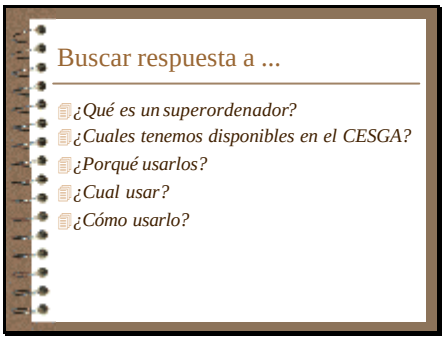
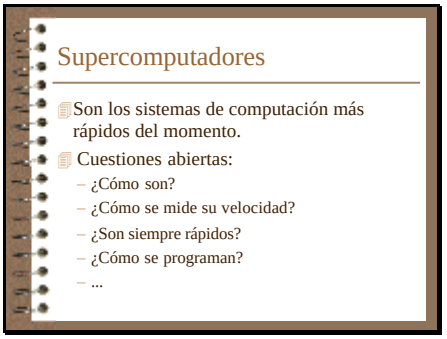
Diapositiva 1

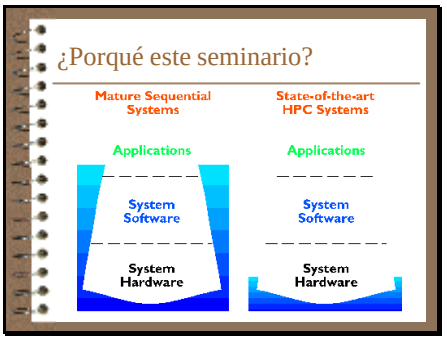
Diapositiva 2



Diapositiva 3

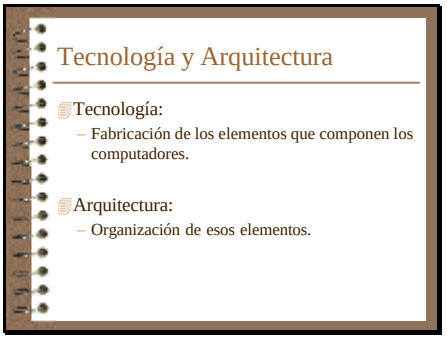


Diapositiva 4



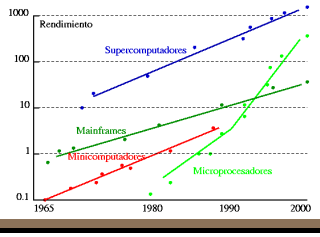
Diapositiva 5

Diapositiva 6



Diapositiva 7

Evolución del rendimiento



Diapositiva 8

Modelo de von Newmann

- Las instrucciones operan sobre datos escalares.
- Secuencialidad en la ejecución de las instrucciones.
  - Ciclo de ejecución de instrucciones.
- Escasas fuentes de paralelismo.

Diapositiva 9

Procesadores escalares (segmentados)

- Objetivos:
  - 1 instrucción por ciclo de reloj
  - Frecuencia de reloj alta
- Mecanismo:
  - Segmentación
- Limitaciones:
  - Dependencias
  - Latencia de la memoria

Diapositiva 10

La segmentación

- El ciclo de ejecución de las instrucciones se ejecuta como una cadena de producción:
  - La primera instrucción tarda mucho en ejecutarse (varios ciclos de reloj).
  - Las siguiente ya está muy avanzada y sólo necesita un ciclo más para finalizar.

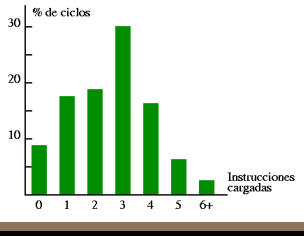
Diapositiva 11

Procesadores superescalares

- Objetivo:
  - Más de una instrucción por ciclo.
- Mecanismo:
  - Ejecución paralela de pequeños grupos de instrucciones.
  - Replicación de Unidades Funcionales.
- Limitaciones:
  - Dependencias.
  - ILP limitado.

Diapositiva 12

¿Cuántas instrucciones paralelas?



Diapositiva 13

Sistemas vectoriales

- Objetivo:
  - Bajar el número de instrucciones.
- Mecanismo:
  - Instrucciones sobre vectores.
  - Unidades vectoriales segmentadas.
  - Vectorización de lazos.
- Limitaciones:
  - Programas con lazos regulares.
  - Dependencias.

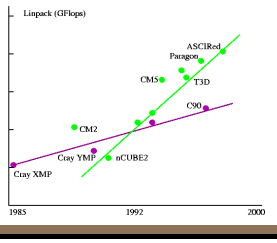
Diapositiva 14

Sistemas paralelos

- Objetivo:
  - Aumentar IPC.
- Mecanismo:
  - Uso de un conjunto de procesadores que **cooperan**.
- Limitaciones:
  - Extracción automática de paralelismo.
  - Dependencias.

Diapositiva 15

Vectoriales .vs. paralelos



Diapositiva 16

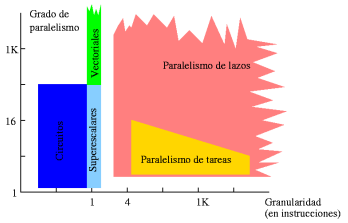
Tiempo de ejecución

T = N * 1/IPC * Tc

- Procesadores Superescalares
  IPC = bajo, Tc bajo.
- Procesadores Vectoriales
  IPC =1, N bajo, Tc bajo.
- Multiprocesadores
  IPC alto.

Diapositiva 17

Grado de paralelismo



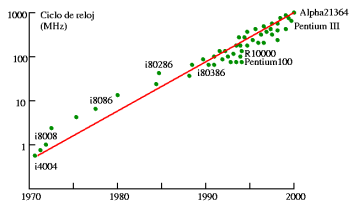
Diapositiva 18

Tendencias tecnológicas

- Crecimiento del número de transistores por chip.
- Crecimiento moderado de la frecuencia de reloj.
- Desarrollo de tecnología de redes.

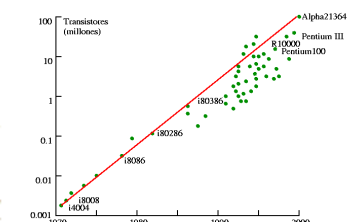
Diapositiva 19

Frecuencia de reloj de los micros



Diapositiva 20

Número de transistores por chip

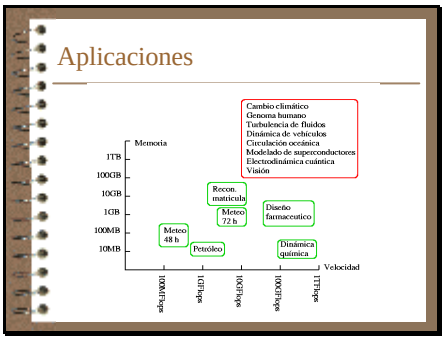


Diapositiva 21

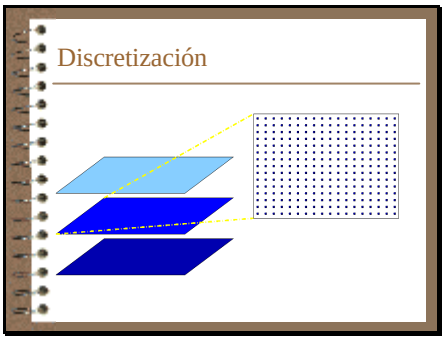
Demanda de las aplicaciones

- Computación científica:
  - Biología, física, química, ...
  - Básicamente en procesos de simulación.
- Computación de propósito general:
  - Vídeo, bases de datos, CAD, transacciones, ...
  - Básicamente tratamiento de grandes cantidades de datos.

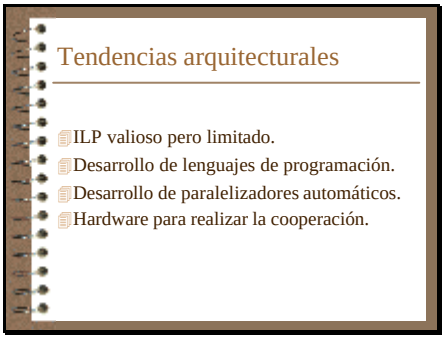
Diapositiva 22



Diapositiva 23



Diapositiva 24



Diapositiva 25

Procesadores vectoriales

- Proporcionan operaciones de alto nivel que trabajan sobre vectores de datos.
- Una única **instrucción vectorial** implica una gran cantidad de trabajo.
- El coste de latencia a memoria se paga una sola vez en lugar de una por cada dato.

Diapositiva 26

Hardware

- Unidades segmentadas específicas para operaciones aritmético/lógicas en punto flotante.
- Memoria entrelazada:
  - Está dividida en módulos que pueden ser accedidos simultáneamente.

Diapositiva 27

Componentes principales

- Registros vectoriales.
- Registros escalares.
- Unidades funcionales segmentadas.
- Memoria entrelazada.
- Unidades de carga y almacenamiento.

Diapositiva 28

El VPP300

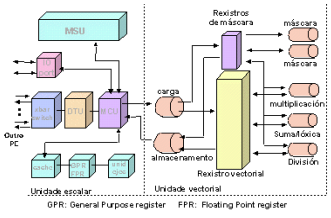


Fig. 2.- Configuración hardware dun elemento de procesamento (PE)

Diapositiva 29

Algunas características

- Se reduce enormemente el número de instrucciones ejecutadas.
- Tiempo de arranque.
- Velocidad de iniciación.
- Adecuación del tamaño de los vectores al hardware.
- Separación entre elementos accedidos.

Diapositiva 30

Algunas características

- Almacenamiento por filas o por columnas.
- Solapamiento entre la unidad escalar y la vectorial.
- Dependencias entre iteraciones.
- Reducciones (sumatorio, máximo, ...).
- Encadenamiento.
- Condicionales (registros de máscara).

Diapositiva 31

El VPP300

- Números:
  - 6 CPUs
  - 12 GB de memoria.
  - 260 Gb de disco.
  - 14,4 Gflops pico.

Diapositiva 32

Configuración

- Crossbar entre CPUs.
- Unidad escalar: LIW
- Unidad vectorial con 7 “pipelines”:
  - Suma y operaciones lógicas.
  - Producto.
  - División.
  - Carga.
  - Almacenamiento.
  - Dos de máscara.

Diapositiva 33

El VPP300

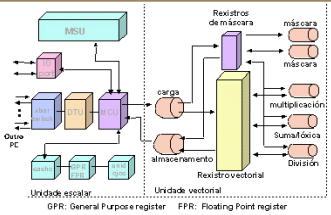


Fig. 2.- Configuración hardware dun elemento de procesamento (PE)

Diapositiva 34

El VPP300

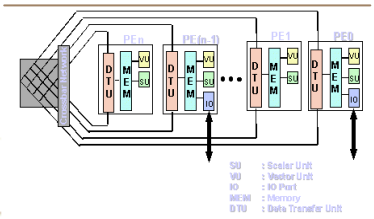


Fig. 3.- Conexión dos elementos de procesamento a través do *crossbar*.

Diapositiva 35

Arquitecturas Paralelas

- Un computador paralelo es una colección de elementos de procesamiento que cooperan para resolver rápidamente grandes problemas computacionales.

Diapositiva 36

Cuestiones abiertas

- ¿Cuántos procesadores?
- ¿De qué potencia?
- ¿Cómo cooperan?
- ¿Cuales son los mecanismos para expresar la cooperación?
- ¿Cual es la influencia en el rendimiento?
- ¿Cómo escala la arquitectura?
- ¿Qué aplicaciones son adecuadas?

Diapositiva 37

¿Son necesarios los sistemas paralelos?

- Claramente: **SI**.
  - De hecho, son inevitables.
- Motivación:
  - Demanda de las aplicaciones.
  - Tendencias tecnológicas.
  - Tendencias arquitecturales.
  - Bajada de costes.
  - Computación científica (Supercomputación).

Diapositiva 38

Tecnología y coste

- Los micros han mejorado en prestaciones
  - Frecuencia de reloj.
  - FPUs segmentadas.
  - Uso eficiente de caches (blocking).
- Los micros son baratos
  - Coste de desarrollo alto > 100 M$.
  - PERO se venden en cantidades enormes.

Diapositiva 39

"Los multiprocesadores masivamente paralelos sustituyen a los supercomputadores tradicionales"

Diapositiva 40

Arquitecturas paralelas

- Tendencia de los fabricantes tradicionales:
  - SGI/Cray, Intel, IBM, Convex, Fujitsu, ...
- Algunos han fracasado:
  - TMC, KSR, nCUBE, ...

Diapositiva 41

Situación histórica

- Modelos rivales y arquitecturas divergentes.
- No existía línea de desarrollo predecible.
  - Arrays sistólicos.
  - Memoria compartida.
  - Pase de mensajes.
  - Dirigido por flujo de datos.
  - SIMD.
- Escaso desarrollo de software paralelo.

Diapositiva 42

Situación actual

- Abstracción de la comunicación
  - Primitivas de comunicación visibles al programador.
  - Se asume que las primitivas se realizan eficientemente.
- Implementación
  - Integración del controlador de comunicaciones.
  - Estructura de la red de interconexión.

Diapositiva 43

Objetivos

- Amplia aplicabilidad.
- Facilidad de programación.
- Escalabilidad.
- Bajo coste.

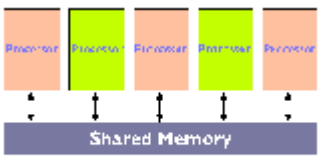
Diapositiva 44

Memoria compartida

- Todos los procesadores pueden acceder de forma **directa** a todas las posiciones de memoria del sistema.
- Todas las escrituras en una región compartida deben ser visibles para todos los procesos. Problemas:
  - Coherencia.
  - Consistencia.
  - Falsa compartición.

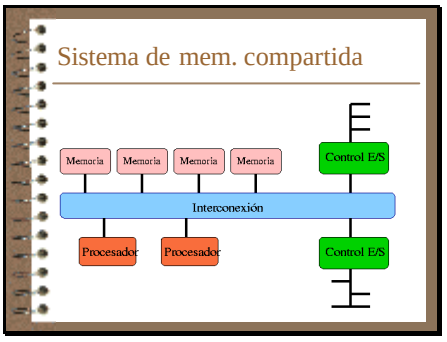
Diapositiva 45

Sistema de memoria compartida

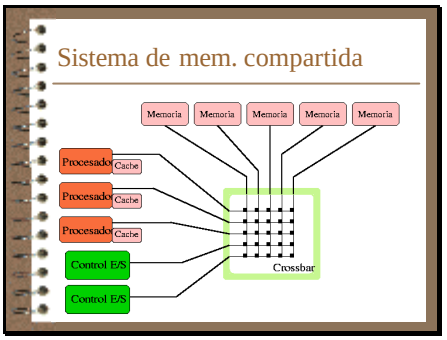


SM-MIMD Machine

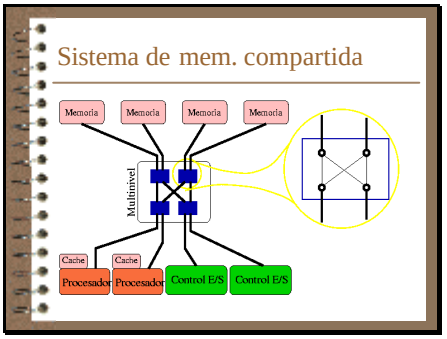
Diapositiva 46



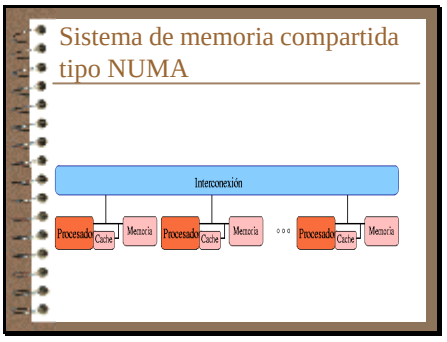
Diapositiva 47



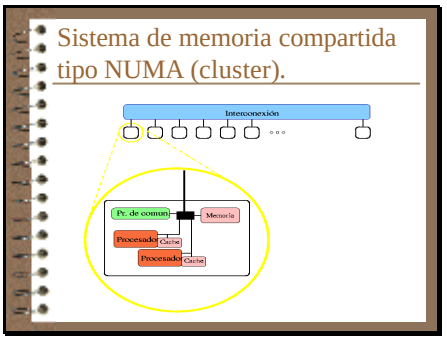
Diapositiva 48



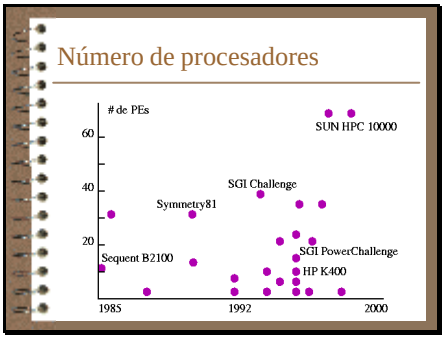
Diapositiva 49



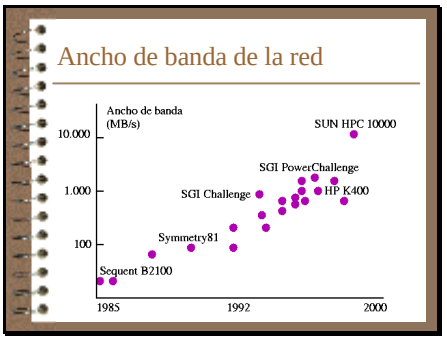
Diapositiva 50



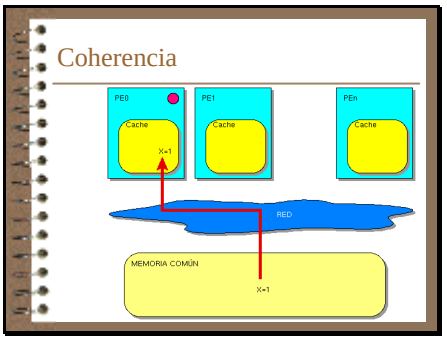
Diapositiva 51



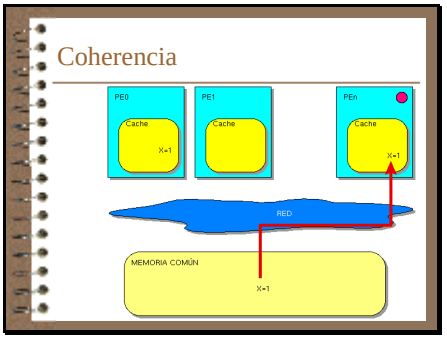
Diapositiva 52



Diapositiva 53

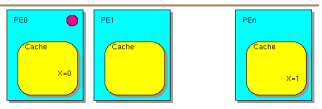


Diapositiva 54



Diapositiva 55

Coherencia



Diapositiva 56

Mecanismos para compartición de la memoria

- La comunicación, compartición y sincronización se hace mediante operaciones de lectura y escritura de variables compartidas.
- Modelo de programación amigable (uniprocesador + sincronización).
- Escasa escalabilidad

Diapositiva 57

La SUN HPC4500

- 10 procesadores UltraSPARC II.
- 40 MB de memoria (4 por procesador).
- 40 GB de disco (4 por procesador).
- Bus a 2,6GB/seg.

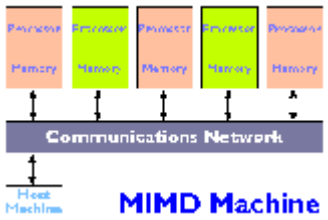
Diapositiva 58

Sistemas de memoria distribuida

- Un grupo de computadores completos conectados por una red de interconexión.
- La cooperación se realiza a través de paso de mensajes.

Diapositiva 59

Sistema de mem. distribuida



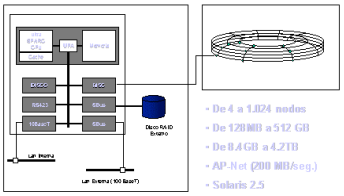
Diapositiva 60

El AP3000

- 20 CPUs en 16 nodos.
- Procesador UltraSPARC.
- 25 GB de memoria.
- 89 GB de disco.
- Red de interconexión estática en toroide (AP-Net).
- *Routing* wormhole.

Diapositiva 61

El AP3000



- De 4 a 1.024 nodos
- De 128 MB a 512 GB
- De 8.4 GB a 4.2 TB
- AP-Net (200 MB/seg.)
- Solaris 2.5

Figura 1. Configuración hardware dos nodos do AP3000 e rede de conexión.

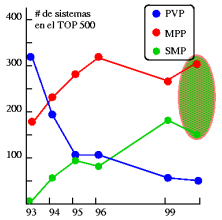
Diapositiva 62

Supercomputadores

- Lista de los más rápidos: TOP500.
  http://www.top500.org/
- Primero: ASCI White con 8192 CPUs.
- En España dos a Junio de 01 (427 y 457).
- Hermanos mayores de los disponibles en el CESGA:
  - **16**: VPP5000 con 100 procesadores.
  - **57**: SUN HPC4500 con 896 procesadores.
- FUTURO: Clusters y GRIDs.

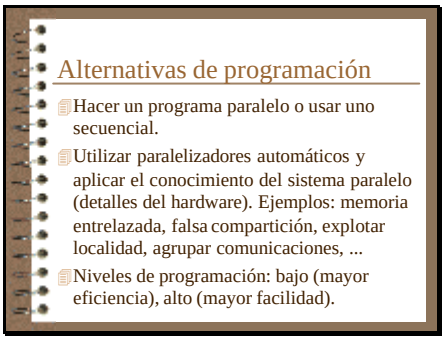
Diapositiva 63

Supercomputadores

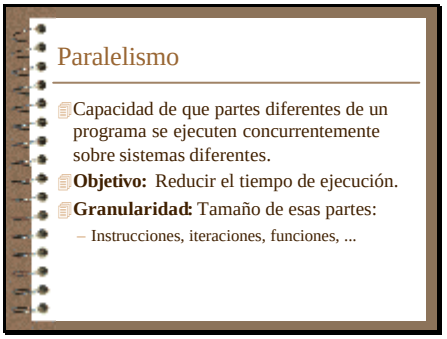


Diapositiva 64

Diapositiva 65

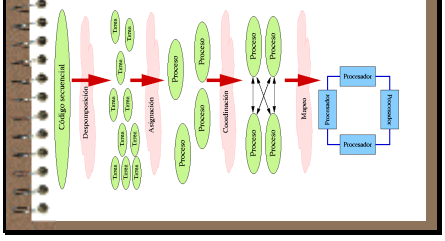


Diapositiva 66



Diapositiva 67

Fases



Diapositiva 68

Tipos de paralelismo

- Paralelismo de datos regulares.
- Paralelismo de datos irregulares.
- Paralelismo basado en tareas.
- Paralelismo temporal (segmentación o “pipeline”).

Diapositiva 69

Cuestión

- ¿Cuándo dos sentencias pueden ser ejecutadas en paralelos?
  - En un procesador:

    sentencia 1;
    sentencia 2;
  - En dos procesadores:
    - *PE1* — sentencia 1;
    - *PE2* — sentencia 2;

Diapositiva 70

Base fundamental

- La ejecución en los dos procesadores de ambas sentencias es independiente: No existe control sobre el **orden** entre los procesadores. Dos posibilidades:
  - *PE1* *PE2*
    sentencia 1;
    sentencia 2;
  - *PE1* *PE2*
    sentencia 2;
    sentencia 1;

Diapositiva 71

¿Son paralelos?

- El orden no debe importar en el resultado final.
- En otras palabras:
  sentencia 1;
  sentencia 2;
  debe producir el mismo resultado que:
  sentencia 2;
  sentencia 1;

Diapositiva 72

Ejemplos

a=1;
b=2;
– Son paralelas.
a=1;
b=a;
– No son paralelas.
a=1;
a=2;
– No son paralelas.

Diapositiva 73

Dependencias

- S2 presenta una **dependencia verdadera** con S1,
  - si y sólo si
  - S2 lee un valor escrito por S1.
- S2 presenta una **antidependencia** con S1,
  - si y sólo si
  - S2 escribe un valor leido por S1.

Diapositiva 74

Dependencias

- S2 presenta una **dependencia de salida** con S1,
  - si y sólo si
  - S2 escribe un valor también escrita por S1.
- Algunas dependencias se pueden eliminar.
- S1 y S2 se pueden ejecutar en paralelo **si y sólo si** no existen dependencias de ningún tipo entre ellas.

Diapositiva 75

Lazos

- Son la principal fuente de paralelismo.
- Ejemplo:

```
for (i=0;i<100;i++)
  a[i]=b[i];
```

  - No presenta dependencias
  - Las iteraciones pueden ser ejecutadas en paralelo.

Diapositiva 76

Lazos

- Ejemplo:

```
for (i=0;i<100;i++) {
  a[i]=i;
  b[i]=2*i; }
```

Iteraciones y sentencias pueden ejecutarse en paralelo.

Diapositiva 77

Lazos

- Ejemplo:

```
for (i=0;i<100;i++) a[i]=i;
for (i=0;i<100;i++) b[i]=2*i;
```

Iteraciones y lazos son paralelos.

- Ejemplo:

```
for (i=0;i<100;i++)
  a[i]= a[i]+100;
```

Hay dependencia consigo misma.

El lazo sigue siendo paralelo

Diapositiva 78

Lazos

- Ejemplo:

```
for (i=0;i<100;i++)
  a[i]=f(a[i-1]);
```

Hay dependencia entre a[i] y a[i-1].

Las iteraciones del lazo no son paralelas.

- **Problema:** Dependencias entre iteraciones.

Diapositiva 79

Lazos

- Ejemplo:

```
for (i=0;i<100;i++)
  for (j=0;j<100;j++)
    a[i][j]= f(a[i][j-1]);
```

  Lazo i independiente.

  Lazo j presenta dependencias entre iteraciones.

- Lazo i paralelo, lazo j no.

Diapositiva 80

Lazos

- Ejemplo:

```
printf("a");
printf("b");
```

  Dependencia de recurso hardware.

- Ejemplo:

```
a=f(x);
b=g(y);
```

  Es necesario analizar qué variables son modificadas por f y g.

Diapositiva 81

Lazos

- Ejemplo:

```
for (i=0;i<100;i++)
  a[i+10]= f(a[i]);
```

  Hay dependencia entre a[10], a[20], ...

  Hay dependencia entre a[11], a[21], ...

- Se puede explotar el paralelismo parcialmente.

Diapositiva 82

Lazos

- Ejemplo:

```
for (i=0;i<100;i++) {
  a[i]= .....;
  ..... = a[i-1]; }
```

Hay dependencia entre los accesos a a[i] y a[i-1].

- Paralelización imposible.
- Segmentación posible.

Diapositiva 83

Lazos

- Ejemplo:

```
for (i=0;i<100;i++)
  a[i] = f(a[ indice[i]]);
```

No se sabe si existen dependencias.

Depende del conocimiento de indice[i].

El compilador no lo puede saber. El usuario o la ejecución si.

Diapositiva 84

Algunos detalles

- Los paralelizadores analizan las dependencias y deciden la paralelización.
- El programador puede hacer este trabajo a mano.
- El compilador es más seguro (correcto).
- El usuario es más eficiente, puede analizar más circunstancias.

Diapositiva 85

Resumen

- El orden de sentencias no debe importar.
- Las sentencias no deben presentar dependencias.
- Algunas dependencias pueden ser eliminadas reprogramando.
- Algunas dependencias no son obvias.

Diapositiva 86

Sincronización

- Capacidad de controlar el orden de los eventos entre los procesadores.
- Algunas partes de un programa pueden ser paralelas, mientras otras puede que no lo sean. Las sincronizaciones garantizan el orden donde sea necesario.
- Primitivas:
  - signal(x)
  - wait(x)

Diapositiva 87

Sincronización

- Presenta un coste elevado.
- Otras primitivas:
  - fork
  - join
  - barrier
- Reducciones:

```
sum=0;
for(i=0;i<100;i++) sum+= a[i];
```

Diapositiva 88

Pase de mensajes

- La comunicación (cooperación) entre procesadores está integrada al nivel de E/S, y no en el sistema de memoria (ahora los bloques de diseño son computadores completos).
- Los procesadores sólo pueden acceder a su memoria local.
- Comunicaciones mediante llamadas explícitas al S.O. (*send* y *receive*).

Diapositiva 89

Mecanismo de pase de mensajes

- La combinación *send-receive* establece una copia de memoria a memoria + una sincronización entre los procesos.
- Coste temporal alto (operación de E/S):
  - Construir cabecera del mensaje.
  - Copiar datos al buffer de la red.
  - Enviar el mensaje.
  - Copiar datos en el buffer receptor.
  - Copiar datos al espacio del proceso receptor.

Diapositiva 90

Pase de mensajes

- El usuario tiene control explícito sobre los procesos.
- Una variable x se refiere a diferentes localizaciones físicas dependientes del procesador.
- Todas las variables son locales.

Diapositiva 91

Pase de mensajes

- El usuario debe:
  - Dividir el programa en partes.
  - Crear y eliminar procesos.
  - Particionar y distribuir datos.
  - Realizar la traslación de índices.
  - Comunicaciones.
  - Sincronización: muchas veces está implícita en las comunicaciones.

Diapositiva 92

Pase de mensajes

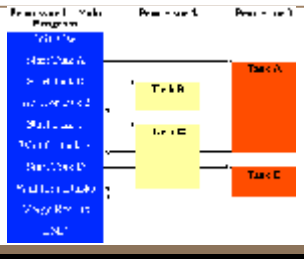
- Funciones:

MPI_init	MPI_bcast
MPI_finalize	MPI_gather
MPI_send	MPI_scatter
MPI_receive	MPI_reduce
MPI_barrier	MPI_allreduce

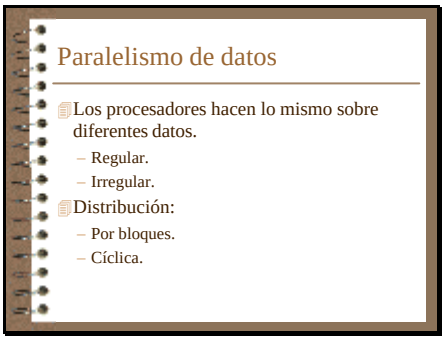
- Estrategias habituales:
  - Master-slave
  - SPMD

Diapositiva 93

Estrategia master-slave

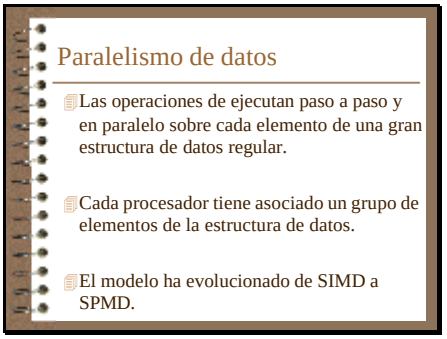


Diapositiva 94



Diapositiva 95

Diapositiva 96



Diapositiva 97

Paralelismo de datos

Diapositiva 98

Aspectos de diseño

- **Denominación:** ¿Cómo se referencian los datos compartidos? ¿De qué operaciones se dispone? ¿Cómo se designan los procesos?
- **Sincronización:** ¿Cómo se coordinan los procesos? ¿Cómo se establece la exclusión mutua?
- **Latencia:** Tiempo de establecer la comunicación o sincronización.
- **Ancho de banda:** Velocidad de la red.

Diapositiva 99

Denominación y sincronización

- Semántica de las primitivas.
  - **Memoria compartida**: read, write, operaciones atómicas (test&set, ...).
  - **Pase de mensajes**: Identificación de procesos y envío / recepción de mensajes.

Diapositiva 100

Latencia y ancho de banda

- Control de la latencia:
  - Reducir la frecuencia de las operaciones de latencia alta.
  - Uso de hardware con caches y controladores de comunicaciones.
  - Solapar comunicaciones con computaciones.

Diapositiva 101

HPF

- Standard para paralelismo de datos.
- Extensión del programa secuencial con directivas de distribución de datos entre los procesadores.
- El compilador produce el programa paralelo con las comunicaciones necesarias.
- En una plataforma secuencial, las directivas son ignoradas (comentarios).

Diapositiva 102

HPF

- Operaciones con arrays de Fortran90.
- Expresiones adicionales para el paralelismo:
  - forall
  - independent
- Directivas de distribución de datos:
  - distribute (block(k), cyclic(k)), align, ...
- Owner-computes rule.

Diapositiva 103

HPF

- Facilidad de programación.
- Difícil obtener altos rendimientos.
- No optimiza las comunicaciones.
- No sirve para computaciones irregulares.

Diapositiva 104

Memoria compartida

- Todos los procesos acceden al mismo espacio de memoria compartida.
- Cada variable x se refiere al mismo objeto independientemente de que procesador la referencia.

Diapositiva 105

Memoria compartida

- El usuario debe:
  - Decidir la descomposición de las computaciones.
  - Crear y eliminar procesos para esa descomposición.
  - Añadir sincronización para garantizar que las dependencias se verifican.

Diapositiva 106

Memoria compartida

- Mecanismo de bajo nivel: pthreads.
- Primitivas:

pthread_create	pthread_cond_init
pthread_exit	pthread_cond_destroy
pthread_join	pthread_cond_wait
pthread_mutex_init	pthread_cond_signal
pthread_mutex_destroy	pthread_cond_broadcast

Diapositiva 107

Comparativa

- Memoria compartida:
  - Más sencilla en su programación.
  - Más difícil cometer errores.
  - Las estructuras de datos cambian.
  - Se puedeparalelizar incrementalmente.
- Pase de mensajes:
  - Más fácil obtener altos rendimientos.
  - Mayor portabilidad.
  - Dificultad en detectar y corregir errores.

Diapositiva 108

Memoria compartida: OpenMP

- Standard de programación en memoria compartida.
- Incluye directivas (#pragmas) de compilación junto con funciones de librería.
- Se basa en la semántica de *fork-join*.
- En una plataforma secuencial, las directivas son ignoradas (comentarios).

Diapositiva 109

OpenMP

- Directivas de paralelización:
  parallel region, parallel for
- Directivas de datos:
  shared, private, firstprivate, threadprivate, reduction
- Directivas de sincronización:
  barrier, critical, flush

Diapositiva 110

OpenMP vs HPF

- OpenMP especifica la distribución de iteraciones:
  - Los datos se mueven hacia el procesador que tiene asignada la computación.
- HPF especifica la distribución de los datos:
  - Las computaciones son realizadas por el propietario del dato resultado.

Diapositiva 111

Convergencia

- Pase de mensajes y memoria compartida no implican restricción en el hardware, sólo son modelos de programación.
- Puede simularse cualquier modelo en cualquiera de las dos arquitecturas.
- El problema básico de eficiencia es similar: rendimiento de la red de interconexión.

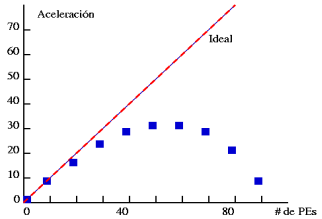
Diapositiva 112

Rendimiento

- Speed-up = Tsec/Tpar
- Eficiencia = Speed-up / P
- Escalabilidad
- Factores que influyen:
  - Comunicaciones, sincronizaciones, coherencia, ...
  - Balanceo de la carga.
  - Dependencias.
  - Optimizaciones del código.

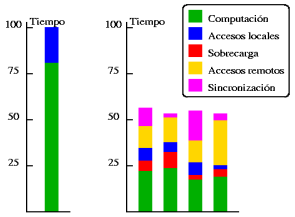
Diapositiva 113

Speed-up



Diapositiva 114

Tiempo de ejecución



Diapositiva 115

Resumen

- Tipos de hardware.
  - Características.
  - Convergencia del hardware.
- Tipos de modelos de programación.
  - Granularidad del paralelismo.
  - Estructuras del programa.
  - Características de facilidad.
  - Relación con el hardware.
  - Convergencia de los modelos.
  - Automatización o hacer un algoritmo específico.

Diapositiva 116

Conclusiones

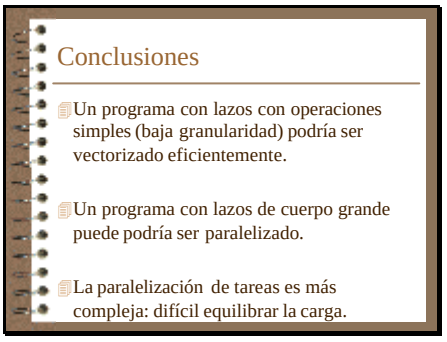
- Programar puede ser sencillo usando librerías y lenguajes de alto nivel. El problema es el rendimiento.
- Las arquitecturas estudiadas generalizan a los sistemas secuenciales: Los sistemas pueden ejecutar programas secuenciales a un rendimiento monoprocesador .

Diapositiva 117

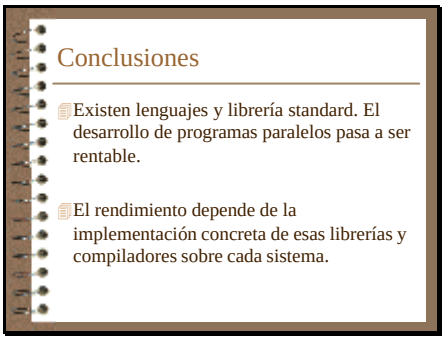
Conclusiones

- Usar librerías de bajo nivel implica mayor esfuerzo, siendo más asequible obtener altos rendimientos.
- En cualquier caso el conocimiento del hardware es indispensable para aprovecharlo.

Diapositiva 118



Diapositiva 119



Diapositiva 120