Relación con el hardware: Estructuras y funciones básicas

Gunnar Wolf

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Qué es un *sistema von Neumann*?

- Computadora *de programa almacenado*
  - En la *memoria principal*
  - *Mismo almacenamiento* para el programa siendo ejecutado que para sus datos
- Todas las computadoras que empleamos hoy en día son, para fines prácticos, sistemas von Neumann[1]

[1] Aunque podrían caracterizarse como *Harvard modificadas* partiendo desde un formalismo excesivo...

Implicaciones de *von Neumann*

- La arquitectura *no considera* la existencia de almacenamiento *persistente*
  - Dentro del procesador hay algunas (¡pocas!) localidaes para almacenar datos *muy limitados*, para *trabajar directamente con ellos* → *Registros*
  - La computadora cuenta únicamente con la *memoria de trabajo* → RAM, *almacenamiento primario*
  - El almacenamiento a largo plazo debe hacerse empleando controladores / mecanismos alternos, en medios específicos → *almacenamiento secundario*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Los registros

¿Qué son los registros?

- Memoria super-rápida (hoy en día, sub-nanosegundo), ubicada *dentro* del procesador
- Manejada *por referencia directa*, no por dirección
- Además de los datos del proceso guardan su *estado*
- ¿Qué significa *gestionados por el compilador*?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Procesadores basados en *acumuladores*

- Primeros procesadores: Uno o pocos *acumuladores*. Por ejemplo:
  - MOS 6502:
    - 1 acumulador (A) de 8 bits
    - 2 registros índice de 8 bits (X y Y)
    - 1 registro de estado del procesador (P), un apuntador al stack de 8 bits (S), un apuntador a instrucciones (PC) de 8 bits
  - Zilog Z80 e Intel 8086:
    - 14 registros (3 de 8 bits, el resto de 16)
    - Pero sólo uno era un acumulador de propósito general

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ejemplo de acumuladores: Intel 8086/8088

Registros de propósito general		
AH	AL	AX (Acumulador)
BH	BL	BX (Base)
CH	CL	CX (Contador)
DH	DL	DX (Datos)
Registros índices		
SI		**S**ource **I**ndex (Índice origen)
DI		**D**estination **I**ndex (Índice Destino)
BP		**B**ase **P**ointer (Puntero Base)
SP		**S**tack **P**ointer (Puntero de Pila)
Registro de Bandera		
- - - - O D I T S Z - A - P - C		Flags (Banderas)
Registros de Segmentos		
CS		**C**ode **S**egment (Segmento de Código)
DS		**D**ata **S**egment (Segmento de Datos)
ES		**E**xtra**S**egment (Segmento Extra)
SS		**S**tack **S**egment (Segmento de Pila)
Registro apuntador de instrucciones		
IP		**I**nstruction **P**ointer

Figura: Imagen de Wikipedia

Registros *bandera* (vector de booleanos): Overflow, Dirección, Interrupción, Trampa/depuración, Signo, Cero, Acarreo auxiliar, Paridad, Acarreo

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Registros de *propósito general*: Legado de RISC

- Procesadores RISC: A partir de los 1980
- Planteamiento base de instrucciones *sencillas y regulares*
  - Fáciles de codificar en un procesador pequeño (en número de transistores)
  - La arquitectura RISC más conocida hoy: ARM
- Típicamente ≥ 32 registros *largos* (32, 64bits) de propósito general
  - Mas algunos de propósito específico

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Tipos de almacenamiento

¿Cómo representar y guardar *adecuadamente* los datos e instrucciones?

- La memoria rápida es muy cara
  - Pero aún así, mucho más lenta que el procesador: *Cuello de botella de von Neumann* (Backus, 1977)
- La memoria barata es muy lenta (*hasta 1000 veces* más lenta que el procesador)
- El almacenamiento *a largo plazo* es mucho más barato, pero muchísimo más lento
  - Los discos llegan a ser *miles a millones* de veces más lentos que la memoria
- El avance en hardware *juega en contra* de reducir la diferencia

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

La memoria

- El procesador puede referirse directamente a los datos ubicados en la *memoria principal* (*almacenamiento primario*)
  - Indicando su dirección (varias notaciones/mecanismos posibles)
  - Algunos procesadores permiten *realizar operaciones* directamente sobre la memoria
    - Mayormente los basados en acumulador
- Memoria *caché*
  - Acelera operaciones aprovechando la *localidad de referencia*
  - Transparente a la programación (gestionada por el controlador)
  - Varios niveles de caché

Jerarquía de almacenamiento

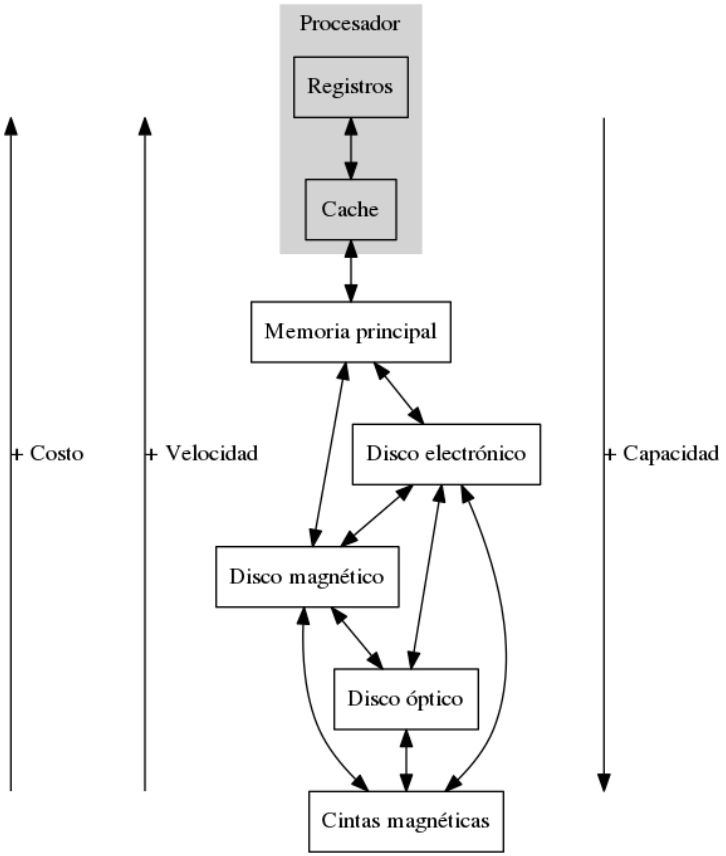


Figura: Diferentes niveles de almacenamiento: Diferencias en costo, velocidad y capacidad

¿Tiempo de acceso? ¿Tasa de transferencia?

Manejaremos muchas mediciones de velocidad. Debemos siempre tener en cuenta:

Tiempo de acceso ¿Cuánto tiempo toma *iniciar* una transferencia? (medido en s, ms, ns). También llamado *latencia*.

Tasa de transferencia Una vez iniciada, ¿a qué velocidad *sostenida* podemos mantenerla? (medido en b/s, Kb/s, Mb/s)

En el *mejor caso*, la transferencia de x bits nos tomará $t_a + xt_t$

Jerarquía de almacenamiento

Cuadro: Velocidad y gestor de los principales niveles de memoria. (Silberschatz, Galvin, Gagne; p.28)

Nivel	1	2	3	4
Nombre	Registros	Cache	Memoria princ.	Disco
Tamaño	<1KB	<16MB	<64GB	>100GB
Tecnología	Multipuerto, CMOS	SRAM CMOS	CMOS DRAM	Magnética
Acceso (ns)	0.25-0.5	0.5-25	80-250	5,000,000
Transf (MB/s)	20,000-100,000	5,000-10,000	1,000-5,000	20-150
Administra	Compilador	Hardware	Sist. Op.	Sist. Op.
Respaldado en	Cache	Memoria princ.	Disco	CD o cinta

Almacenamiento *primario* y *secundario*

- El procesador *sólo puede manejar directamente* a la memoria principal
  - Se le conoce también como *almacenamiento primario*
  - Sólo éste es parte de lo que la arquitectura von Neumann llama *computadora*
- Discos, cintas, almacenamiento *estado sólido* son *almacenamiento secundario*
  - Todas las computadoras lo manejan a través de *controladores*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Canales y puentes

- Los componentes *no directamente referenciables* de un sistema se comunican a través de *canales* (*buses*)
  - Líneas de comunicación entre el procesador y el *chipset*
- Acomodo *muy frecuente* en sistemas x86 recientes[2]:

  Puente norte (Northbridge) Conectado directamente al CPU, encargado de los buses de alta velocidad y los dispositivos fundamentales para el inicio del sistema — Memoria, video (AGP)

  Puente sur (Southbridge) Controla el resto de los dispositivos del sistema; de él se desprenden varios buses (SCSI / SATA / IDE, PCI / PCIe, USB / Firewire, puertos *heredados*)

[2] Después del 2010, las funciones del *Northbridge* han tendido a ser absorbidas por el CPU mismo

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Canales y puentes

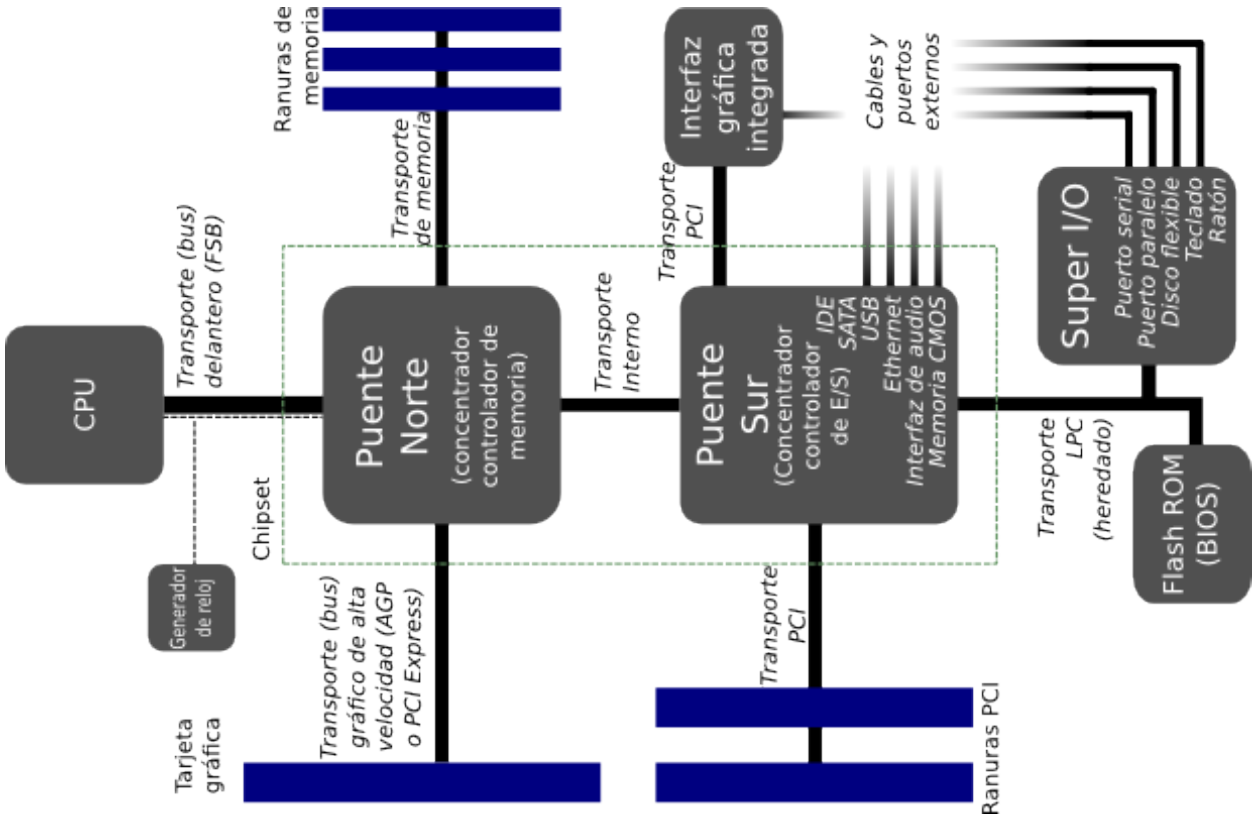


Figura: Diagrama de la comuniacación entre componentes de un sistema de cómputo basado en *puente norte* y *puente sur*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Por qué tantos canales?

- Frecuencia acorde a distintas categorías de dispositivos
  - Criterio económico: Más barato usar señalización más lenta
  - Distintos mecanismos de acceso
- Permitir transferencias paralelas, agregarlas conforme subimos la jerarquía
  - Jerarquizar la comunicación
- Pero... Cuando el sistema requiere transferir datos de o hacia dispositivos pasando por el mismo bus, frecuentemente ocurre contención
  - Algunos canales, como el USB, permiten *hasta 127* dispositivos conectados *serialmente*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ejemplo: Chipset Intel 875 (2003)

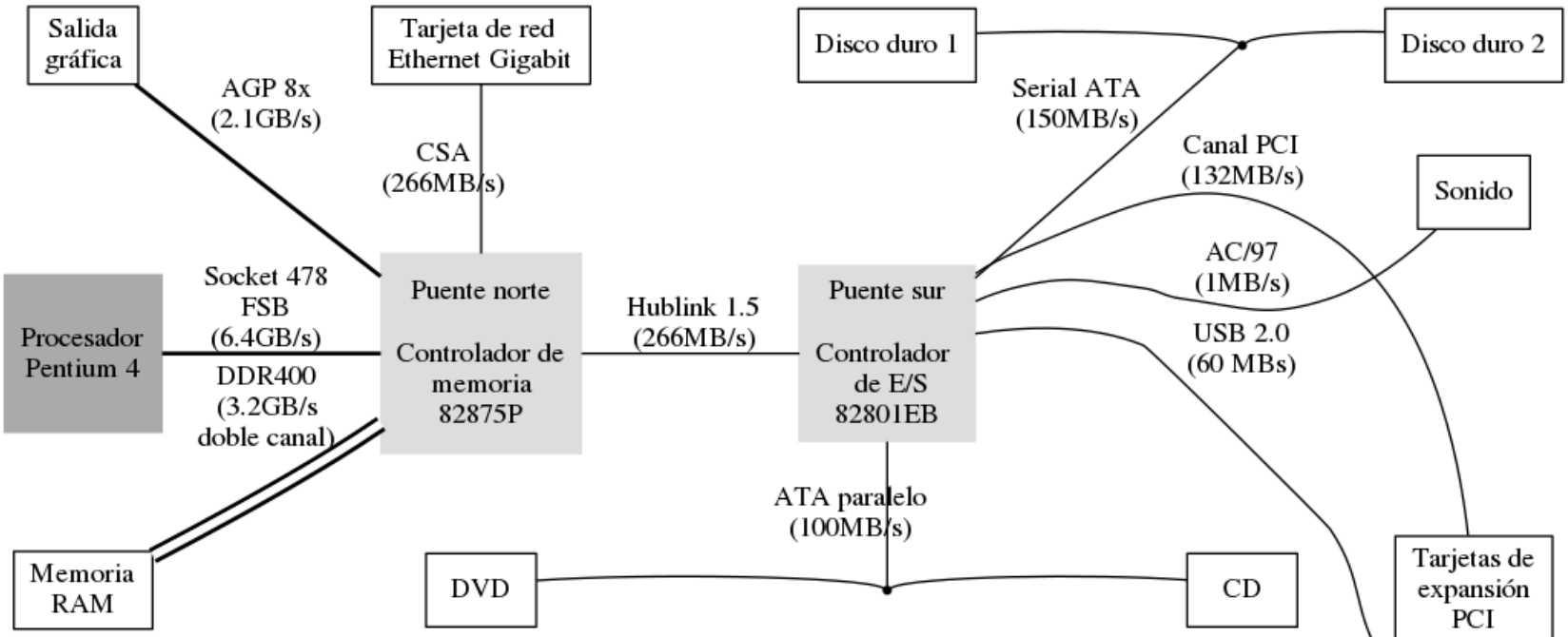


Figura: Diagrama de la comuniacación entre componentes de un sistema de cómputo basado en el chipset Intel 875 (Pentium 4, 2003)

Otros arreglos

- El ejemplo presentado es para un sistema de escritorio típico
  - Y bastante viejo ya
- En entornos de alto rendimiento, puede haber múltiples canales entre los componentes
  - Para reducir el impacto de la contención
  - Particularmente en los componentes con mayor demanda de ancho de banda
- En entornos móviles / de bajo costo, se hacen *concesiones* permitiendo mayor contención
  - P.ej. estructurar la comunicación a todos los periféricos alrededor del bus USB
  - Frecuente en equipos ARM

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ejemplo: Raspberry Pi

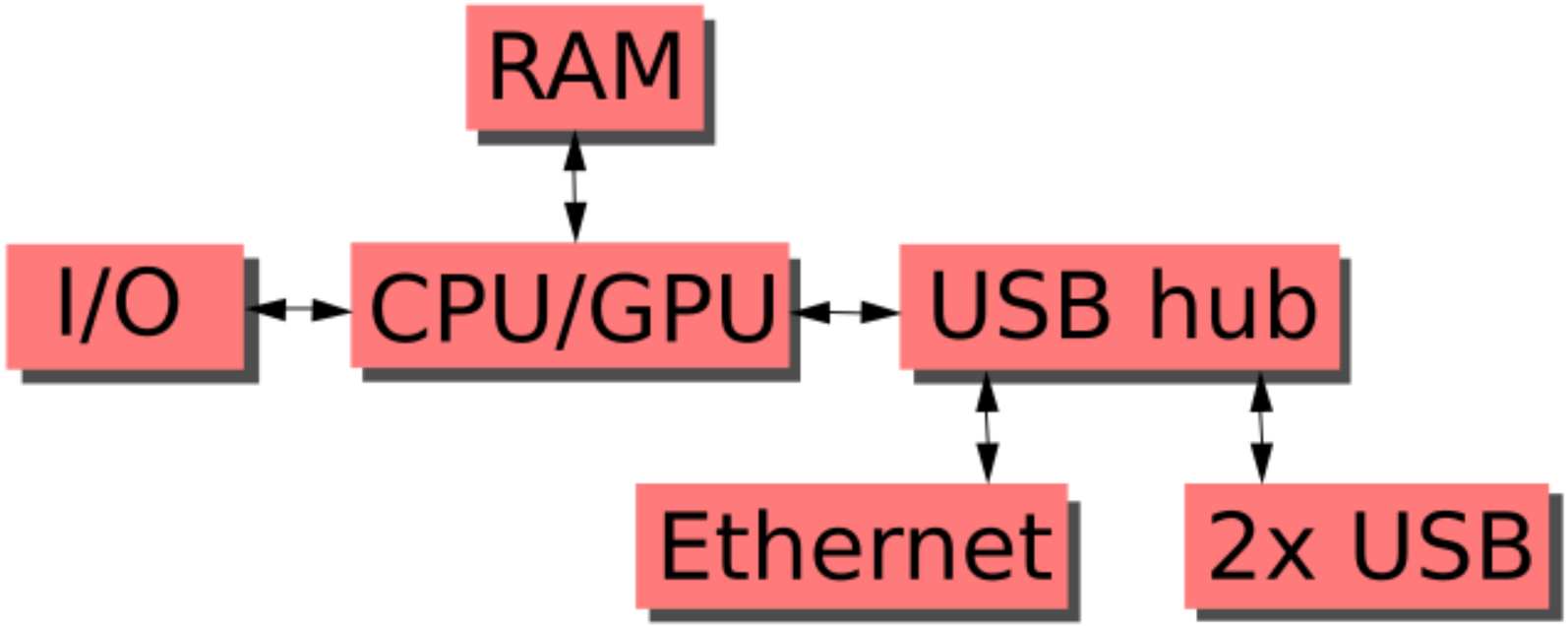


Figura: Diagrama de bloques de la popular plataforma ARM *Raspberry Pi* de primera generación (Imagen de Wikipedia)

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El procesador y los *eventos* externos

- El procesador no tiene cómo reaccionar *internamente* a eventos que ocurran en el sistema
- La ejecución es lineal: Avanza por la lista de instrucciones del programa
- Lo que permite el manejo de toda la E/S, interactividad, multiprogramación es el mecanismo de *interrupciones* y *excepciones*.

Interrupciones y excepciones

Interrupción Todo evento recibido por el sistema, de origen *externo* al flujo de la ejecución

- Actividad en la red
- Teclado o mouse
- Alarma del temporizador
- Datos del disco listos

Excepción Eventos inesperados originados por el flujo del proceso

- División sobre cero
- Instrucción ilegal
- Acceso a memoria no direccionada
- También conocidas como *trampas* (*traps*)

Manejo de interrupciones y excepciones

- Todo *evento* es recibido por el sistema operativo (no por los procesos)
- Cuando ocurre cualquier *evento*, el hardware *lanza una interrupción* que interrumpe el flujo de ejecución
- Rutina de *manejo de interrupciones*
  - Grabar estado del proceso desplazado y *cambiar contexto*
  - Atender la interrupción en *modo privilegiado* (¡el menor tiempo posible!)
  - Una vez procesada, volver a invocar al *planificador*

¿Cuándo una interrupción es *no enmascarable*?

- Depende de la arquitectura y los objetivos del sistema
- Algunos ejemplos:
  - Error de paridad en la memoria (IBM PC)
  - Llamadas a hardware incompatible (primeros *clones* de IBM; llamada atrapada y procesada por el manejador de interrupciones en el BIOS)
  - Diversas combinaciones de teclas para invocar a un reinicio (o lanzar un depurador)
  - Consolas de 8 bits (Nintendo NES): Bloquear modificaciones al buffer de pantalla durante el refresco vertical

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Llamadas al sistema

- De cierto modo análogas/complementarias a las interrupciones
- Mecanismo para que un proceso *solicite un servicio* al sistema operativo
- Cada sistema operativo *expone* un diferente juego de llamadas al sistema a través de su API
- Es en buena medida lo que determina la *compatibilidad de código* entre sistemas operativos
  - Contraposición: Compatibilidad binaria
  - APIs implementados por diversos sistemas: POSIX, Win32

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Flujo de control en una llamada al sistema

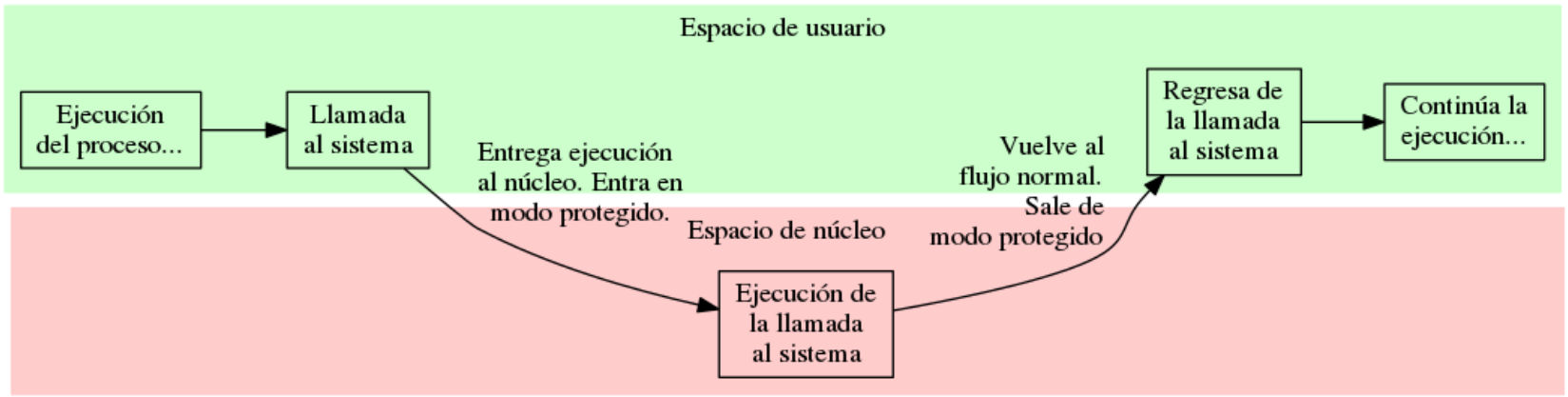


Figura: Transición del flujo entre espacio usuario y espacio núcleo en una llamada al sistema

Tipos de llamadas al sistema (1)

Lista incompleta, meramente ejemplificando

Control de procesos Crear o finalizar un proceso, obtener atributos del proceso, esperar cierto tiempo, asignar o liberar memoria, etc.

Manipulación de archivos Crear, borrar o renombrar un archivo; abrir o cerrar un archivo existente; modificar sus *metadatos*; leer o escribir de un *descriptor de archivo* abierto, etc.

Tipos de llamadas al sistema (2)

Lista incompleta, meramente ejemplificando

Manipulación de dispositivos Solicitar o liberar un dispositivo; leer, escribir o reposicionarlo, y otras varias. Muchas de estas llamadas son análogas a las de manipulación de archivos, y varios sistemas operativos las ofrecen como una sola.

Mantenimiento de la información Obtener o modificar la hora del sistema; obtener detalles acerca de procesos o archivos, etc.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Tipos de llamadas al sistema (3)

Lista incompleta, meramente ejemplificando

Comunicaciones Establecer una comunicación con determinado proceso (local o remoto), aceptar una solicitud de comunicación de otro proceso, intercambiar información sobre un canal establecido

Protección Consultar o modificar la información relativa al acceso de objetos en el disco, otros procesos, o la misma sesión de usuario

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Depuración por *trazas*

- La mayor parte de los sistemas operativos ofrecen programas que ayudan a *depurar la ejecución* de otros programas
- Pueden *envolver* al API del sistema, y permitir seguir la *traza* (*trace*) de la ejecución de un proceso
- Por ejemplo:
  - `strace` en Linux
  - `truss` en Unixes históricos
  - `ktrace`, `kdump` en *BSD
  - `dtrace` en Solaris $\geq$ 10 (2005)
- Permite entender *buena parte* de lo que realiza un proceso — Prácticamente, toda su interacción con el sistema
  - A veces, demasiada información

Ejemplo: $ strace pwd

```
execve("/bin/pwd", ["pwd"], [/* 43 vars */]) = 0
brk(0)                                  = 0x8414000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such
    file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb773d000
                (...)
getcwd("/home/gwolf/vcs/sistemas_operativos", 4096) = 36
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1),
    ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb773c000
write(1, "/home/gwolf/vcs/sistemas_operati"...,
    36/home/gwolf/vcs/sistemas_operativos
) = 36
close(1)                                = 0
munmap(0xb773c000, 4096)                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Acceso directo a memoria (DMA)

Problema: En la sección de un proceso en que está *limitado por entrada-salida*, la transferencia de información se vuelve cuello de botella

- Modo *entrada/salida programada*
- Gran cantidad y frecuencia de interrupciones
- Imposible realizar trabajo real

Respuesta: *Acceso Directo a Memoria*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Acceso directo a memoria (DMA)

- Orientado a dispositivos de gran ancho de banda
  - Unidades de disco
  - Multimedia
  - Red
  - Memoria y caché
- Transferencia en bloques, empleando un *controlador*
  - Dirección física base de memoria
  - Cantidad de datos a transferir
  - *Puerto* del dispositivo
  - Dirección de la transferencia (*desde* o *hacia* memoria)
- Limitante: Contención en el bus de memoria

Coherencia de caché

¿Qué problema puede causar una transferencia *no iniciada por el procesador*? (O por *otros procesadores*)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Coherencia de caché

¿Qué problema puede causar una transferencia *no iniciada por el procesador*? (O por *otros procesadores*)

- Incongruencia entre la memoria real y *páginas* de caché existentes

  Caché coherente Mecanismos *en hardware* que notifican a los controladores de caché que las páginas están *sucias*

  No coherente El sistema operativo debe realizar esta operación
- Sistemas híbridos en lo relativo a la coherencia
  - Caché de nivel superior (en el procesador) no coherente, cachés inferiores coherentes

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Multiprocesamiento y multiprogramación

Conceptos relacionados, empleados coloquialmente de forma intercambiable, pero muy distintos:

Multiprogramación *Ilusión* de estar ejecutando varios procesos al mismo tiempo, por medio de la *alternación rápidamente* entre ellos

- En un principio, no tan rápidamente; hoy en día, cientos a miles de veces por segundo

Multiprocesamiento Entorno donde hay más de un procesador central (CPU).

- Por muchos años, reservado a usos muy especializados
- Hoy en día, la norma.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Multiprogramación, multiprocesamiento, híbrido...

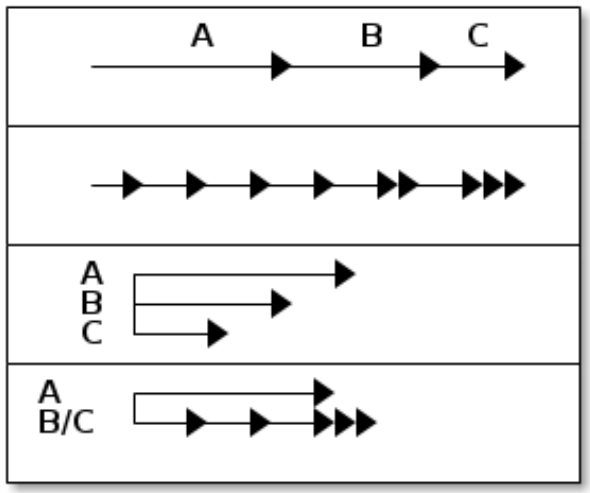


Figura: Esquema de la ejecución de tres procesos en un sistema secuencial, multiprogramado, multiprocesado, e híbrido

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Formalizando un poco

- *Sólamente* un sistema multiprocesador tiene la capacidad *real* de atender *simultáneamente* a diversos procesos
- Hoy en día es muy raro aplicar *de forma exclusiva* a cualquiera de estas modalidades
  - Multiprogramado puro: Sigue siendo común en equipos no-multiprocesados
  - . . . Pero cada vez son menos

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Acostumbrémonos, porque...

El tema que más recurrentemente abordaremos en el curso es *la complejidad* que proviene tanto de la multiprogramación como de la multitarea

Particularmente la abordaremos en las dos siguientes unidades: *Administración de procesos* y *Planificación de procesos*.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

La irrupción del multiprocesamiento

- Existe desde los 1960
- Entornos de alto rendimiento, *muy especializados*
  - Requieren una programación hecha *con cuidado especial*
- Hace apenas 10-15 años, muchos sistemas operativos no detectaban siquiera más de un procesador
  - Incluso de rango servidor
  - Era muy raro encontrarlos en hardware de uso frecuente
- Pero hacia el 2005... Nos alcanzó Gordon E. Moore

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Gordon E. Moore?

- Ingeniero electrónico, fundador de *Fairchild Semiconductor*
- Publicó en 1965 *Cramming more components onto integrated circuits* (*Apretujando más componentes en circuitos integrados*)
- Observando el desarrollo desde 1959, predice que cada año se duplicará la cantidad de transistores en los circuitos integrados, al menos por los próximos 10 años

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

La Ley de Moore, 1965

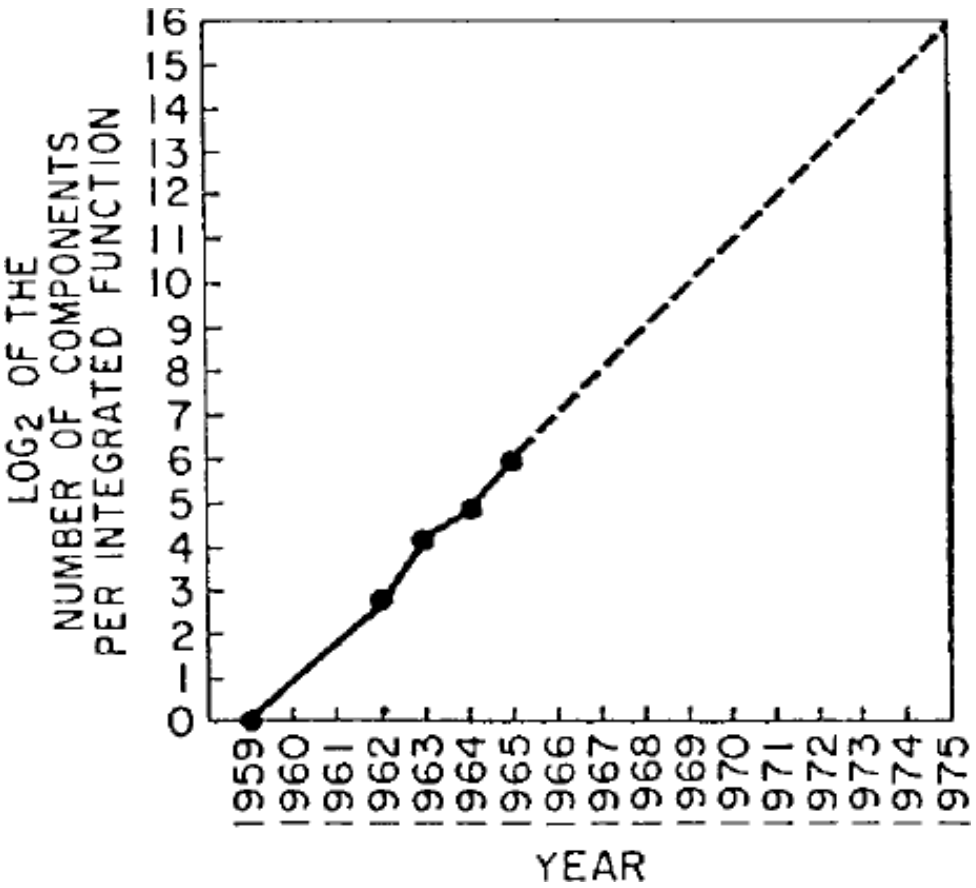
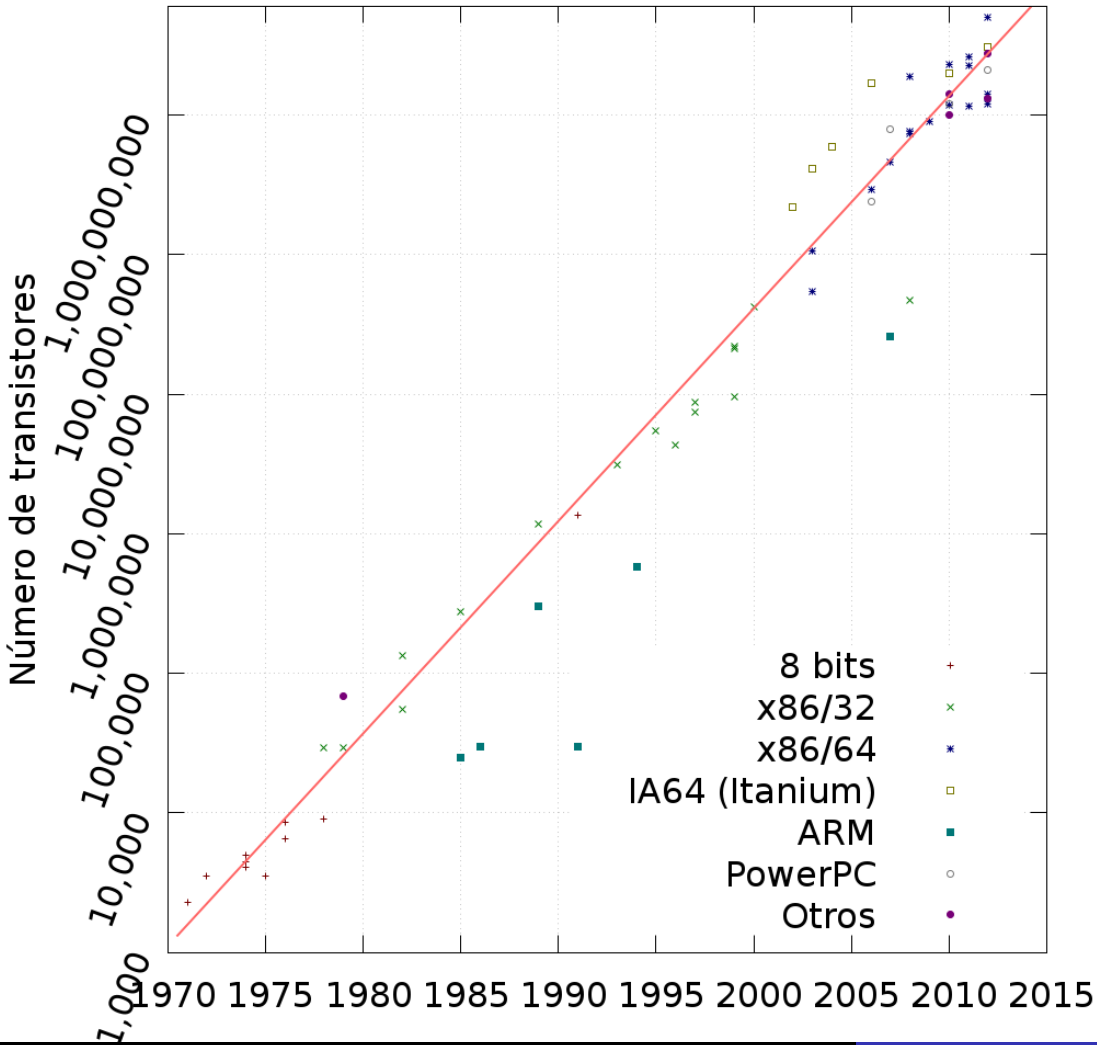


Figura: La *Ley de Moore*, en su artículo publicado en 1965, prediciendo la miniaturización por diez años

La Ley de Moore, hoy



Y a cinco decadas, la Ley de Moore se sigue cumpliendo...

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Moore y el multiprocesamiento?

¿Qué tiene que ver la Ley de Moore y el multiprocesamiento?

- Hasta $\approx$ 2005, la velocidad de los CPUs crecía constantemente
  - En el ámbito comercial, excediendo los 3 GHz
  - Llevando a problemas serios de calentamiento
- El diferencial de velocidad con el acceso a memoria crece cada vez más
- Hace falta un cambio de estrategia...

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Chips multiprocesador

- Los principales fabricantes de CPUs comenzaron a diseñar chips conteniendo más de un *núcleo* — Esto es, más de un CPU independiente
  - (Y algunas facilidades adicionales — Por ejemplo, mucho mayor caché *dentro del mismo* chip)
- El reloj de los procesadores no sólo no ha crecido, sino que en general *se ha reducido* a cerca de 1GHz
  - Obviamente, el rendimiento y la densificación siguen avanzando
- Pero ahora la ganancia de velocidad ya no llega en automático con el nuevo hardware
  - El sistema operativo tiene que saber *aprovechar* al multiprocesamiento

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Cómo multiprocesamos?

Hoy en día, cuando hablamos de multiprocesamiento, *casi* siempre hablamos de multiprocesamiento simétrico

. . . Pero no es el único tipo de multiprocesamiento que hay

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Multiprocesamiento simétrico (SMP)

- Todos los procesadores *son iguales*
- Pueden realizar las mismas operaciones en el mismo tiempo
- Un proceso podría *migrarse* de un procesador a otro de forma transparente
  - Únicamente manteniendo información básica
- Tienen *acceso a la misma memoria*
  - Aunque cada uno puede tener su propio caché) → ¡Necesidad de coherencia!

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Multiprocesamiento asimétrico

Puede haber varios puntos de asimetría:

- Distinta *arquitectrura* → Típicamente algunos se dedicarán a una tarea específica
- Coprocesadores, o procesadores coadyuvantes
  - Caso de las tarjetas gráficas (GPUs)
  - Podrían verse *casi* como una red alta velocidad de computadoras *independientes*
- Memoria y responsabilidades muy distintas a las del sistema central

Acceso no-uniforme a memoria (NUMA)

(*Non-Uniform Memory Access*)

- Procesadores de la misma arquitectura/tipo
  - Podrían formar parte de un sistema SMP
- Pero con *afinidad* a bancos específicos de memoria
  - Memoria *cercana* o *lejana*
  - Al *planificar* los procesos, se destinan los procesadores cercanos
  - Hay un *canal compartido* de memoria
- Típicamente computadoras grandes con *nodos* o *blades*
- Pueden ubicarse como en un punto medio entre SMP y el *cómputo distribuído*
  - Cómputo distribuído fuertemente acoplado

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Más allá: Cómputo distribuído

- Distribuir un proceso de cómputo para ser realizado *entre computadoras independientes*
  - Más formalmente: Entre procesadores *que no comparten memoria* (almacenamiento primario)
- Dos modalidades principales (mas un *adosado*):
  - Cúmulos (*clusters*)
  - Mallas (*grids*)
  - Cómputo *en la nube*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Cúmulos (clusters)

- Computadoras conectadas por una red local de alta velocidad
- Cada una corre su propia instancia de sistema operativo y programas
- Principales orientaciones:

  Alto rendimiento Cómputo matemático, cálculos...
  Alta disponibilidad Prestación de servicios críticos
  Balanceo de cargas Atención a solicitudes complejas, que pueden saturar a servidores individuales
- Típicamente son equipos homogéneos y dedicados
- Muy comunes en universidades; bajo costo, altas prestaciones

Mallas (grids)

- Computadoras distribuidas geográficamente
- Conectadas sobre Internet (o redes de área amplia)
- Pueden ser heterogéneas (en capacidades y en arquitectura)
- Presentan *elasticidad* para permitir conexiones/desconexiones de nodos en el tiempo de vida de un cálculo

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Mallas (grids)

- Computadoras distribuidas geográficamente
- Conectadas sobre Internet (o redes de área amplia)
- Pueden ser heterogéneas (en capacidades y en arquitectura)
- Presentan *elasticidad* para permitir conexiones/desconexiones de nodos en el tiempo de vida de un cálculo

¿Quién quiere presentar el ejemplo de alguna malla?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Cómputo en la nube

- Caso específico y *de moda*
- Partición de recursos (*cliente-servidor*)
- Orientado a la *tercerización* de servicios específicos
- La *implementación* de un servicio deja de ser relevante → Procesos *opacos*
- Conceptos relacionados:
  - *Servicios Web*
  - Plataforma como servicio (*PaaS*)
  - Software como servicio (*SaaS*)
  - Infraestructura como servicio (*IaaS*)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Límites del paralelismo

- Gene Amdahl (1967) observó que la ganancia derivada de paralelizar el código llega a un límite natural: El tiempo requerido por la *porción secuencial* del código
- La *porción paralelizable* puede repartirse entre varios procesadores, pero siemprehay una importante proporción no paralelizable
  - Inicialización
  - Puntos de control/acumulación de datos
  - Interacción con el usuario
  - etc.

Ley de Amdahl

$T(1)$ Tiempo de ejecución del programa con un sólo procesador

$T(P)$ Tiempo de ejecución con P procesadores

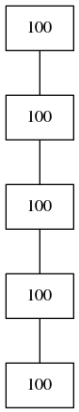
t_s Tiempo requerido por la porción secuencial

$t_p(P)$ Tiempo requerido para la porción paralela entre P procesadores

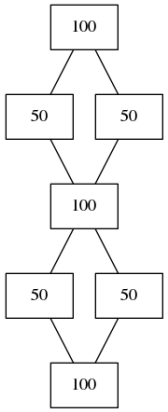
$$g = \frac{T(1)}{T(P)} = \frac{t_s + t_p(1)}{t_s + \frac{t_p(1)}{P}}$$

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

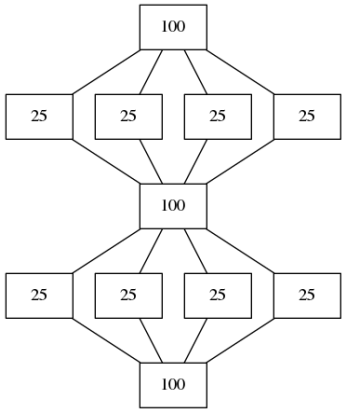
Ilustrando la ley de Amdahl (1)



Procesadores: 1, $t = 500$



Procesadores: 2, $t = 400$, ganancia: 1.25x



Procesadores: 4, $t = 350$, ganancia: 1.4x

Ley de Amdahl: ejecución de un programa con 500 unidades de tiempo total de trabajo con uno, dos y cuatro procesadores.

Ilustrando la ley de Amdahl (2)

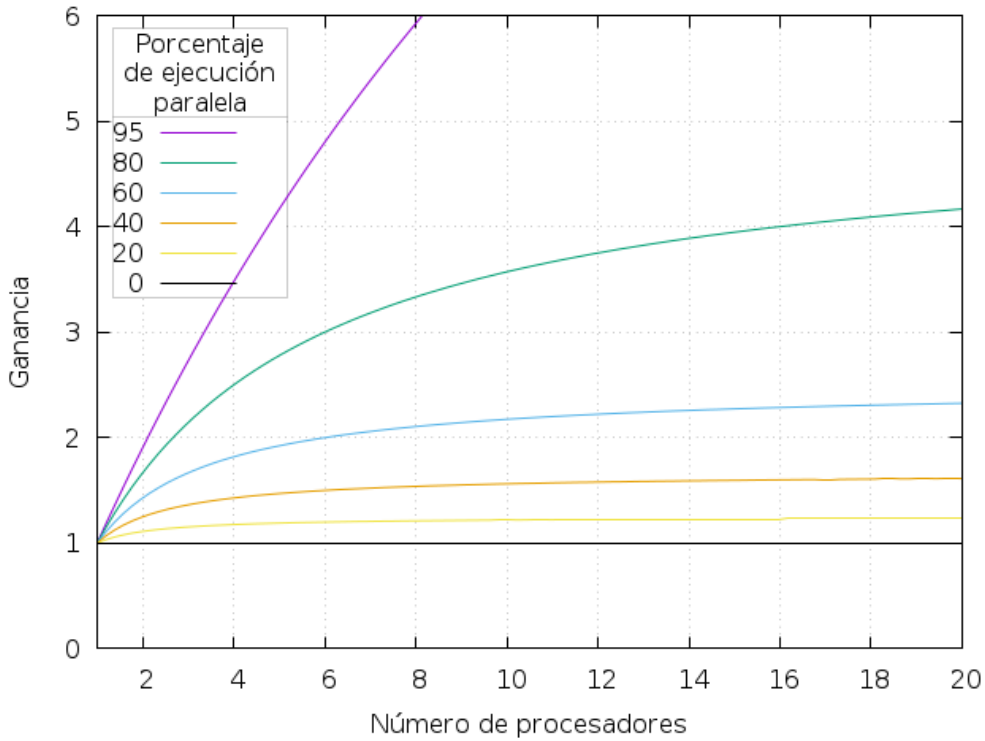


Figura: Ganancia máxima al paralelizar un programa, según la Ley de Amdahl

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Desmenuzando a Amdahl

- El paralelismo puede aumentar fuertemente nuestro rendimiento
  - ... ¿O no?
- Incluso con un 95 % de ejecución paralela, se llega a un techo de ganancia al llegar a los 20 CPUs
  - Cada procesador tiene un costo económico no trivial
  - ¿Vale realmente la pena ir migrando hacia un paralelismo cada vez mayor?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Observación de Gustafson

- Por muchos años, la ley de Amdahl disminuyó el interés en el cómputo paralelo
- En 1988, John Gustafson obtuvo ganancias superiores a 1020x en una supercomputadora de 1024 procesadores
- Publicó una observación que hizo re-evaluar este conocimiento
- La modificación no (sólo) será al tiempo de cómputo, *sino que al problema mismo*
- Un artículo de apenas una cuartilla y lenguaje muy sencillo rompió el *bloqueo mental* contra el paralelismo

Observación de Gustafson

(. . .)Asumen implícitamente que el tiempo que se ejecuta en paralelo es independiente del número de procesadores, lo cual virtualmente nunca ocurre de este modo. Uno no toma un problema de tamaño fijo para ejecutarlo en varios procesadores como no sea para hacer un ejercicio académico; en la práctica, el tamaño del problema crece con el número de procesadores.

Al obtener procesadores más poderosos, el problema generalmente se expande para aprovechar las facilidades disponibles. Los usuarios tienen control sobre cosas como la resolución de la malla, el número de pasos, la complejidad de los operadores y otros parámetros que usualmente se ajustan para permitir que el programa se ejecute en el tiempo deseado.

Por tanto, podría ser más realista decir que el tiempo de ejecución, no el tamaño del problema, es constante.