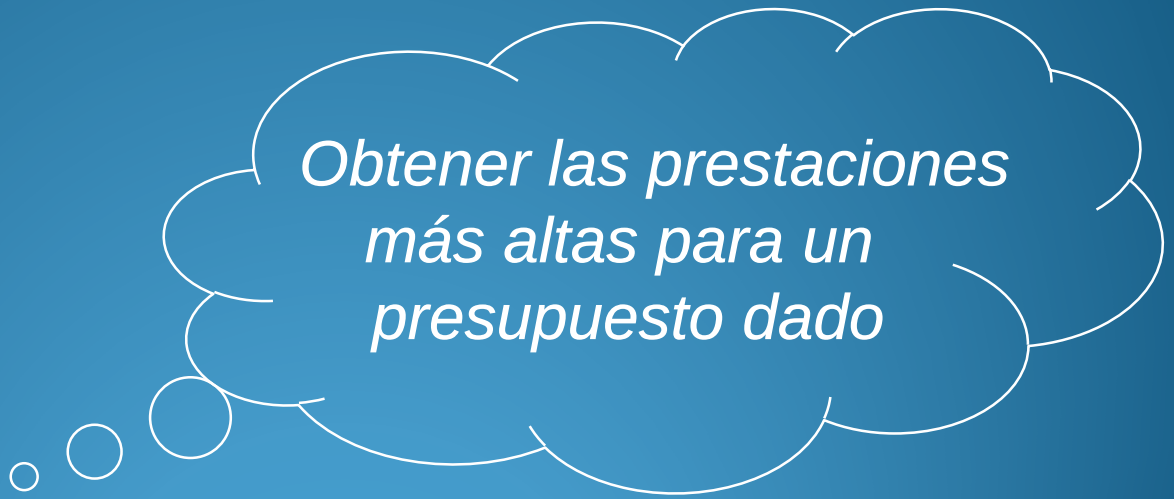
Tema 1. Introducción a la Ingeniería de Servidores



Ingeniero de Servidores

Objetivos del Tema

- Identificar el concepto de sistema informático y sus distintas clasificaciones.
- Conocer los conceptos básicos relacionados con la ingeniería de servidores.
- Ofrecer una visión general de la evaluación de las prestaciones de un servidor.
- Entender las consecuencias de la leyes de Amdahl en el proceso de mejora del tiempo de respuesta de un trabajo dentro de un servidor.

Bibliografía

- *Distributed Systems: Principles and Paradigms*. Andrew S. Tanenbaum, Maarten Van Steen. Prentice Hall, 2006.
  - Capítulo 1
- *The art of computer system performance analysis*. R. Jain. John Wiley & Sons, 1991.
  - Capítulos 1 y 3
- *Evaluación y modelado del rendimiento de los sistemas informáticos*. Xavier Molero, C. Juiz, M. Rodeño. Pearson Educación, 2004.
  - Capítulo 1
- *Measuring computer performance: a practitioner's guide*. D. J. Lilja, Cambridge University Press, 2000.
  - Capítulos 1, 2 y 7
- *Arquitectura de computadores*. Julio Ortega, Mancia Anguita, Alberto Prieto. Thomson, 2005.
  - Capítulo 1

Contenido

- Concepto de Servidor
- Fundamentos de Ingeniería de Servidores
- Comparación conjunta entre prestaciones y coste
- Límites en la mejora del tiempo de respuesta: Ley de Amdahl

1.1. ¿Qué es un servidor?

Sistema Informático

- Conjunto de elementos **hardware**, **software** y **peopleware** interrelacionados entre sí que permite obtener, procesar y almacenar información.

  - **Hardware:** conjunto de componentes *físicos* que forman el sistema informático: procesadores, memoria, almacenamiento, cables, etc.

  - **Software:** conjunto de componentes *lógicos* que forman el sistema informático: sistema operativo y aplicaciones.

  - **Peopleware:** conjunto de recursos humanos. En nuestro caso, personal técnico que instala, configura y mantiene el sistema (administradores, analistas, programadores, operarios, etc.) y los usuarios que lo utilizan.

Clasificación de Sistemas Informáticos

- Los Sistemas Informáticos pueden clasificarse según numerosos criterios. Por supuesto las clasificaciones no son estancas y es común encontrar sistemas híbridos que no encajen en una única categoría.
- Ejemplos de clasificación:
  - **Según el nivel de paralelismo de su arquitectura:**
    - SISD, SIMD, MISD o MIMD.
  - **Según su uso:**
    - De uso general o de uso específico.
  - **Según la arquitectura de servicio (para el caso de servidores):**
    - Sistema aislado, Arquitectura cliente-servidor, Arquitectura de *n* capas o Arquitectura Cliente-Cola-Cliente.

Clasificación. Paralelismo arquitectura

- Según el **paralelismo** de su arquitectura:
  - SISD: Single Instruction Single Data
  - SIMD: Single Instruction Multiple Data
  - MISD: Multiple Instruction Single Data
  - MIMD: Multiple Instruction Multiple Data

	Una instrucción	Múltiples instrucciones
Un dato	SISD	MISD
Múltiples datos	SIMD	MIMD

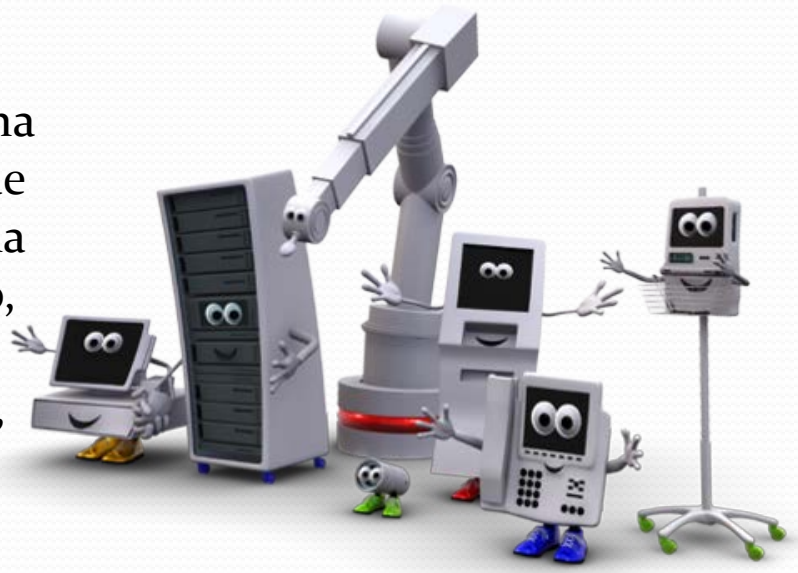
Clasificación: Según su uso

- Según su **uso**, un sistema informático puede considerarse:
  - De **uso general,** como los computadores personales (PC) que son utilizados por un usuario para ejecutar muy diversas aplicaciones.
    - PC de sobremesa (desktop)
    - PC portátil (laptop)

  - De **uso específico:**
    - Sistemas empotrados (embedded systems)
    - Servidores (servers)

Clasificación. Uso específico

- **Sistemas empotrados (embedded systems)**
  - Sistemas informáticos **acoplados** a otro dispositivo o aparato, diseñados para realizar una o algunas funciones dedicadas, frecuentemente con fuertes restricciones de tamaño, tiempo de respuesta (sistemas de tiempo real), consumo y coste.
  - Suelen estar formados por un microprocesador, memoria y una amplia gama de interfaces de comunicación.
    - Ejemplo: un taxímetro, un sistema de control de acceso, el sistema de control de una fotocopiadora, una cámara de vigilancia, un teléfono, la electrónica que controla un automóvil, un cajero automático, una lavadora, etc.



Clasificación. Uso específico: Servidores

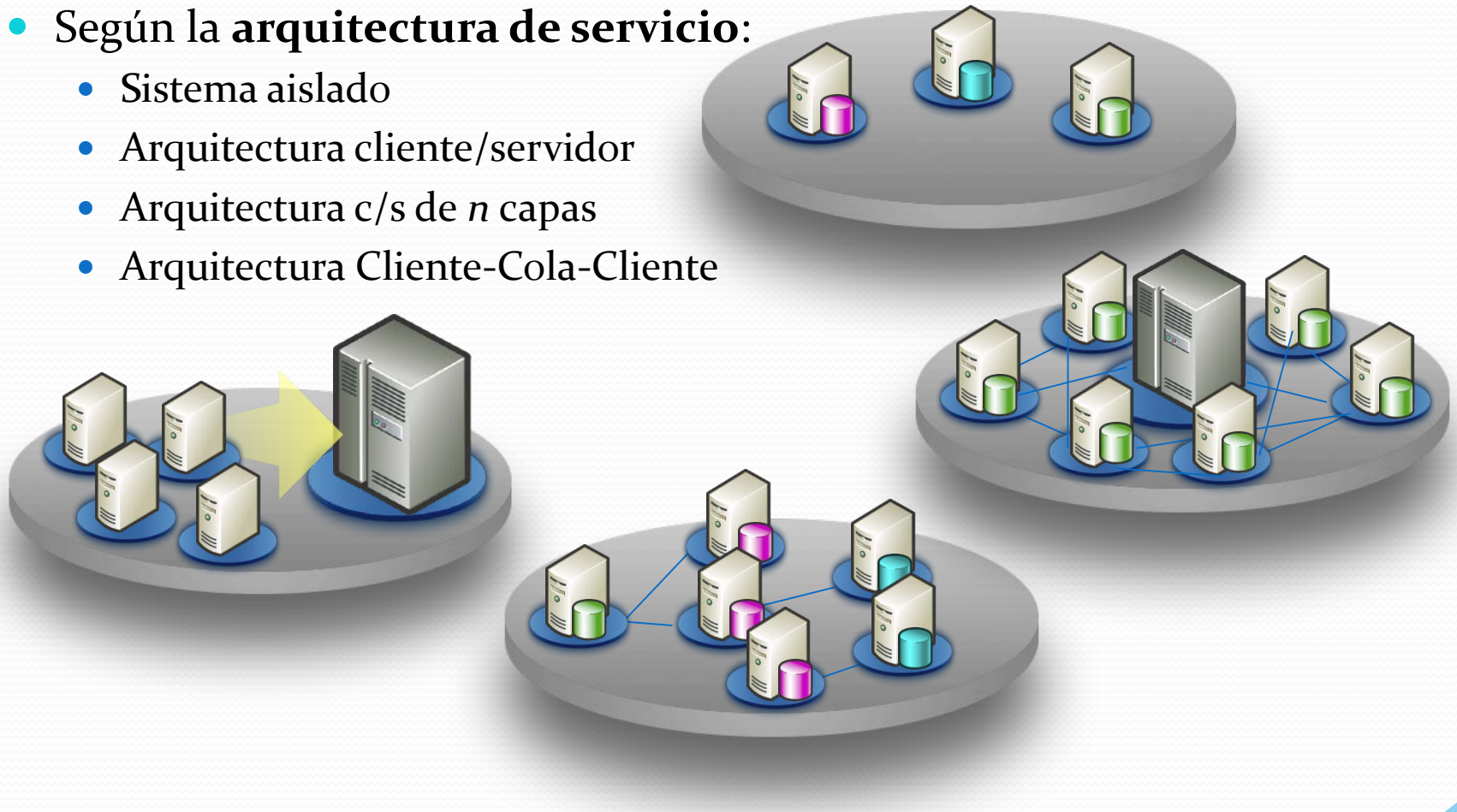
- **Servidores**
  - Son sistemas informáticos que, formando parte de una red, proporcionan servicios a otros sistemas informáticos denominados clientes.
  - Un servidor no es necesariamente una máquina de última generación de grandes proporciones; un servidor puede ser desde un computador de gama baja (coste bajo) hasta un conjunto de **clusters de computadores** (=asociación de computadores de modo que pueden ser percibidos externamente como un único sistema) en un Centro de Procesamiento de Datos (CPD).

Tipos de servidores

- **Servidor de archivos**: permite el acceso remoto a archivos almacenados en él o directamente accesibles por este.
- **Servidor web**: almacena documentos HTML, imágenes, archivos de texto, etc. y distribuye este contenido a clientes que lo soliciten en la red.
- **Servidor de base de datos**: provee servicios de base de datos a otros programas u otras computadoras.
- **Servidor de comercio-e**: cumple o procesa transacciones comerciales (comercio electrónico). Valida el 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 otros clientes de la red.
- **Servidor de correo-e**: almacena, envía, recibe, re-enruta y realiza otras operaciones relacionadas con e-mail para los clientes de la red.

Clasificación. Arquitectura de Servicio

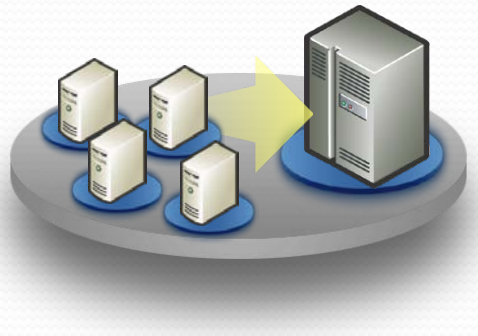
- Según la **arquitectura de servicio**:
  - Sistema aislado
  - Arquitectura cliente/servidor
  - Arquitectura c/s de *n* capas
  - Arquitectura Cliente-Cola-Cliente



Clasificación. Arquitectura de servicio

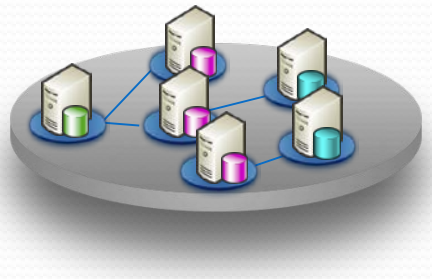
- **Sistema aislado**
  - Sistema computacional que no interactúa con otros sistemas.
  - Arquitectura monolítica en la que no existe distribución de la información.

- **Arquitectura cliente/servidor**
  - Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.
  - Suele tener dos tipos de nodos en la red
    - los clientes (remitentes de solicitudes)
    - los servidores (receptores de solicitudes)



Clasificación. Arquitectura de servicio

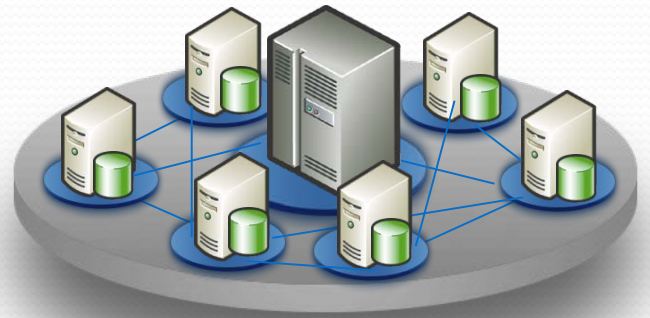
- **Arquitectura cliente/servidor de *n* capas**
  - Es una arquitectura cliente/servidor que tiene ***n*** tipos de nodos en la red.
  - Mejora la distribución de carga entre los diversos servidores. Es más escalable.
  - Pone más carga en la red.
  - Difícil de programar y administrar.



  - Ejemplo de arquitectura de 3 capas:
    - Capa 1: Servidores que interactúan con los usuarios finales (clientes).
    - Capa 2: Servidores de comercio-e que procesan los datos para los servidores de la capa 1.
    - Capa 3: Servidores de bases de datos que almacenan/buscan/gestionan los datos para los servidores de comercio-e.

Clasificación. Arquitectura de servicio

- **Arquitectura Cliente-Cola-Cliente**
  - Habilita a todos los clientes para desempeñar tareas semejantes interactuando cooperativamente para realizar una actividad distribuida, mientras que el servidor actúa como una cola que va capturando las peticiones de los clientes y sincronizando el funcionamiento del sistema.
  - La arquitectura P2P se basa en el concepto "Cliente-Cola-Cliente".
  - Aplicaciones:
    - Intercambio y búsqueda de ficheros (BitTorrent, eDonkey2000, eMule).
    - Sistemas de telefonía por Internet (Skype).



1.2. Fundamentos de Ingeniería de Servidores

Fundamentos de Ingeniería de Servidores

- **Diseño, configuración y evaluación de un Servidor**
  - Recursos físicos, lógicos y humanos
    - Placa base
    - Memoria
    - Microprocesador
    - Fuente de alimentación
    - Periféricos (E/S, Almacenamiento)
    - Sistema Operativo
    - Aplicaciones
    - Conexiones de red
    - Cableado
    - Refrigeración
    - Administración…
  - Requisitos funcionales
    - Prestaciones
    - Seguridad
    - Mantenimiento
    - Disponibilidad
    - Extensibilidad
    - Coste
    - Fiabilidad
    - Escalabilidad
    - …

Fundamentos de Ingeniería de Servidores

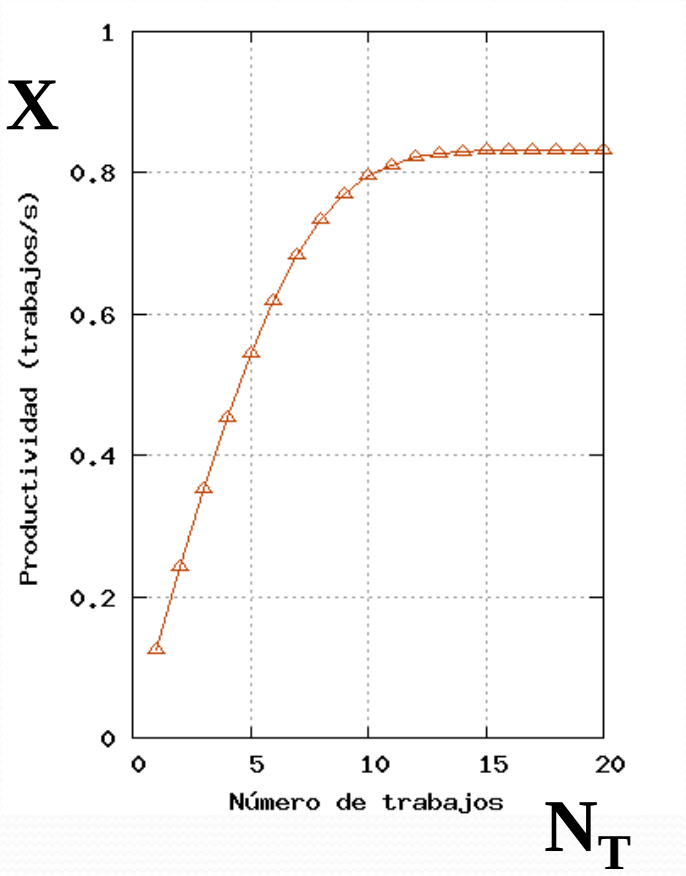
- **Prestaciones** (*Performance*)
  - Medida o cuantificación de la velocidad con que se realiza una determinada cantidad de trabajo (**carga**, *workload*/*load*).
- **Prestaciones** del servidor ↔ el **tiempo**
  - El servidor que realiza la misma cantidad de trabajo (carga) en el menor tiempo es el que *mejor* prestaciones tiene.
- Magnitudes medibles ↔ índices de prestaciones
  - Tiempo que tarda un componente o el sistema en realizar una tarea.
  - Número de trabajos realizados por algún componente o por el sistema completo por unidad de tiempo.

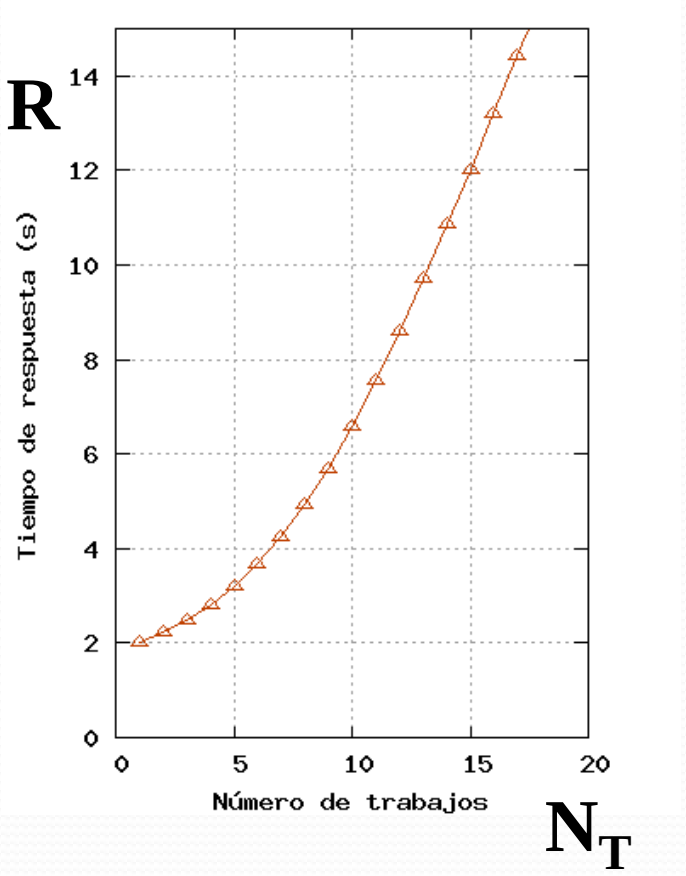
Fundamentos de Ingeniería de Servidores

- Medidas fundamentales de **prestaciones** de un servidor
  - **Tiempo de respuesta** (*response time*) o **latencia** (*latency*)
    - Tiempo total desde que se solicita una tarea al servidor o a un componente del mismo y la finalización de la misma. Por ejemplo:
      - Tiempo de ejecución de un programa.
      - Tiempo de acceso a un disco.
  - **Productividad** (*throughput*) o **ancho de banda** (*bandwidth*)
    - Cantidad de trabajo realizado por el servidor o por un componente del mismo por unidad de tiempo. Por ejemplo:
      - Programas ejecutados por hora.
      - Páginas por hora servidas por un servidor web.
      - Correos por segundo procesados por un servidor de correo.
      - Peticiones por minuto procesados por un servidor de comercio electrónico.

Fundamentos de Ingeniería de Servidores

Formas típicas de la productividad y tiempo de respuesta frente a la carga





Fundamentos de Ingeniería de Servidores

- ¿Qué afecta a las **prestaciones**? ¿Cómo podemos mejorarlas?

Componentes *hardware* del sistema:
- Características y configuración

Parámetros del sistema operativo:
- Tipos de sistema operativo.
- Políticas de planificación de procesos
- Configuración de memoria virtual, etc.

Diseño de los programas:
- Fallos de caché.
- Swapping a disco.
- Acceso a E/S, etc.

Actualización de componentes:
- Reemplazar por dispositivos más rápidos.
- Añadir nuevas unidades.

Configuración de dispositivos.

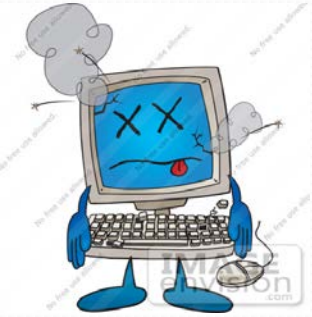
Ajuste o sintonización:
- Parámetros del sistema operativo.
- Optimización de programas.

Distribución de la carga (*load balancing*): Mayor carga a componentes más rápidos.

Una de nuestras principales misiones será analizar nuestro servidor para determinar los factores que afectan a su rendimiento y encontrar posibles soluciones para su mejora.

Fundamentos de Ingeniería de Servidores

- **Disponibilidad** (*Availability*)
  - Un servidor está disponible si se encuentra en estado operativo.
  - **Tiempo de inactividad** (*Downtime*): cantidad de tiempo en el que el sistema no está disponible.
    - Tiempo de inactividad planificado
      - Por ejemplo, actualizaciones de sw o hw que requieran rearranques.
    - Tiempo de inactividad no planificado
      - Surgen de algún evento físico tales como fallos en el hardware, anomalías ambientales o fallos software. Una alta disponibilidad implica que el sistema sea **tolerante a fallos.**

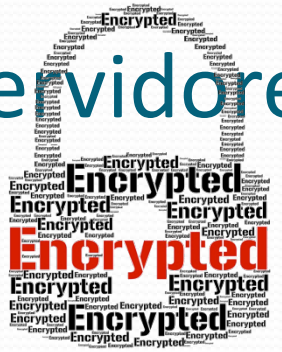


Fundamentos de Ingeniería de Servidores

- Algunas soluciones para aumentar la disponibilidad:
  - Reemplazo en caliente de componentes (*hot-swapping*).
  - Sistemas redundantes de discos (RAID 1).
  - Sistemas redundantes de alimentación.
  - Sistemas de red redundantes.
  - Sistemas distribuidos.

- **Fiabilidad** (Reliability)
  - Un sistema es fiable cuando desarrolla su actividad sin presencia de errores.
  - **MTTF** (*Mean Time To Failure*): tiempo medio que tiene un sistema (disco, memoria, etc.) hasta que ocurre un error.
  - Soluciones: uso de sumas de comprobación (checksums, bits de paridad) para detección y/o corrección de errores (memorias ECC, *Error Correcting Code*), comprobación de recepción de paquetes de red y su correspondiente retransmisión, etc.

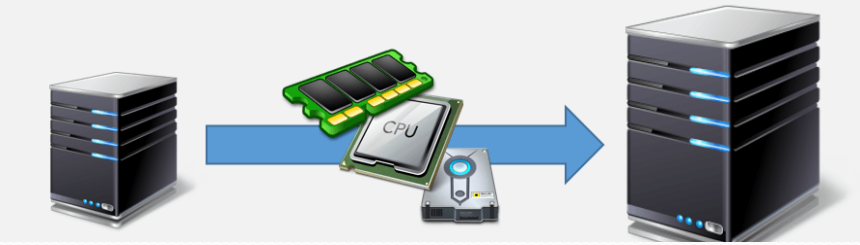
Fundamentos de Ingeniería de Servidores



- **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.
  - Soluciones:
    - Autenticación segura de usuarios.
    - Encriptación de datos.
    - Cortafuegos (firewalls).
    - Antivirus.
    - Parches de seguridad actualizados.

Fundamentos de Ingeniería de Servidores

- **Extensibilidad-expansibilidad**
  - Hace referencia a la **facilidad** que ofrece el sistema para aumentar sus características o recursos.
  - Soluciones:
    - Tener bahías libres para poder añadir más almacenamiento, memoria, etc.
    - Uso de Sistemas Operativos modulares de código abierto (para extender la capacidad del S.O.)
    - Uso de interfaces de E/S estándar (para facilitar la incorporación de más dispositivos al sistema)
    - Cualquier solución que facilite que el sistema sea *escalable*.



Fundamentos de Ingeniería de Servidores

- **Escalabilidad**
  - Un servidor es escalable si sus prestaciones pueden aumentar **significativamente** ante un incremento **significativo** en su carga.

  - Soluciones:
    - Cloud computing.
    - Virtualización.
    - Servidores modulares /clusters.
    - Arquitecturas distribuidas /arquitecturas por capas.
    - Storage Area Networks (SAN).
    - Programación paralela (software escalable).

  - Todos los sistemas escalables son extensibles pero no a la inversa.

Fundamentos de Ingeniería de Servidores

- **Mantenimiento** (*Maintenance, support*)
  - Hace referencia a todas las acciones que tienen como objetivo prolongar el funcionamiento correcto del sistema.
  - Es importante que el servidor sea fácil de mantener.
    - Sistema operativo actualizado.
    - Drivers actualizados.
    - Chequeo periódico de componentes.
    - Garantía de componentes.
    - Copias de seguridad (backup).

Fundamentos de Ingeniería de Servidores

- **Coste** (*Cost*)
  - Un diseño que sea asequible y se ajuste al presupuesto.
    - Coste hardware.
    - Coste software (S.O. + aplicaciones)
    - Coste actualizaciones hw/sw.
    - Coste personal (administrador, técnicos, apoyo…)
    - Coste proveedores de red.
    - Coste alquiler local.
    - Coste consumo eléctrico tanto del hardware como de la refrigeración: **Eficiencia energética.**

Fundamentos de Ingeniería de Servidores

- **Eficiencia Energética** (*Energy Efficiency*)
  - Uso eficiente de los recursos computacionales minimizando el coste energético, cuidando el impacto ambiental y observando deberes sociales.
  - ¿Por qué preocuparse por la eficiencia energética?
    - Reducir costes (consumo potencia servidores + refrigeración).
    - Mayor vida útil de los componentes (temperatura).
    - Preservar el medio ambiente.
  - Soluciones:
    - Ajuste automático del consumo de potencia de los componentes electrónicos según la carga.
    - Free cooling: Utilización de bajas temperaturas exteriores para refrigeración gratuita.
    - Consolidación de servidores (=combinar o fusionar varios servidores con baja utilización en un único servidor).

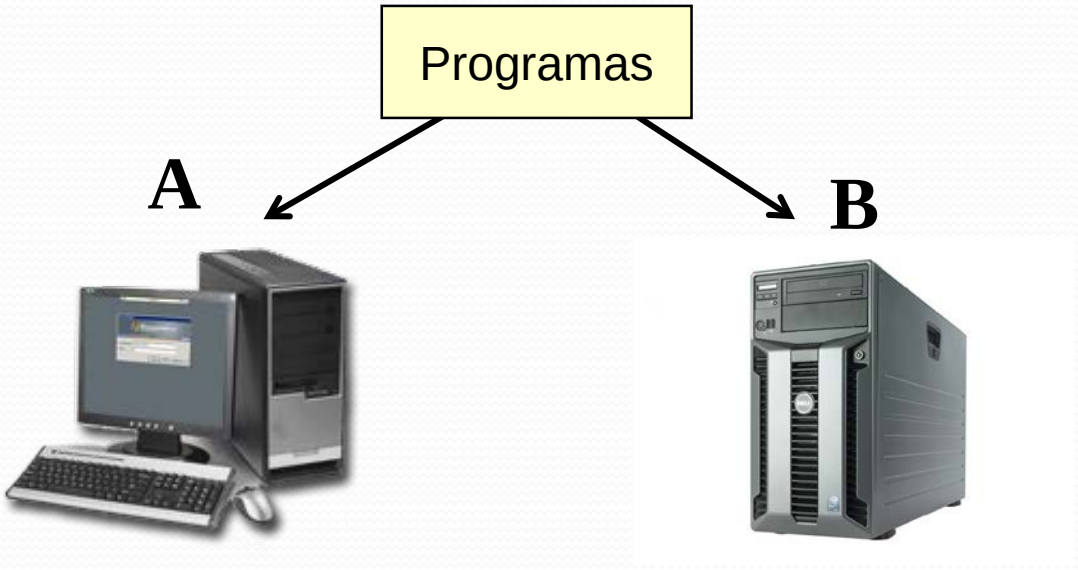
1.3. Comparación conjunta entre prestaciones y coste

Relación de prestaciones: *Speedup*

Relación prestaciones/coste

Comparación de prestaciones

- El computador de mejores prestaciones (el más rápido), para un determinado conjunto de programas, será aquel que ejecuta dicho conjunto de programas en el tiempo más corto.
  - ¿Cuántas veces es más **rápido** un computador que otro?
  - ¿Qué **tanto por ciento de mejora** conseguimos cuando reemplazamos un computador por otro más rápido?



Tiempos de ejecución mayores/menores

- Sea t_A=tiempo de ejecución de un determinado programa en la máquina A (ídem para t_B).
  - t_A es t_A/t_B veces t_B. **Ejemplo: t_A=10s, t_B=5s** → t_A es 10/5=2 veces t_B (el doble).
  - Igualmente, en el ejemplo anterior: t_B es 5/10= 0,5 veces t_A (la mitad).
- El "cambio relativo de t_A con respecto a t_B", $\Delta \boldsymbol{t_{A,B}}(\%)$, viene dado por:

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

- De donde: $\Delta t_{A,B}(\%) = \frac{t_A - t_B}{t_B} \times 100 = \left(\frac{t_A}{t_B} - 1\right) \times 100$
  - En el ejemplo anterior, el cambio relativo de t_A con respecto a t_B es 100%.
  - Igualmente, el cambio relativo de t_B con respecto a t_A sería -50%.
- Uso del "lenguaje común" en el ejemplo anterior:
  - "t_A es un 100% **mayor que** t_B". "t_B es un 50% **menor que** t_A".
  - "t_A es 2 veces ***mayor que*** t_B"(¡ojo!, "1 vez mayor" quiere decir "iguales").

¿Qué máquina es más rápida? Speedup

- Sea t_A=tiempo de ejecución de un determinado programa en la máquina A (ídem para t_B).
- La "velocidad" de la máquina A para ejecutar dicho programa será inversamente proporcional a t_A: $v_A=D/t_A$ (siendo D la "distancia recorrida por la máquina" = cómputo realizado). Igualmente $v_B=D/t_B$, donde hemos utilizado la misma distancia "D" ya que han realizado la misma cantidad de cómputo (el mismo programa).
- Para ese programa, se define la ganancia en velocidad (speedup o aceleración) de la máquina A con respecto a la máquina B como:

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

- El cambio relativo de v_A con respecto a v_B (% de cambio en velocidad al reemplazar la máquina A por la B) viene dado por:

$$\Delta v_{A,B}(\%) = \frac{v_A - v_B}{v_B} \times 100 = \left(\frac{v_A}{v_B} - 1\right) \times 100 = (S_B(A) - 1) \times 100$$

¿Qué máquina es más rápida? Ejemplo

- Supongamos que, para un determinado programa, **t_A=36s** y **t_B=45s**.
- En ese caso, la ganancia en velocidad (=speedup) de la máquina A con respecto a la máquina B sería:

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

- El cambio relativo de v_A con respecto a v_B (% de mejora) viene dado por:

$$\Delta v_{A,B}(\%) = \frac{v_A - v_B}{v_B} \times 100 = (S_B(A) - 1) \times 100 = 25\%$$

- Usando el "lenguaje común" diremos que, para ese programa:
  - La máquina A es 1,25 veces "**más rápida que**" la B (¡ojo!, "1 vez más rápido" o "ganancia en velocidad = 1" quieren decir "misma velocidad")
  - La máquina A es un 25% **más rápida que** la B.
- Igualmente:
  - La máquina B es ($S_A(B)$-1=$^{36}/_{45}$-1 = -0,2) un 20% **más lenta que** la A.

Relación prestaciones/coste

- Supongamos que, siguiendo el ejemplo anterior (t_A=36s y t_B=45s):
  - El computador **A cuesta 625 €**.
  - El computador **B cuesta 550 €**.
- El computador A es 625/550 = 1,14 veces "**más caro que**" el B (un 14% más caro)
- ¿Cuál ofrece mejor relación prestaciones/coste para nuestro programa?

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

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

- El computador A presenta una mejor relación prestaciones/coste que el B (1,1 veces *"mayor"* = un 10% mayor) para nuestro programa.

$$\frac{Prestaciones_A/Coste_A}{Prestaciones_B/Coste_B} = \frac{4{,}4\times10^{-5}}{4{,}0\times10^{-5}} = 1{,}1$$

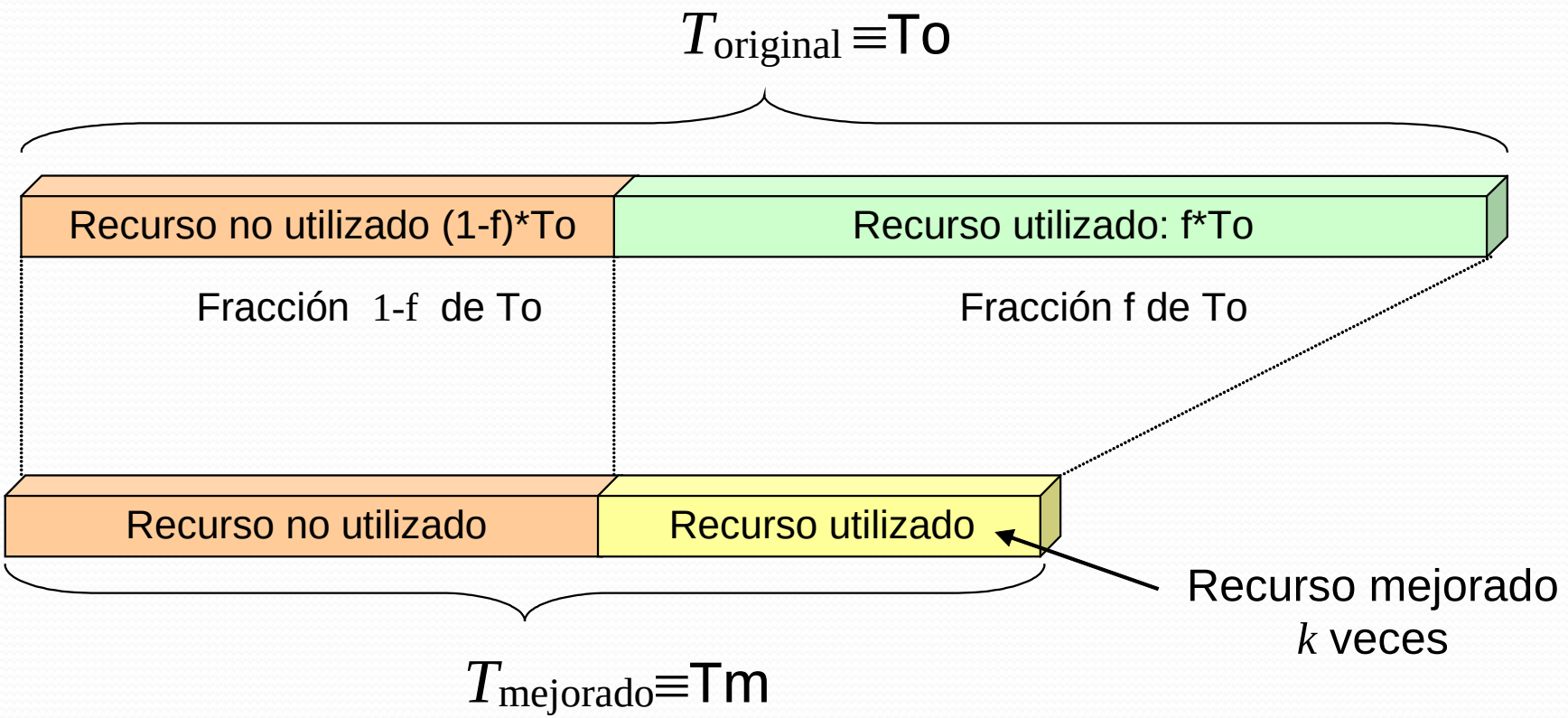
1.4. Límites en la mejora del tiempo de respuesta

La ley de Amdahl

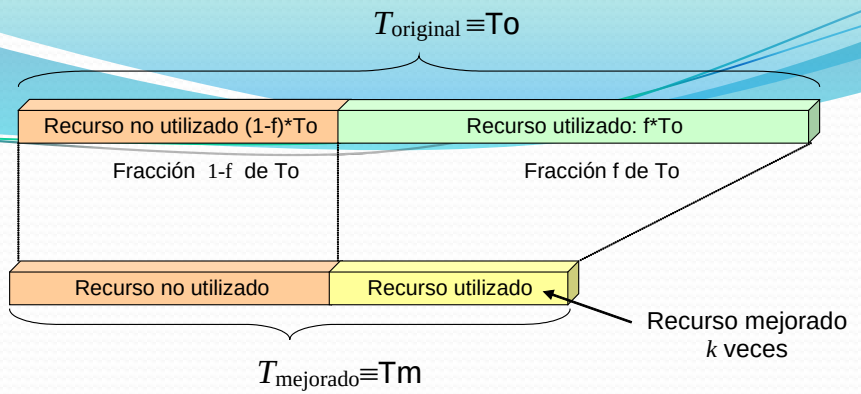
Mejora del Tiempo de Respuesta

- La mejora del tiempo de respuesta (en nuestro caso, tiempo de ejecución de un proceso) no es ilimitada.
  - Hay que saber hacia dónde dirigir los esfuerzos de optimización.
- La mejora de cualquier sistema usando un componente más rápido depende de la fracción de tiempo que éste se utilice.
- Planteamiento:
  - Un sistema tarda un tiempo $T_{original}$ en ejecutar un programa monohebra (=no hay acceso simultáneo a dos o más recursos del sistema).
  - Mejoramos el sistema reemplazando uno de sus componentes por otro *k* veces "más rápido".
  - Este componente se utilizaba durante una fracción f del tiempo $T_{original}$ (f=1 significa que se usaba el 100% del tiempo).
  - ¿Cuál es la ganancia en prestaciones (*speedup*) para la ejecución de ese programa que conseguimos con el cambio?

Tiempo original vs tiempo mejorado



Ley de Amdahl



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

$$T_m = (1-f)\cdot T_o + \frac{f\cdot T_o}{k}$$

$$S \equiv S_{original}(mejorado) = \frac{v_m}{v_o} = \frac{T_o}{T_m} = \frac{T_o}{(1-f)\cdot T_o + \frac{f\cdot T_o}{k}} = \frac{1}{1-f+{}^{f}/_{k}}$$

$$S = \frac{1}{1-f+{}^{f}/_{k}}$$ **Ley de Amdahl**

- Casos particulares de la ley
  - Si $f = 0 \Rightarrow S = 1$: no hay ninguna mejora en el sistema.
  - Si $f = 1 \Rightarrow S = k$: el sistema mejora tantas veces como el componente.
  - Si $k \to \infty$: $S \to \lim_{k\to\infty} S = \frac{1}{1-f}$

Ejemplo de cálculo

- La utilización de un disco duro es del 60% para un programa monohebra dado ejecutado sobre un determinado sistema informático.
- ¿Cuál será la ganancia en velocidad del sistema si se duplica la velocidad del disco?

$$S = \frac{1}{1 - 0{,}6 + {}^{0{,}6}/_{2}} = 1{,}43$$

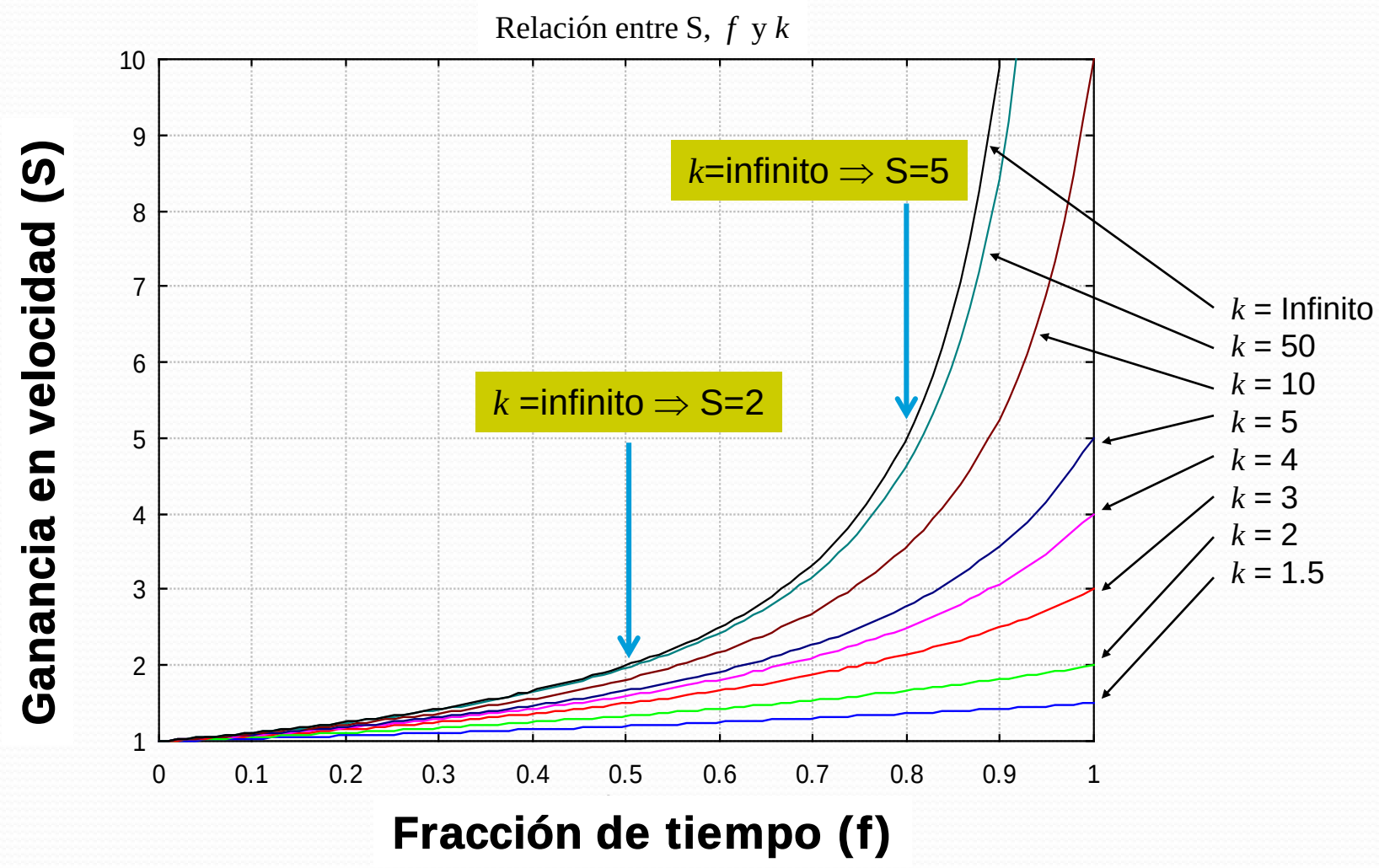
El sistema es ahora 1,43 veces *"más rápido"* (un 43% más rápido) que antes.

- Ganancia máxima que se podría conseguir actuando solo sobre el disco:

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

El sistema como mucho podría llegar a ser 2,5 veces *"más rápido"* (un 150% más rápido) que antes actuando solo sobre el disco.

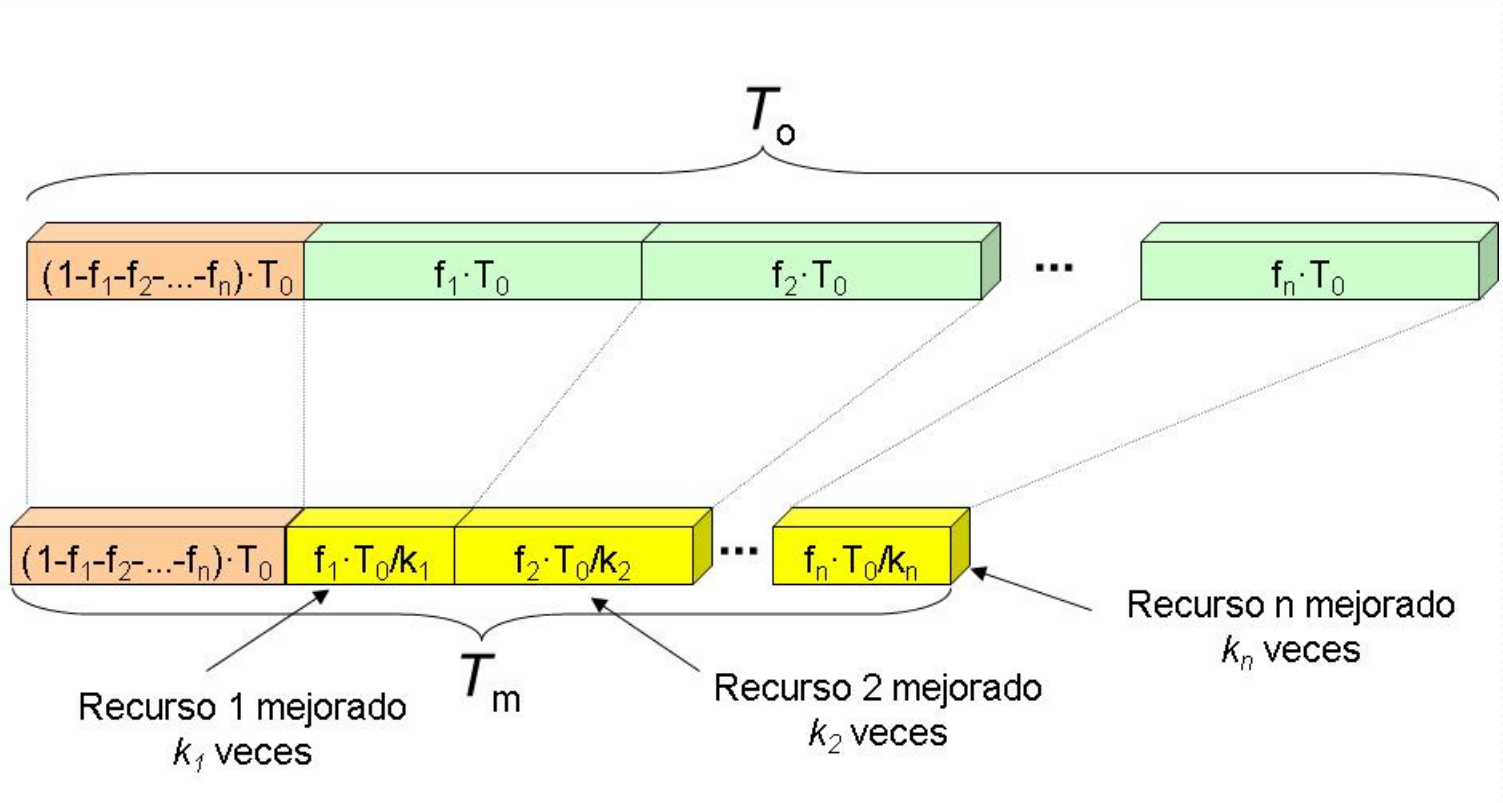
Análisis: Relación entre S, f y k



Generalización de la ley de Amdahl

- Caso general con n mejoras:

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



Algunas reflexiones finales

- Una mejora es más efectiva cuanto más grande es la fracción de tiempo en que ésta se aplica. Es decir, hay que optimizar los elementos que se utilicen durante la mayor parte del tiempo (caso más común)
- Con la ley de Amdahl podemos estimar la ganancia en velocidad (*speedup*) de la ejecución de **un único trabajo (un hilo)** en un sistema después de mejorar ***k*** veces un componente, es decir, su tiempo de respuesta óptimo en ausencia de otros trabajos.

¿Qué ocurre cuando tenemos varios trabajos ejecutándose simultáneamente en el servidor?
¿Qué ocurre si en lugar de mejorar un componente lo que hago es añadir uno nuevo?