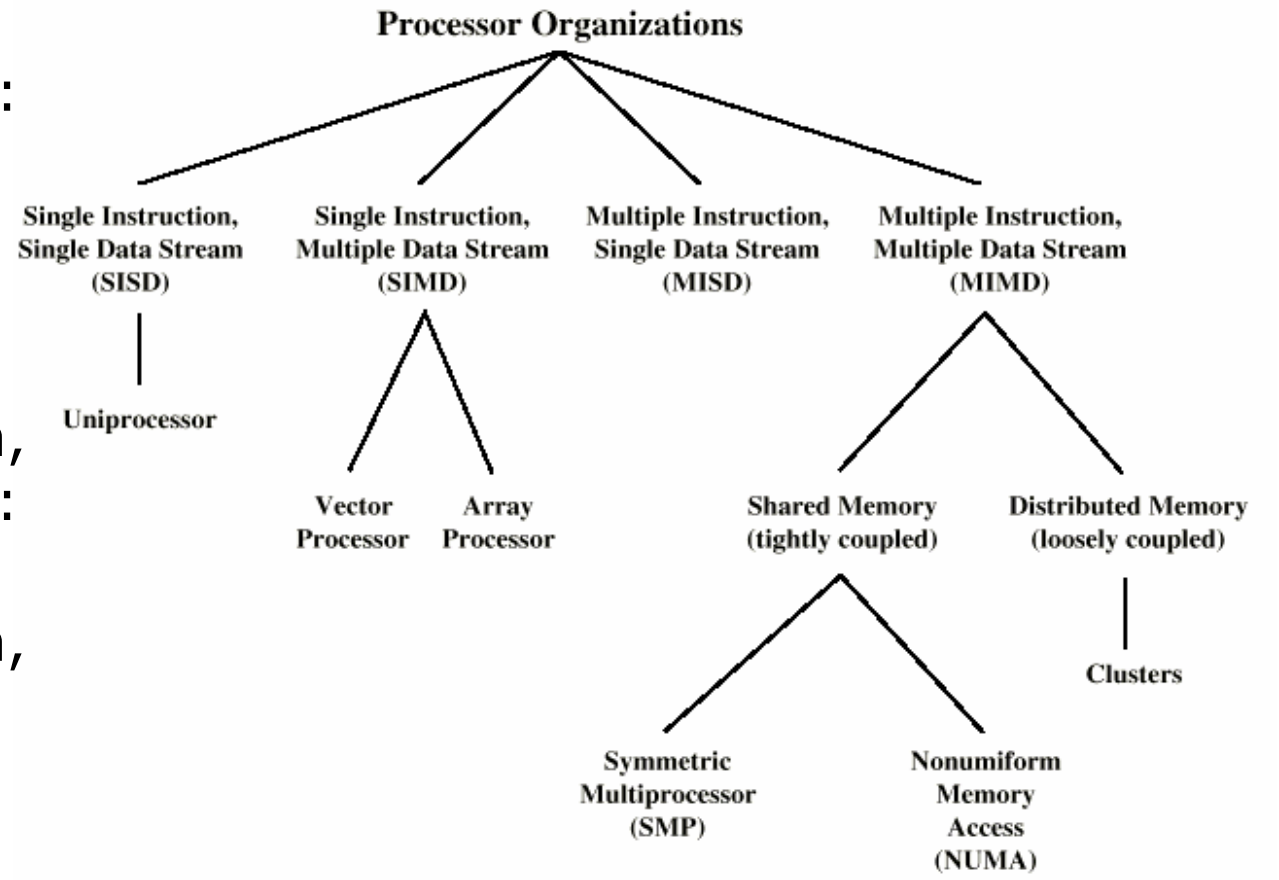
Instituto de Computación
Curso 2010

Veremos

- Arquitecturas Paralelas

Taxonomía de Arquitecturas Paralelas (Flynn)

- Single instruction, single data stream: SISD
- Single instruction, multiple data stream: SIMD
- Multiple instruction, single data stream: MISD
- Multiple instruction, multiple data stream: MIMD



Single Instruction, Single Data Stream - SISD

- Procesador único
- Flujo de instrucciones único
- Datos almacenados en memoria no compartida

- Ya lo vimos

Single Instruction, Multiple Data Stream - SIMD

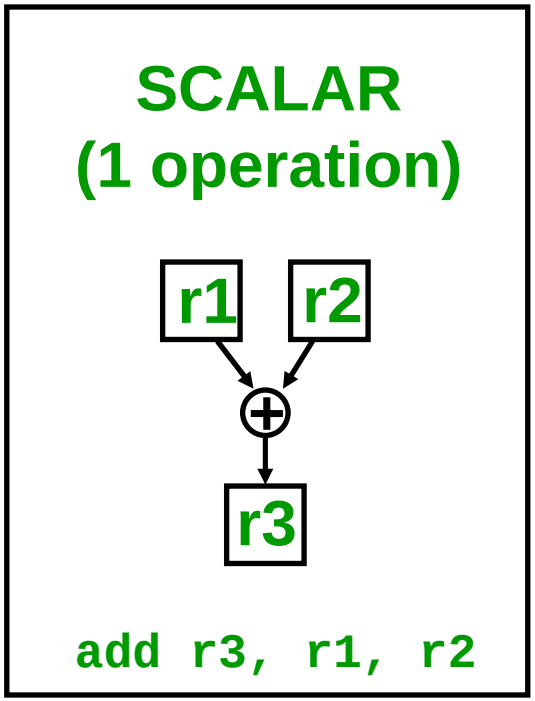
- Cada instrucción de máquina controla la ejecución simultánea y sincronizada paso a paso sobre una cantidad de elementos de proceso
- Cada instrucción es ejecutada por cada procesador sobre un conjunto de datos diferentes
- Cada elemento de proceso puede tener su propia memoria asociada o puede haber memoria compartida
- Procesadores vectoriales y matriciales
  - Procesamiento multimedia

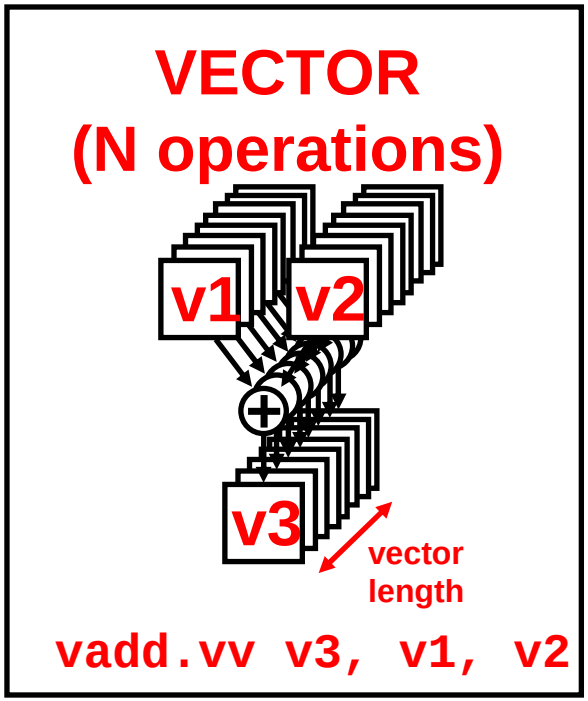
Procesadores Vectoriales (1/2)

- Problemas matemáticos que involucran procesos físicos (en general, simulación de campos continuos)
  - Dinámica de fluidos, sismología, meteorología, física nuclear.
- Cálculos repetitivos de punto flotante sobre grandes arreglos de números con alta precisión
- Supercomputadores manejan estos problemas
  - Millones de flops y millones de dólares
  - Optimizados para cálculo, no para multitarea ni operaciones de Entrada/Salida
  - Mercado limitado
    - I+D, gobierno, meteorología

Procesadores Vectoriales (2/2)

- Inicialmente desarrollados para aplicaciones de supercomputación, hoy en día son importantes en proceso multimedia
- Instrucciones operan sobre arrays de escalares: "vectores"





Procesamiento vectorial

- Se completa mucho trabajo en una sola instrucción vectorial (se ahorran loops)
  - Menos fetch de instrucciones
- Menos loops -> menos hazards de control
- Verificación de hazards de datos sólo entre vectores, no para cada una de las operaciones escalares involucradas en un cálculo vectorial
  - Diseño simple, alta frecuencia de reloj
  - Posible implementación mediante pipeline muy largo
- Instrucciones vectoriales acceden a memoria con un patrón conocido
  - Prefetch efectivo
  - Usando bancos de memoria se amortiza la latencia de memoria con la carga de un gran número de elementos
  - Podría eliminarse cache de datos.

Estilos de Arquitecturas Vectoriales

- Procesadores vectoriales memoria-memoria
  - Todas las operaciones vectoriales son de memoria a memoria

- Procesadores vector-registro
  - Todas las operaciones vectoriales son entre registros vectoriales (excepto load/store de vectores)
  - Equivalente vectorial de arquitecturas load-store
  - Máquinas vectoriales a partir de fin de los 80s

Componentes de un Procesador Vectorial

- CPU escalar: registros, datapaths, lógica de fetch de instrucciones
- Registros vectoriales
  - Banco de memoria de largo fijo almacena un único vector
  - Típicamente 8-32 registros vectoriales (1-8 Kbits)
  - Se pueden ver como elementos de 64b, 32b, 16b, 8b
- Unidades funcionales vectoriales (FUs)
  - En general trabajan en pipeline
  - Típicamente 2 a 8 FUs: enteros y PF
- Unidades vectoriales load-store (LSUs)
  - En pipeline, load/store de vectores
  - Pueden haber múltiples LSUs
- Cross-bar para interconexión de FUs , LSUs, registros

Operaciones vectoriales de memoria

- Operaciones load/store mueven vectores entre registros y memoria
- Tres tipos de direccionamiento
  - Unit stride: elementos consecutivos en memoria
  - Non-unit stride: elementos equidistantes entre sí en memoria
    - Ej.: producto escalar de una fila y una columna de una matriz
  - Indexed (gather-scatter)
    - Incluye características de direccionamiento indirecto
    - Adecuado para *representaciones dispersas*
- Soporte de varias combinaciones de anchos de los datos en memoria
  - {.L,.W,.H.,.B} x {64b, 32b, 16b, 8b}

Ejemplo de instrucciones vectoriales

Instr.	Operandos	Operación	Comentario
VADD.VV	V1,V2,V3	V1=V2+V3	vector + vector
VADD.SV	V1,R0,V2	V1=R0+V2	scalar + vector
VMUL.VV	V1,V2,V3	V1=V2xV3	vector x vector
VMUL.SV	V1,R0,V2	V1=R0xV2	scalar x vector
VLD	V1,R1	V1=M[R1..R1+63]	load, stride=1
VLDS	V1,R1,R2	V1=M[R1..R1+63*R2]	load, stride=R2
VLDX	V1,R1,V2	V1=M[R1+M[V2+i],i=0..63]	indexed("gather")
VST	V1,R1	M[R1..R1+63]=V1	store, stride=1
VSTS	V1,R1,R2	M[R1..R1+63*R2]=V1	store, stride=R2
VSTX	V1,R1,V2	M[R1+M[V2+i],i=0..63]=V1	indexed("scatter")

+ todas las instrucciones "normales" (estilo RISC)…

Ejemplo de código vectorial

Y[0:63] = Y[0:63] + a*X[0:63]

SAXPY de 64 elementos : escalar

```
        LD      R0,a
        ADDI    R4,Rx,#512
loop:   LD      R2, 0(Rx)
        MULTD   R2,R0,R2
        LD      R4, 0(Ry)
        ADDD    R4,R2,R4
        SD      R4, 0(Ry)
        ADDI    Rx,Rx,#8
        ADDI    Ry,Ry,#8
        SUB     R20,R4,Rx
        BNZ     R20,loop
```

SAXPY de 64 elementos : vectorial

```
LD      R0,a      #load scalar a
VLD     V1,Rx     #load vector X
VMUL.SV V2,R0,V1  #vector mult
VLD     V3,Ry     #load vector Y
VADD.VV V4,V2,V3  #vector add
VST     Ry,V4     #store vector Y
```

Procesamiento Multimedia

- Desktop y servidores
  - Gráficos 3D, animación computarizada, modelado 3D (juegos, cad, películas)
  - Reconocimiento de voz
  - Procesamiento de imágenes
  - Codificación y decodificación video/audio (mpeg-mp3, telefonía IP)
  - Digital libraries/media mining

- Embedded:
  - Gráficos 3D (cosolas de juegos)
  - Codif/Decodif. de video/audio (mpeg/mp3, set top boxes)
  - Procesamiento de imágenes (cámaras digitales)
  - Procesamiento de señales (teléfonos celulares)

Extensiones SIMD para GPP

- Motivación
  - Baja performance de procesamiento mmedia en GPPs
  - Costo y poca flexibilidad de ASICs especializados para gráficos/video
  - Tipos de datos angostos -> infrautilización de datapaths y registros
- Idea básica : sub-word parallelism
  - Tratar un registro de 64-bits como un vector de 2x32-bits, 4x16-bits o 8x8-bits (short vectors)
  - Partición de datapaths de 64 bits para manejar múltiples operaciones angostas en paralelo

Revisión de extensiones SIMD

Vendor	Extension	Year	# Instr	Registers
HP	MAX-1 & 2	94,95	9,8 (int)	Int 32x64b
Sun	VIS	95	121 (int)	FP 32x64b
Intel	MMX	97	57 (int)	FP 8x64b
AMD	3DNow!	98	21 (fp)	FP 8x64b
Motorola	Altivec	98	162 (int,fp)	32x128b (new)
Intel	SSE	98	70 (fp)	8x128b (new)
MIPS	MIPS-3D	?	23 (fp)	FP 32x64b
AMD	E 3DNow!	99	24 (fp)	8x128 (new)
Intel	SSE-2	01	144(int,fp)	8x128 (new)
Intel	SSE-3	04	13 new, hyper-thread	

- MMX: MultiMedia eXtension
- SSE: Streaming SIMD Extensions

Programación con extensiones SIMD

- Librerías optimizadas
  - Escritas en assembler, distribuidas por el fabricante
  - Se necesita API bien definida para formatear datos y usarla
- Macros para lenguajes de programación como C/C++
  - Permiten scheduling de instrucciones y optimizaciones de asignaciones de registros para procesadores específicos
  - No portable, no estándar
- Compiladores para extensiones SIMD
  - En los hechos, no hay compiladores disponibles
  - Assembler!

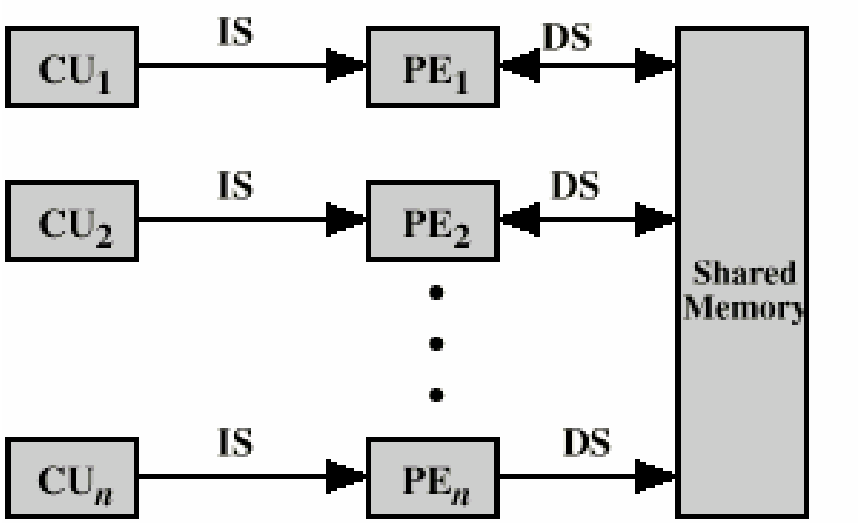
Multiple Instruction, Multiple Data Stream- MIMD

- Varios procesadores ejecutan **simultáneamente** secuencias de **instrucciones diferentes** sobre conjuntos de **datos diferentes**

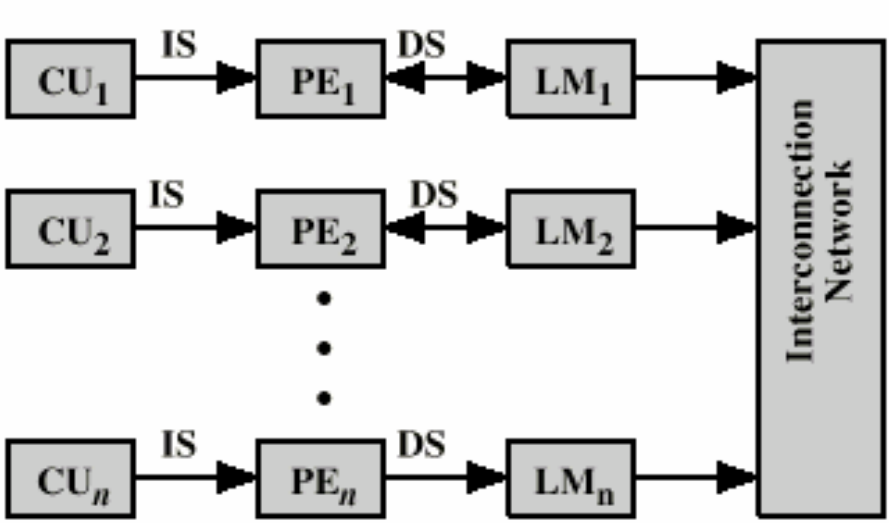
- Procesadores de propósito general

- Se pueden clasificar según cómo sea la comunicación:
  - Multiprocesadores Simétricos (SMPs)
    - Varios procesadores, memoria compartida
  - Sistemas de Acceso No Uniforme a Memoria (NUMA)
  - Clusters
    - Conjunto de computadores conectados en red

MIMD, organización de la memoria



- MIMD Memoria Compartida



- MIMD Memoria Distribuida

Multiprocesadores Simétricos:
Sistemas fuertemente acoplados

- Computador "stand alone" con las siguientes características
  - Dos o más procesadores similares, de capacidad comparable
  - Procesadores comparten memoria y E/S
    - Conectados por un bus u otra conexión interna
    - Tiempo de acceso a memoria es similar para cada procesador
    - Acceso a los mismos dispositivos de E/S usando canales compartidos (o no)
  - Todos los procesadores pueden desempeñar las mismas funciones -> **simétricos**
  - Sistema Operativo integrado
    - Provee interacción entre procesadores
    - Interacción a nivel de job, tarea, archivos y datos
      - Por oposición a los clusters (débilmente acoplados), que interactuán a nivel de mensajes y/o archivos completos

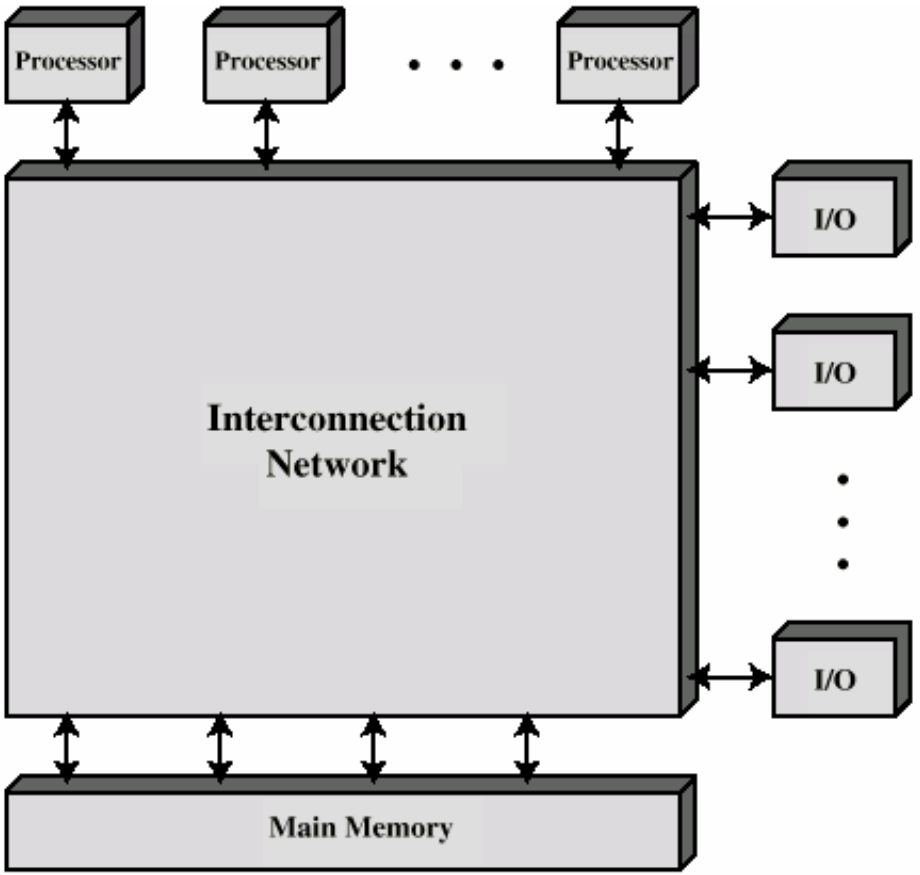
Ventajas SMP

- En un SMP el SO planifica la distribución de procesos o threads entre todos los procesadores (transparente al usuario). Ventajas con respecto a mono-procesador:

- Performance
  - Si el trabajo se puede paralelizar
- Disponibilidad
  - Dado que todos los procesadores desempeñan las mismas funciones, el fallo de un procesador no detiene el sistema
- Crecimiento incremental
  - Se pueden aumentar las prestaciones agregando procesadores (hasta cierto punto)
- Escalabilidad
  - Se pueden ofrecer distintas configuraciones/prestaciones basadas en la cantidad de procesadores

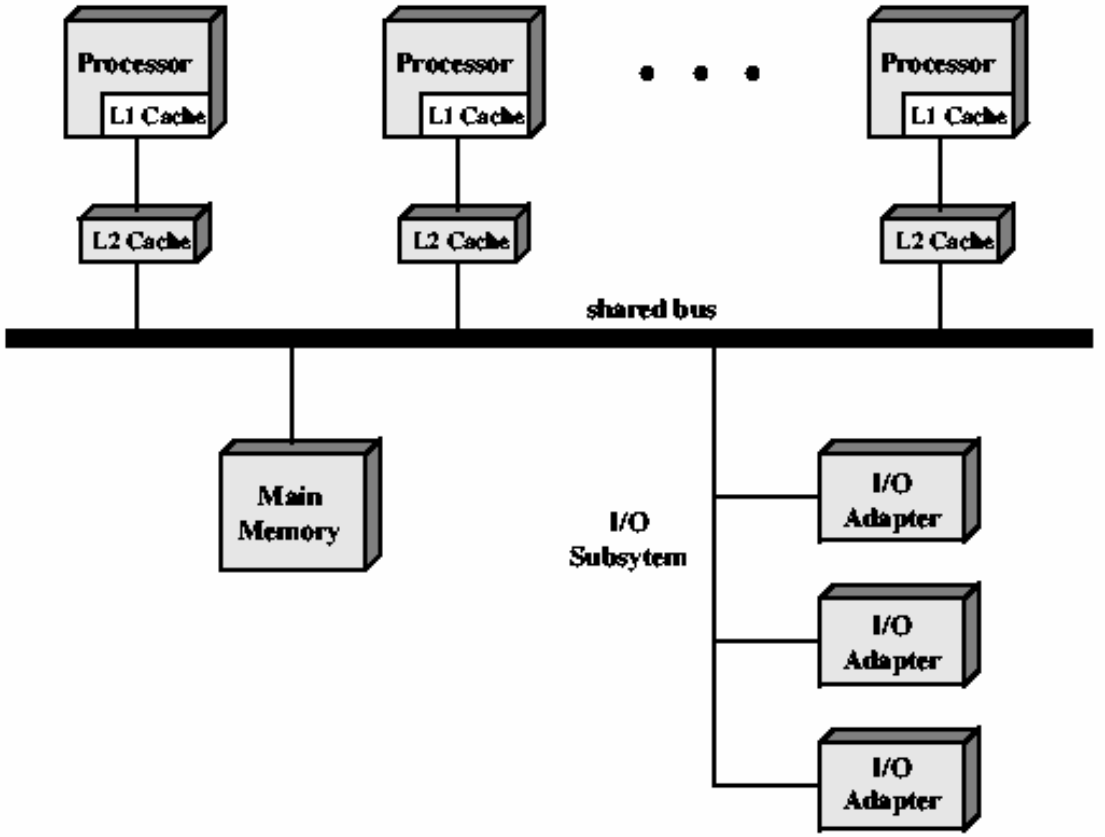
Sistemas fuertemente acoplados:
Diagrama de bloques

- Alternativas de organización
  - Bus común o de tiempo compartido
  - Memoria Multipuerto
  - Unidad de control central
- Los procesadores se comunican mediante la memoria (área de datos compartida), y pueden intercambiar señales directamente



Bus de tiempo compartido (1/2)

- Estructura e interfaces similares a sistemas mono-procesador
  - Direccionamiento, para distinguir módulos en el bus
  - Arbitraje: cada módulo puede ser master temporalmente
  - Tiempo compartido: si un módulo controla el bus, el resto debe esperar



Bus de tiempo compartido (2/2)

- Ventajas
  - Simplicidad
  - Flexibilidad
  - Confiabilidad
- Desventajas
  - Rendimiento limitado por el ciclo de bus
  - Cada procesador debe tener una cache local
    - Reducir la cantidad de accesos al bus
  - Lleva a problemas con la coherencia del cache
    - Se resuelve en hardware

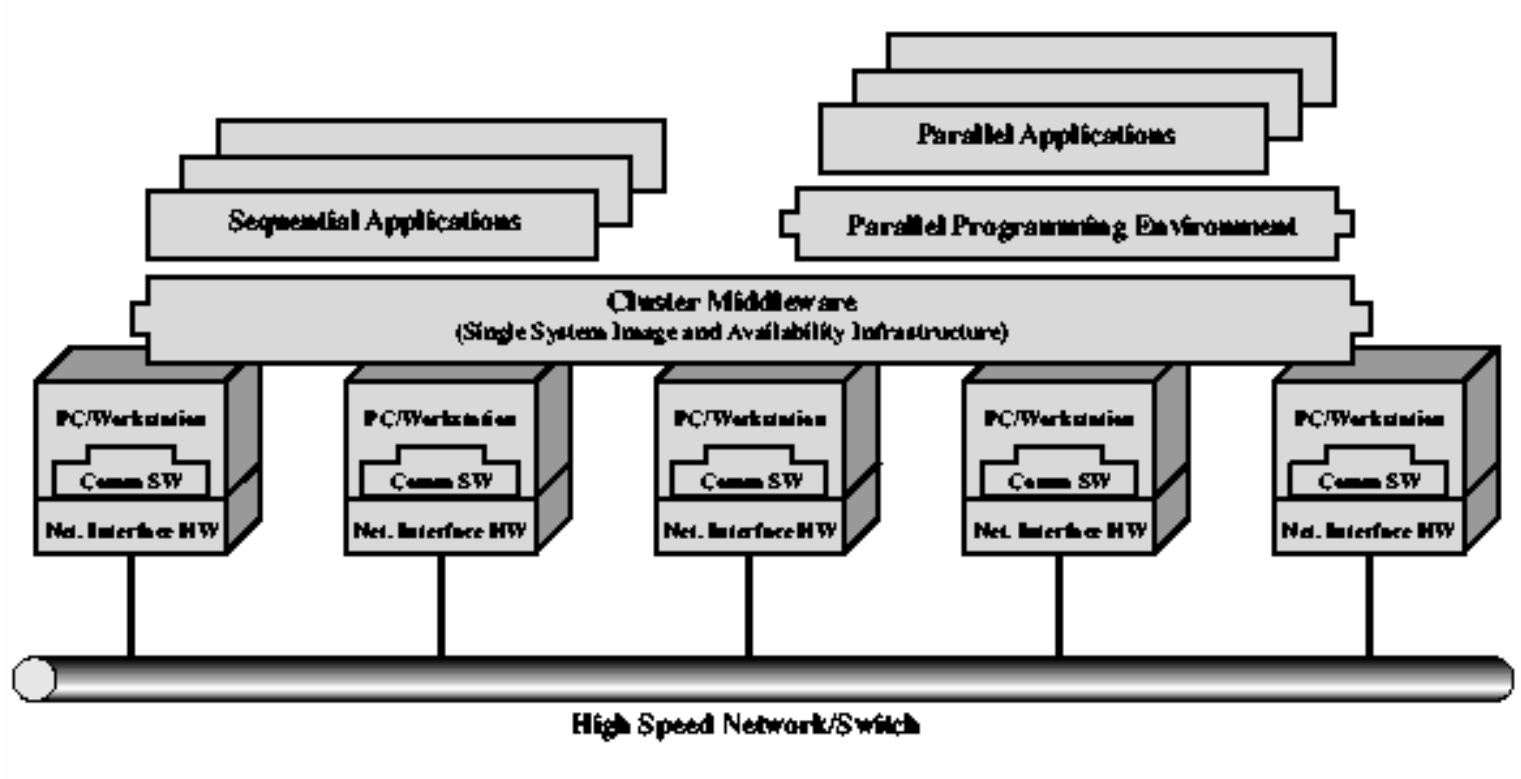
Coherencia del Cache

- Problema: múltiples copias de los mismos datos en diferentes caches
  - Write back puede llevar a inconsistencias
  - Write through puede dar problemas si las caches no monitorizan el tráfico de memoria
- Soluciones de software
  - Compilador y SO se encargan del problema
  - Datos potencialmente problemáticos se marcan como no-cacheables
  - El compilador toma decisiones "conservadoras", uso ineficiente de la cache
- Solución de hardware, protocolos de coherencia del cache
  - Reconocimiento dinámico de problemas en run time
    - Uso más eficiente del cache, transparente al programador
  - Protocolos de directorio (usando controlador central)
  - Protocolos de sondeo (Snoopy protocols)

Clusters

Arquitectura Cluster

- Grupo de computadores (nodos) interconectados trabajando en conjunto como un recurso unificado



Clusters

- Alternativa a SMP
- Alto rendimiento

- Beneficios del cluster
  - Escalabilidad incremental
  - Alta disponibilidad
  - Mejor relación precio / rendimiento

Middleware de Cluster

- Imagen unificada al usuario (Single system image)
  - Único punto de entrada
  - Única jerarquía de archivos
  - Espacio de memoria único
  - Gestión de "jobs" unificada
  - Interfaz de usuario única
  - Espacio de E/S único
  - Espacio de procesos único
- Checkpointing (recuperación ante fallas)
- Migración de Procesos (balance de carga)

Clusters vs. SMP

- Ambos proveen soporte multiprocesador para aplicaciones demandantes
- Ambos disponibles comercialmente
  - SMP "más viejo"
- SMP:
  - Gestión y Control simplificado
  - Más parecido a sistemas mono-procesador
    - La planificación es la diferencia fundamental
  - Menos espacio físico
  - Menos consumo de energía
- Cluster:
  - Escalabilidad superior
  - Mayor disponibilidad
    - Redundancia

Preguntas?