Ingeniería de Servidores

grado en ingeniería informática

Resumen del temario

Autor
Carlos Sánchez Páez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

Índice de figuras

1. Tema 1. Introducción a la Ingeniería de Servidores

1.1. ¿Qué es un servidor?

Un **sistema informático** es un conjunto de elementos *hardware* (componentes físicos), *software* (componentes lógicos) y *peopleware* (recursos humanos) que permiote obtener, procesar y almacenar información.
Los sistemas informáticos se pueden clasificar atendiendo a varios factores:

- Según el nivel de paralelismo (*SISD*,*SIMD*,*MISD* o *MIMD*)
- Según su uso (propósito general o específico)

* Si son servidores, según la arquitectura de servicio (sistema aislado, cliente-servidor, n capas o cliente-cola-cliente)

1.1.1. Según su nivel de paralelismo

- *SISD*: **S**ingle **I**nstruction **S**ingle **D**ata
- *SIMD*: **S**ingle **I**nstruction **M**ultiple **D**ata
- *MISD*: **M**ultiple **I**nstruction **S**ingle **D**ata
- *MIMD*: **M**ultiple **I**nstruction **M**ultiple **D**ata

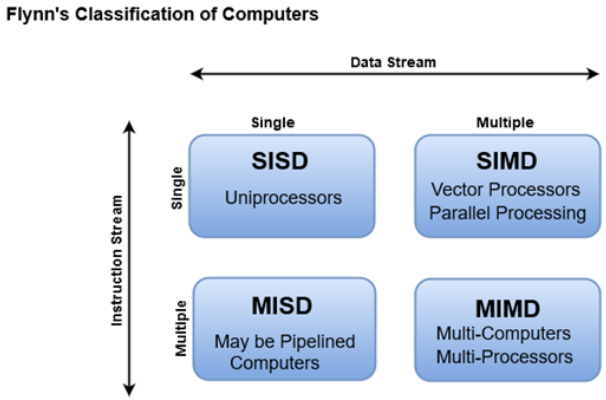

Figura 1: Sistemas informáticos según su nivel de paralelismo

1.1.2. Según su uso

- De **uso general**: sirven para ejecutar diversas aplicaciones (PC sobremesa, portátil).
- De **uso específico**: ejecutan una función concreta.
  - **Sistemas empotrados (*embedded systems*)**. Son sistemas acoplados a otro dispositivo o aparato que realizan una o varias funciones dedicadas. Suelen tener grandes restricciones de tamaño, tiempo de respuesta, etc. Ejemplo: taxímetro, cámara de vigilancia, lavadora, etc.

- **Servidores**. Son sistemas informáticos que forman parte de una red y proporcionan servicios a otros sistemas informáticos (clientes). Puede ser cualquier computador o *clúster* (agrupación de computadores que son percibidos externamente como uno solo).

Hay varios tipos de servidores:

- **Servidor de archivos**: permite el acceso remoto a archivos almacenados o directamente accesibles por él.
- **Servidor web**: almacena documentos HTML, imágines, etc. y distribuye el contenido a los clientes que lo soliciten.
- **Servidor de base de datos**: provee servicios de base de datos a otros programas o sistemas.
- **Servidor de *e-commerce***: cumple o procesa transacciones comerciales. Valida al cliente y genera un pedido al servidor de bases de datos.
- **Servidor de impresión**: controla una o más impresoras y acepta trabajos de impresión de los clientes de la red.
- **Servidor de correo electrónico**: almacena, envía, recibe, etc. correos electrónicos para los clientes de la red.

1.1.3. Según su arquitectura de servicio

- **Sistema aislado**: es un sistema que no interactúa con otros. La arquitectura es monolítica y no se distribuye la información
- **Arquitectura cliente-servidor**: las tareas se reparten entre los servidores (reciben solicitudes) y los clientes (remiten solicitudes). Los nodos son los servidores y los clientes.
- **Arquitectura cliente-servidor de n capas**: es una arquitectura cliente-servidor que tiene n tipos de nodos en la red, por lo que se mejora la distribución de la carga (mejorando la escalabilidad). Sus principales puntos negativos son la sobrecarga de la red que conlleva y la dificultad de programación y administración.
  Ejemplo de arquitectura de 3 capas:
  1. Servidores que interactúan con los clientes.
  2. Servidores de *e-commerce* que procesan los datos para los servidores de la capa 1.
  3. Servidores de bases de datos que buscan, gestionan y almacenan los datos para los servidores de la capa 2.
- **Arquitectura cliente-cola-cliente**: el servidor únicamente pone en contacto a los clientes y sincroniza el sistema, mientras que los clientes se encargan de cooperar para realizar la función necesaria. La arquitectura *P2P* está basada en este concepto. Ejemplos: *Skype*, *eMule*, *BitTorrent*.

1.2. Fundamentos de la Ingeniería de Servidores

Un servidor se compone de:

- Recursos físicos (placa base, memoria, CPU, etc.).
- Recursos lógicos (sistema operativo, aplicaciones).
- Recursos humanos (administración).
- Requisitos funcionales (prestaciones, seguridad, mantenimiento, disponibilidad, extensibilidad, escalabilidad, coste, fiabilidad...)

Las **prestaciones** cuantifican la velocidad con la que se realiza una determinada cantidad de trabajo o carga (*workload*). Aquel servidor que realiza la misma carga de trabajo en menor tiempo es el que mayores prestaciones tiene.
Las medidas fundamentales para medir las prestaciones de un servidor son:

- **Tiempo de respuesta** o **latencia**. Tiempo desde que se solicita una tarea al servidor hasta que finaliza. Por ejemplo: tiempo de ejecución de un programa, de acceso al disco, etc.
- **Productividad** (*throughput*) o **ancho de banda** (*bandwidth*). Cantidad de trabajo realizado por el servidor por unidad de tiempo. Por ejemplo: programas ejecutados por hora o páginas por hora servidas en el caso de un servidor web.

Los principales elementos que afectan a las prestaciones de un servidor son los siguientes:

- Hardware del sistema (características y configuración).
- Parámetros del sistema operativo (configuración de memoria virtual, políticas de planificación...) y diseño de los programas (acceso a E/S, fallos de caché...).
- Actualización de componentes (reemplazar por dispositivos más rápidos) y configuración de dispositivos.
- Ajuste o sintonización del SO y los programas.
- Distribución de carga (*load balancing*): dar mayor carga a los dispositivos más rápidos.

1.2.1. Requisitos funcionales de un servidor

- **Disponibilidad** (*Availability*). Un servidor está disponible si está en estado operativo.
  El tiempo de inactividad (*downtime*) es la cantidad de tiempo en la que el sistema no está disponible. Puede ser de dos tipos:
  - Planificado (actualizaciones de hardware o software que no se puedan realizar en caliente).
  - No planificado (causa de algún fallo). Si el sistema tiene *alta disponibilidad* será *tolerante a fallos*.

Algunas soluciones para aumentar la disponibilidad son:

  - Reemplazo en caliente de los componentes (*hot-swapping*).
  - Sistemas redundantes de discos (*RAID*), alimentación y red.
  - Sistemas distribuidos.

- **Fiabilidad** (*Reliability*). Un sistema es fiable cuando realiza su actividad sin errores. El **MTTF** (***M**ean **T**ime **T**o **F**ailure*) es el tiempo medio que tiene un sistema hasta que ocurre un error. Los mecanismos para asegurar la fiablilidad son la comprobación (*checksums*, bits de paridad, etc.)

- **Seguridad**. Un servidor debe ser seguro ante:

  - La incursión de individuos no autorizados (*confidencialidad*).
  - La corrupción o alteración no autorizada de datos (*integridad*).
  - Las interferencias (ataques) que impidan el acceso a los recursos.

  Las principales soluciones son la encriptación de datos, el uso de cortafuegos y la autenticación segura de usuarios.

- **Extensibilidad-expansibilidad**. Es la facilidad que ofrece el sistema para aumentar sus características o recursos (tener bahías libres para discos duros o memoria, uso de sistemas operativos modulares, uso de interfaces de entrada y salida estándar, etc.). Cualquier solución que facilite la escalabilidad facilita también la extensibilidad.

- **Escalabilidad**. Un servidor es escalable si sus prestaciones pueden aumentar significativamente ante un incremento significativo de la carga. Las soluciones más usadas son la virtualización, la programación paralela y la tecnología *cloud*. Todos los sistemas escalables son extensibles (pero no al revés).

- **Mantenimiento**. Son las acciones que sirven para prolongar el funcionamiento correcto del sistema. Es importante que el servidor sea fácil de mantener (sistema operativo actualizado, garantía de componentes, copias de seguridad, etc.).

- **Coste**. Debemos escoger un diseño que sea asequible y se ajuste al presupuesto teniendo en cuenta el coste hardware, software, de actualizaciones, de personal, de proveedores de red, de alquiler de local y de **eficiencia energética**. La eficiencia energética es clave ya que reduce en gran cantidad los costes (consumo de potencia, refrigeración) y además ayuda a preservar el medio ambiente. Las soluciones más comunes son:

  - Ajuste automático de potencia de los componentes según la carga.
  - *Free cooling*, aprovechamiento de las bajas temperaturas exteriores para tener refrigeración gratuita.
  - Fusionar varios servidores con poco uso en un único servidor.

1.3. Comparación conjunta entre prestaciones y coste

El computador de mejores prestaciones (el más rápido) para un determinado conjunto de programas será el que lo ejecute en menor tiempo.

1.3.1. Tiempos de ejecución mayores o menores

Sea t_A el tiempo de ejecución de un programa en la máquina A (ídem para t_B).

- t_A es $\frac{t_A}{t_B}$ veces t_B. Por ejemplo, si $t_A = 10s$ y $t_B = 5s$, t_A es $\frac{10}{5} = 2$ veces t_B (el doble).
- Igualmente, t_B es $\frac{5}{10} = 0{,}5$ veces t_A (la mitad).

El **cambio relativo de t_A con respecto a t_B**; $\Delta t_{A,B}(\,\%)$ viene dado por:

$$t_A = t_B + \frac{\Delta t_{A,B}(\,\%)}{100} * t_B$$

Es decir, es el porcentaje que le falta a t_B para ser t_A.
En el ejemplo anterior el cambio relativo de t_A con respecto a t_B es del 100 %. Es decir, a t_B le falta un 100 % de sí mismo (el doble) para llegar a ser t_A.
De igual forma, el cambio relativo de t_B con respecto a t_A es del $-50\,\%$ (a t_A le sobra la mitad para ser t_B).
Usando el lenguaje común podríamos traducir los resultados en:

- t_A es un 100 % mayor que t_B y t_B es un 50 % menor que t_A.
- t_A es 2 veces mayor que t_B (¡ojo! ¡¡"1 vez mayor"significa igual!!).

La velocidad de ejecución de una máquina será inversamente proporcional al tiempo que tarda en ejecutarse. Teniendo ésto en cuenta podemos definir la aceleración (*speedup*) de A con respecto a B de la siguiente forma:

$$S_B(A) = \frac{v_A}{v_B} = \frac{t_B}{t_A}$$

El cambio relativo de v_A con respecto de v_B (es decir, el porcentaje de cambio en velocidad al reemplazar A por B) viene dado por:

$$\Delta v_{A,B}(\,\%) = (S_B(A) - 1) * 100$$

Supongamos que para un programa, $t_A = 36s$ y $t_B = 45s$.
La ganancia de velocidad de A con respecto a B sería:

$$S_B(A) = \frac{v_A}{v_B} = \frac{t_B}{t_A} = \frac{45s}{36s} = 1{,}25$$

El cambio relativo de v_A con respecto a v_B (es decir, el porcentaje de mejora) viene dado por:

$$\Delta v_{A,B}(\,\%) = (S_B(A) - 1) * 100 = (1{,}25 - 1) * 100 = 0{,}25 * 100 = 25\,\%$$

Usando el lenguaje común diríamos que:

- A es 1.25 veces más rápida que B.
- A es un 25 % más rápida que B

De igual forma:

$$S_A(B) = \frac{t_A}{t_B} = \frac{36s}{45s} = 0{,}8$$

$$\Delta v_{B,A}(\,\%) = (S_A(B) - 1) * 100 = (0{,}8 - 1) * 100 = -0{,}2 * 100 = -20\,\%$$

Es decir, B es un 20 % más lenta que A.

¡¡Ojo!!
A es un x % más rápida que B $\not\Longrightarrow$ B es un x % más lenta que A.
Por ejemplo, 10 es un 100 % mayor que 5 $\not\Longrightarrow$ 5 es un 100 % menor que 10

Supongamos que en el ejemplo anterior el computador A cuesta 625 € y el computador B, 550 €.

- A es $\frac{625}{550} = 1{,}14$ veces más caro que B (un 14 % más caro).
- ¿Cuál ofrece mejor relación prestaciones/precio para nuestro programa?

$$\frac{Prestaciones_A}{Coste_A} = \frac{v_A}{Coste_A} \propto \frac{\frac{1}{t_A}}{Coste_A} = \frac{\frac{1}{36s}}{625€} = 4{,}4 * 10^{-5}\frac{s^{-1}}{€}$$

$$\frac{Prestaciones_B}{Coste_B} = \frac{v_B}{Coste_B} \propto \frac{\frac{1}{t_B}}{Coste_B} = \frac{\frac{1}{45s}}{550€} = 4{,}0 * 10^{-5}\frac{s^{-1}}{€}$$

Pasamos ahora a comparar ambas relaciones:

$$\frac{\frac{Prestaciones_A}{Coste_A}}{\frac{Prestaciones_B}{Coste_B}} = \frac{4{,}4 * 10^{-5}}{4{,}0 * 10^{-5}} = 1{,}1$$

- Por tanto, el computador A presenta una mayor relación prestaciones/coste que B (un 1.1 mayor = un 10 % mayor) para nuestro programa.

1.4. Límites en la mejora del tiempo de respuesta

La mejora dekl tiempo de respuesta no es ilimitada, por lo que debemos saber hacia dónde dirigir los esfuerzos. Por ejemplo, si un programa utiliza la CPU en un 80 % del tiempo y la impresora en un 20 % debemos centrarnos en mejorar la CPU. El planteamiento sería el siguiente:

- Un sistema tarda un tiempo $T_{original}$ en ejecutar un programa monohebra.
- Mejoramos el sistema reemplazando un componente por otro k veces más rápido.
- Este comonente se utilizaba durante una fracción f de $T_{original}$ (si $f = 1$, el componente se utilizaba el 100 % del tiempo).
- ¿Qué ganancia en prestaciones (*speedup*) hemos conseguido con el cambio?

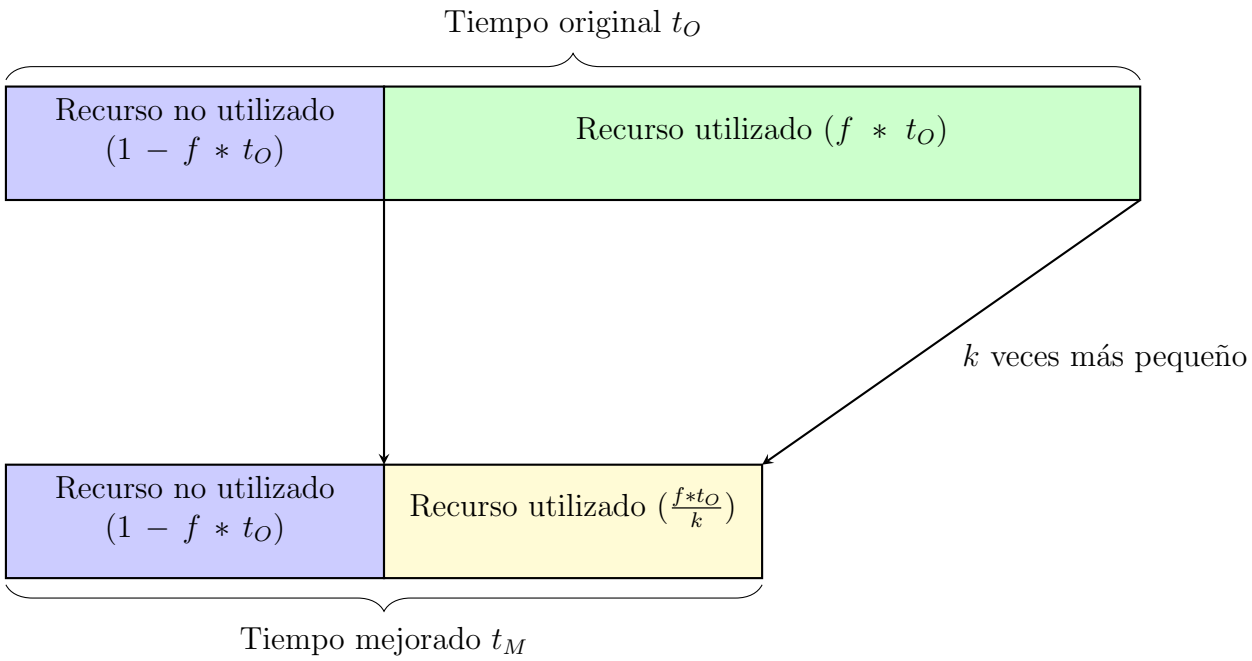

Figura 2: Tiempos original y mejorado

1.4.1. Ley de Amdahl

¿Cuál es la ganancia en velocidad (S) del sistema después de mejorar k veces un componente?

$$S = \frac{1}{1 - f + \frac{f}{k}} \tag{1}$$

Figura 3: Ley de Amdahl

Donde:

- Si $f = 0 \implies S = 1$, por lo que no hay mejora en el sistema (el recurso no se usaba).
- Si $f = 1 \implies S = k$, el sistema mejora tantas veces como el componente (se usa siempre).
- Si $f \to \infty \implies S \to \lim_{k\to\infty} S = \frac{1}{1-f}$

Veamos un ejemplo:
La utilziación de un disco duro es del 60 % para un programa monohebra. ¿Cuál será la velocidad de ejecución si duplicamos la velocidad del disco?

$$S = \frac{1}{1 - f + \frac{f}{k}} \implies S = \frac{1}{1 - 0{,}6 + \frac{0{,}6}{2}} = 1{,}43$$

Por tanto, el sistema es 1,43 veces más rápido (un 43 % más rápido que antes).
Si sólo nos centráramos en mejorar el disco, la ganancia máxima que podríamos obtener sería:

$$S_{max} = \lim_{k\to\infty} S = \frac{1}{1 - f} = \frac{1}{1 - 0{,}6} = 2{,}5$$

Es decir, el sistema podría llegar a ser 2,5 veces más rápido (un 150 % más rápido) que antes si sólo mejoramos el disco.

1.4.2. Generalización de la ley de Amdahl

Para n mejoras, el *speedup* vendría dado por:

$$S = \frac{1}{(1 - \sum_{i=1}^{n} f_i) + \sum_{i=1}^{n} \frac{f_i}{k_i}}$$

También podemos calcular la aceleración con una sola mejora en cascada, teniendo en cuenta que debemos recalcular f en cada paso.

2. Ejercicios resueltos

2.1. Tema 1

1.