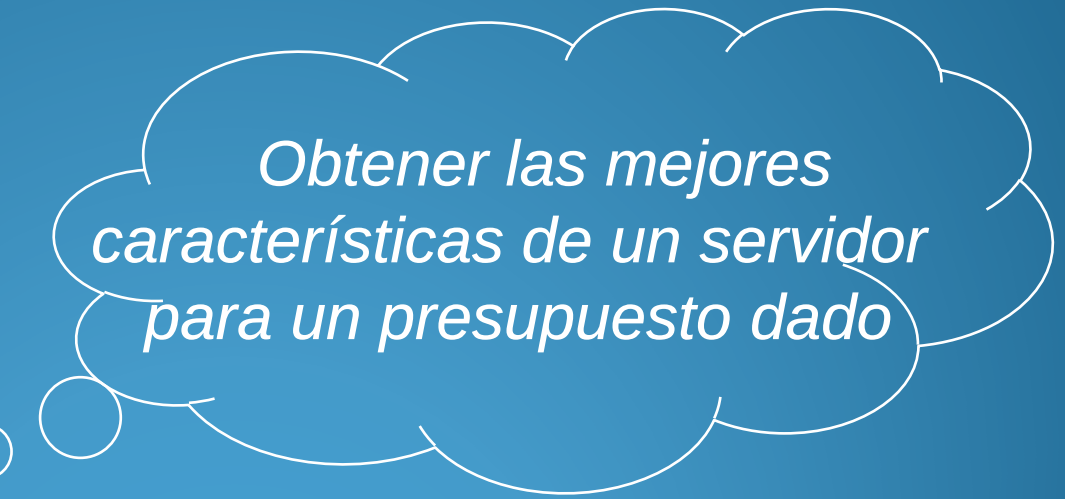
Tema 1. Introducción a la Ingeniería de Servidores



Ingeniero de Servidores

Objetivos del tema

- Identificar el concepto de servidor y sus distintas características.
- Conocer los conceptos básicos relacionados con la ingeniería de servidores.
- Ofrecer una visión general de cómo comparar prestaciones y otras magnitudes similares entre servidores.
- Entender las consecuencias de la ley de Amdahl en el proceso de mejora del tiempo de respuesta de una solicitud a 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.
- Introducción a la comparación de características entre servidores.
- Introducción a la mejora del tiempo de respuesta de un servidor: Ley de Amdahl.

1.1. ¿Qué es un servidor?

Sistema Informático (S.I.)

- 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.

- Por ejemplo, podríamos clasificarlos según el **paralelismo** de su arquitectura de procesamiento en:

  - SISD: Single Instruction Single Data
  - SIMD: Single Instruction Multiple Data
  - MISD: Multiple Instruction Single Data
  - MIMD: Multiple Instruction Multiple Data

	Una instrucción	Muchas instrucciones
Un dato	SISD	MISD
Muchos datos	SIMD	MIMD

Clasificación de S.I. según su uso

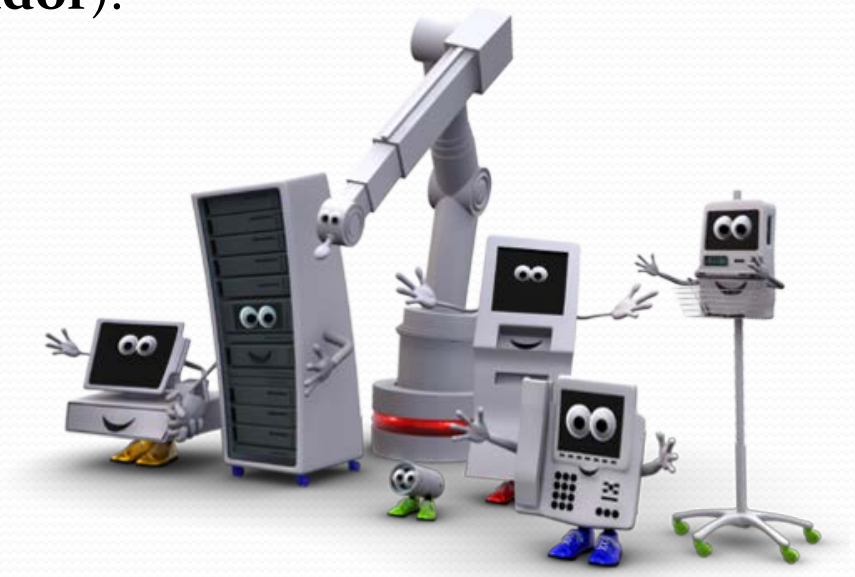
- Según su **uso**, un sistema informático puede considerarse:
  - De **uso general,** como los computadores personales (PC) que son utilizados por un usuario para ejecutar muy diversas aplicaciones.
    - PC de sobremesa (desktop)
    - PC portátil (laptop)

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

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 (= **microcontrolador**).

- Ejemplos: 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.



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).

Ejemplos de servidores según la función que realizan

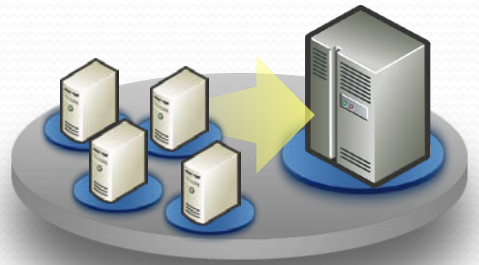
- **Servidor web**: procesa solicitudes realizadas a través de HTTP o HTTPS (*Hypertext Transfer Protocol Secure*).
- **Servidor de archivos**: permite el acceso remoto a archivos almacenados en él o directamente accesibles por este.
- **Servidor de base de datos**: provee servicios de base de datos a otros programas u otras computadoras.
- **Servidor de comercio-e**: facilita y procesa transacciones comerciales. P.ej. valida al cliente, procesa un pago, genera consultas al servidor de bases de datos…
- **Servidor de correo-e**: almacena, envía, recibe, re-enruta y realiza otras operaciones relacionadas con e-mail para los clientes de la red.
- **Servidor DHCP** (*Dynamic Host Configuration Protocol*): asigna dinámicamente una dirección IP y otros parámetros de configuración de red a cada dispositivo que entra en una red.
- **Servidor DNS** *(Domain Name System)*: devuelve la dirección IP asociada a un nombre de dominio.
- **Servidor de impresión**: controla una o más impresoras y acepta trabajos de impresión de otros clientes de la red.

Posibles arquitecturas de servicio

Según la **arquitectura de servicio**, es decir, cómo se establece la interacción entre el servidor y sus clientes, podemos distinguir entre:

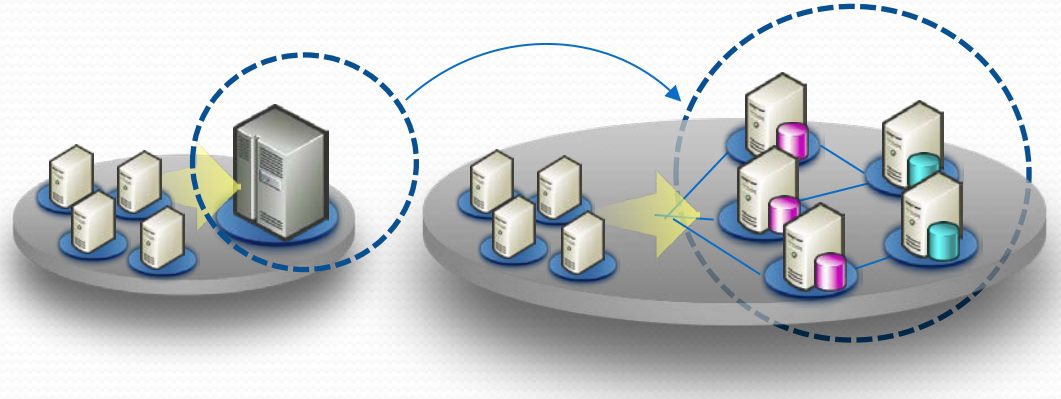
- Arquitectura cliente/servidor.
- Arquitectura cliente-cola-cliente.

- **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/niveles en la red:
    - los clientes (remitentes de solicitudes).
    - los servidores (receptores de solicitudes).



Posibles arquitecturas de servicio (continuación)

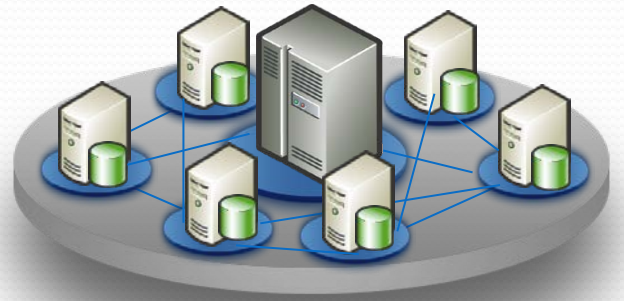
- **Arquitectura cliente/servidor de varios niveles**
  - El servidor se sub-divide en varios niveles de (micro-)servidores más sencillos con funcionalidades diferentes en cada nivel.
  - Permite la distribución de carga entre los diversos servidores. Es más escalable.
  - Pone más carga en la red.
  - Más difícil de programar y administrar.



- Ejemplo de arquitectura de 3 niveles:
  - Nivel 1: Clientes.
  - Nivel 2: Servidores de comercio-e que interactúan con los clientes.
  - Nivel 3: Servidores de bases de datos que almacenan/buscan/gestionan los datos para los servidores de comercio-e.

Posibles arquitecturas de servicio (continuación)

- **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.
  - 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

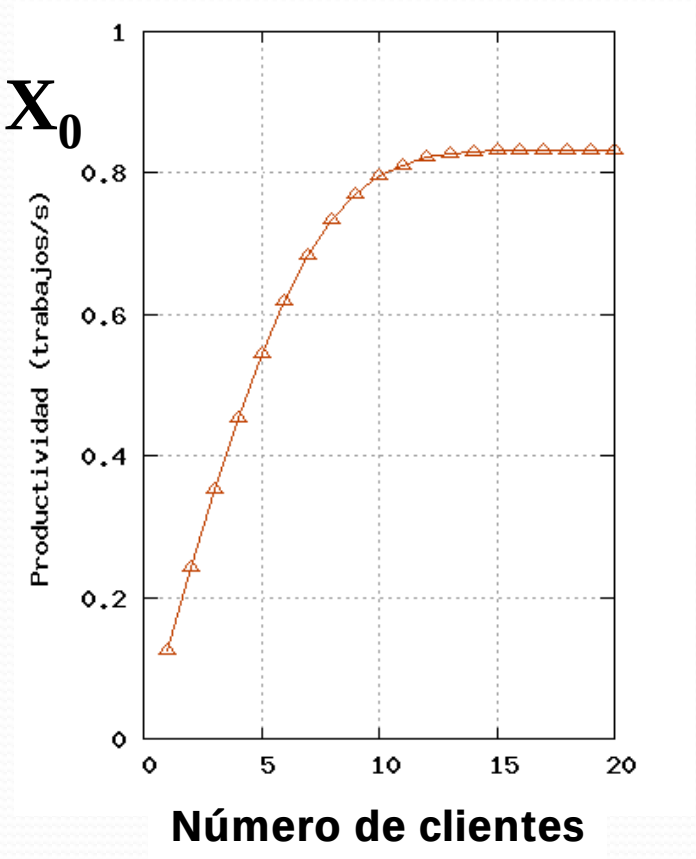
- **Diseño, configuración y evaluación de un Servidor**
  - Recursos físicos, lógicos y humanos:
    - Placa base
    - Memoria
    - Microprocesador
    - Fuente de alimentación
    - Periféricos (E/S, almacenamiento)
    - Sistema Operativo
    - Aplicaciones
    - Conexiones de red
    - Cableado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

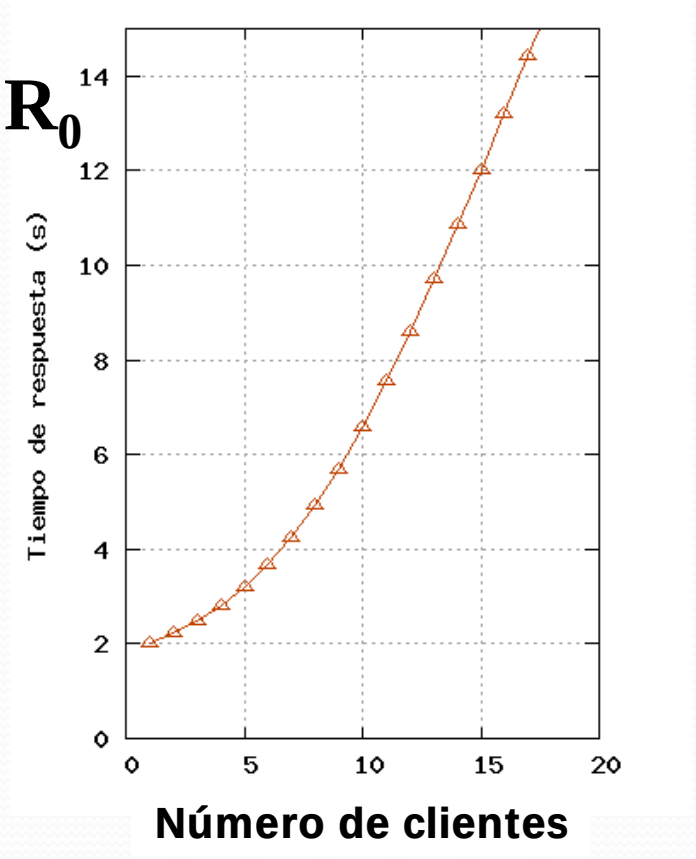
Prestaciones o rendimiento (*Performance*)

- Medida o cuantificación de la velocidad con que se realiza una determinada **carga** o cantidad de trabajo (*workload/load*).
- Medidas fundamentales de **prestaciones** de un servidor o de sus componentes:
  - **Tiempo de respuesta** (*response time*) o **latencia** (*latency*): Tiempo transcurrido desde que se solicita una tarea al servidor o a un componente y la finalización de dicha tarea. Por ejemplo:
    - Tiempo que tarda el servidor web en responder a una petición de una página concreta.
    - Tiempo de ejecución de un programa.
    - Tiempo de lectura de un determinado fichero de un disco.
  - **Productividad** (*throughput*) o **ancho de banda** (*bandwidth*): Cantidad de trabajo por unidad de tiempo (segundo, minuto, hora…) realizado por el servidor o por un componente. Por ejemplo:
    - Bytes por segundo transmitido a través de una interfaz de red.
    - Programas ejecutados por minuto.
    - Páginas por hora servidas por un servidor web.
    - Correos por segundo procesados por un servidor de correo electrónico.

Prestaciones: productividad y tiempo de respuesta

- Formas típicas de la productividad y tiempo de respuesta de un servidor frente a la carga:





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

Componentes hardware del sistema:
- Características y configuración (cuello de botella).

Sistema operativo:
- Tipo de sistema operativo.
- Políticas de planificación de procesos.
- Configuración de memoria virtual, etc.

Aplicaciones:
- Hot spots.
- Acceso a E/S, *swapping*, etc.
- Fallos de caché, de página, etc.

Actualización de componentes:
- Reemplazar por dispositivos más rápidos.
- Añadir nuevos componentes: distribución de carga (*load balancing*). Mayor carga a componentes más rápidos.

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

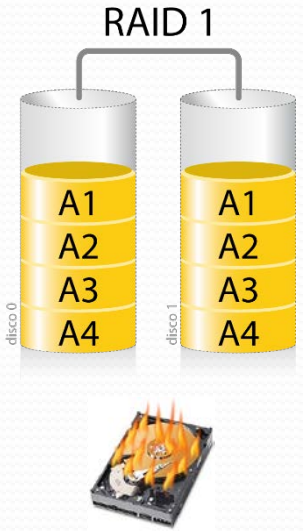
- Una de nuestras principales misiones será analizar nuestro servidor para determinar los factores que afectan a sus prestaciones y encontrar posibles soluciones para su mejora.

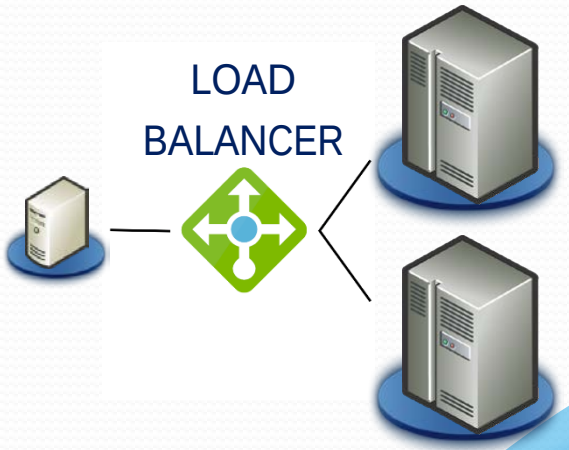
Fiabilidad (*Reliability*)

- Es la probabilidad de que el servidor (o el componente/sistema de que se trate) funcione **correctamente** durante un determinado intervalo de tiempo.
- Un servidor/sistema/componente 100% fiable debe ser tolerante a todo tipo de fallos como, por ejemplo:
  - Fallos físicos permanentes: Un componente se estropea y hay que reemplazarlo.
  - Fallos físicos temporales: Un componente da una respuesta incorrecta puntual (p. ej. un bit cambia de valor) debido a algún efecto transitorio (ruido electromagnético, fluctuaciones de la fuente de alimentación, alta temperatura), pero el componente sigue estando operativo.
  - Fallos software: El servidor da una respuesta equivocada debido a algún fallo de programación.
- Algunas métricas que reflejan la fiabilidad de un componente:
  - MTTF: *Mean Time To Failure*.
  - AFR: *Annualized failure rate*.
  - FIT: *Failures in Time*.
  - MTBF: *Mean Time Between Failures*.

¿Cómo mejorar la fiabilidad de un servidor?

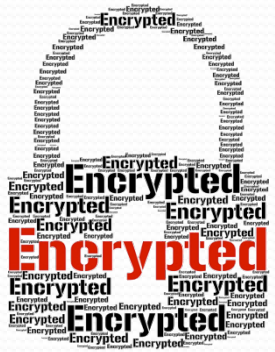
- Uso de sumas de comprobación (checksums, bits de paridad) para detección y/o corrección de errores (memorias ECC, *Error Correcting Code*), comprobación de recepción de paquetes de red y su correspondiente retransmisión, etc.
- Sistemas de alimentación ininterrumpida (SAI, *Uninterruptible power supply - UPS*).
- Sistemas **redundantes** de discos (*RAID 1*), de alimentación, de red o incluso de servidores completos (distribución de carga, *load balancing*) en los que se pueda detectar el fallo y reaccionar ante él para poder seguir devolviendo respuestas correctas a las peticiones que se hagan.





Seguridad

- Hace referencia a todas las acciones que tienen como objetivo evitar:
  - 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.

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. Para ello, puede ser conveniente usar:
  - S.O. con actualizaciones automáticas (parches de seguridad, actualización de drivers, etc.)
  - *Cloud computing*: El proveedor se encarga del chequeo periódico de componentes y su actualización.
  - Automatización de copias de seguridad (respaldo o *backup*).
  - Automatización de tareas de configuración y administración (*Ansible, Chef, Puppet*).

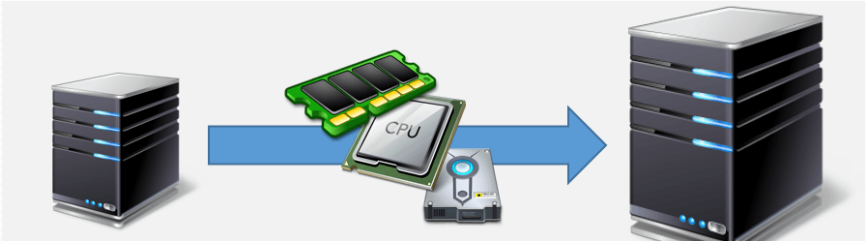
Disponibilidad (*availability*)

- Es el porcentaje de tiempo en que el servidor (o el componente de que se trate) está operativo (=respondiendo a las peticiones que se le hagan), independientemente de que las respuestas a dichas peticiones sean correctas o no.
- **Tiempo de inactividad** (*downtime*): cantidad de tiempo en el que el servidor (o el componente de que se trate) no está disponible.
  - Tiempo de inactividad planificado
    - Por ejemplo, labores de mantenimiento que requieran apagados o re-arranques (actualización de S.O., cambio de componentes, etc.)
  - Tiempo de inactividad no planificado
    - Surgen de algún evento no esperado tales como fallos hw/sw, ataques, etc.

- ¿Cómo mejorar la disponibilidad de un servidor?
  - S.O. modulares que permitan actualizaciones sin re-iniciar el sistema.
  - Inserción de componentes en caliente (*hot-plugging*).
  - Reemplazo en caliente de componentes (*hot-swapping*).
  - Cualquier solución que haga al servidor **más fiable** frente a fallos físicos permanentes de algún componente o **más seguro** ante ataques que interfieran en el acceso a recursos.

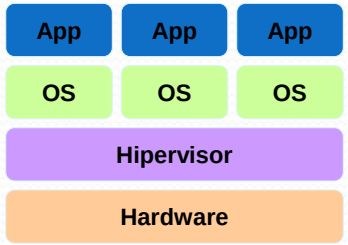
Extensibilidad-expansibilidad

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



Escalabilidad

- Hace referencia a la facilidad que ofrece el sistema para poder **aumentar** de forma **significativa** sus características o recursos para enfrentarnos a un aumento **significativo** de la carga.
- Soluciones:
  - *Cloud computing* + virtualización.
  - Servidores modulares /clusters.
  - Arquitecturas distribuidas /arquitecturas por capas.
  - Programación paralela (software escalable).



Virtualización

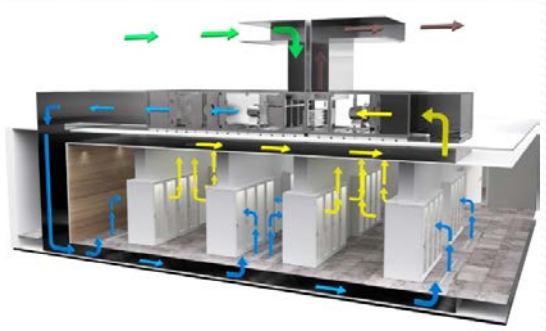
Escalabilidad vertical:

Escalabilidad horizontal:

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

Coste

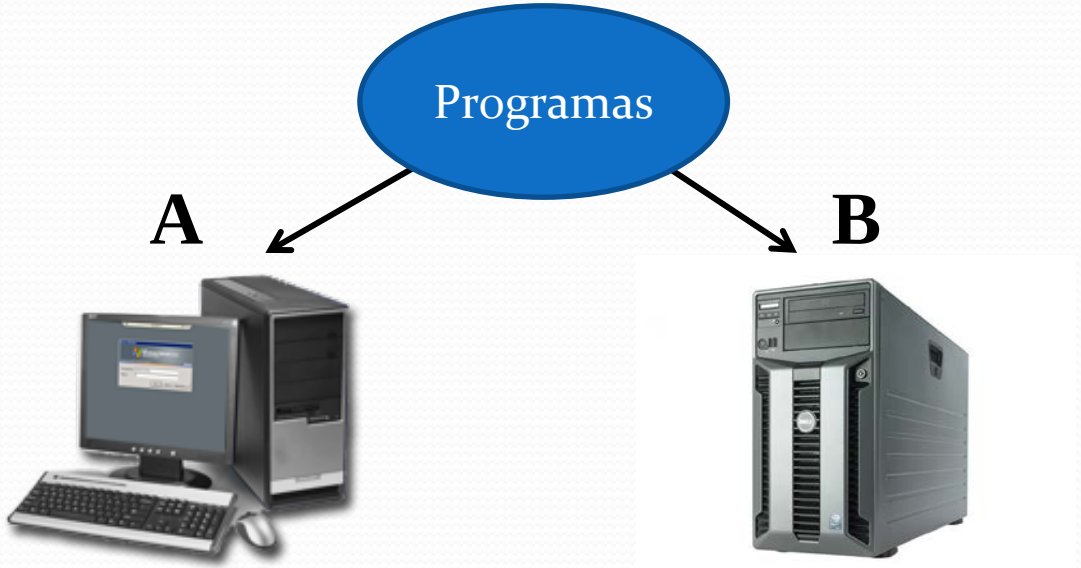
- Tenemos que adaptarnos al presupuesto. No solo se ha de tener en cuenta el coste *hw* y *sw* sino también el coste de: mantenimiento, personal (administrador, técnicos, apoyo…), proveedores de red, alquiler del local donde se ubica el servidor, consumo eléctrico (tanto del servidor como de la refrigeración).
- Podría contribuir a abaratar el coste:
  - *Cloud computing*.
  - Usar *sw* de código abierto (open source).
  - Reducir costes de electricidad (eficiencia energética):
    - Ajuste automático del consumo de potencia de los componentes electrónicos según la carga.
    - Free cooling: Utilización de bajas temperaturas exteriores para refrigeración gratuita.



1.3. Introducción a la comparación de características entre servidores

Motivación: 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.
- Cuando se comparan las prestaciones, en lugar de mostrar los tiempos de ejecución obtenidos se suele indicar:
  - ¿Cuántas **veces es más rápido** un computador que otro?
  - ¿Qué **tanto por ciento de mejora en velocidad** aporta el equipo más rápido con respecto al más lento?



Tiempos de ejecución mayores/menores

- Sea t_A=tiempo de ejecución de un determinado programa en la máquina A (ídem para t_B).
  - **Ejemplo: t_A=10s, t_B=5s** → t_A es 10/5=2 veces t_B (el doble). En general, t_A es t_A/t_B veces t_B.
  - Igualmente, en el ejemplo anterior: t_B es t_B/t_A = 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:

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

- 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 $\left(\frac{10}{5} - 1\right) \times 100 = 100\%$
  - Igualmente, el cambio relativo de t_B con respecto a t_A es $\left(\frac{5}{10} - 1\right) \times 100 = -50\%$
- 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" 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:

$$\Delta v_{A,B}(\%) = \frac{v_A - v_B}{v_B} \times 100 = (S_B(A) - 1) \times 100 = 0{,}25 \times 100 = 25\%$$

- 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 un 20% **más lenta que** la A.

$$\Delta v_{B,A}(\%) = (S_A(B) - 1) \times 100 = \left(\frac{36}{45} - 1\right) \times 100 = (0{,}8 - 1) \times 100 = -20\%$$

Coste y 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. En este caso, "*mayor*" es "*mejor*".

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

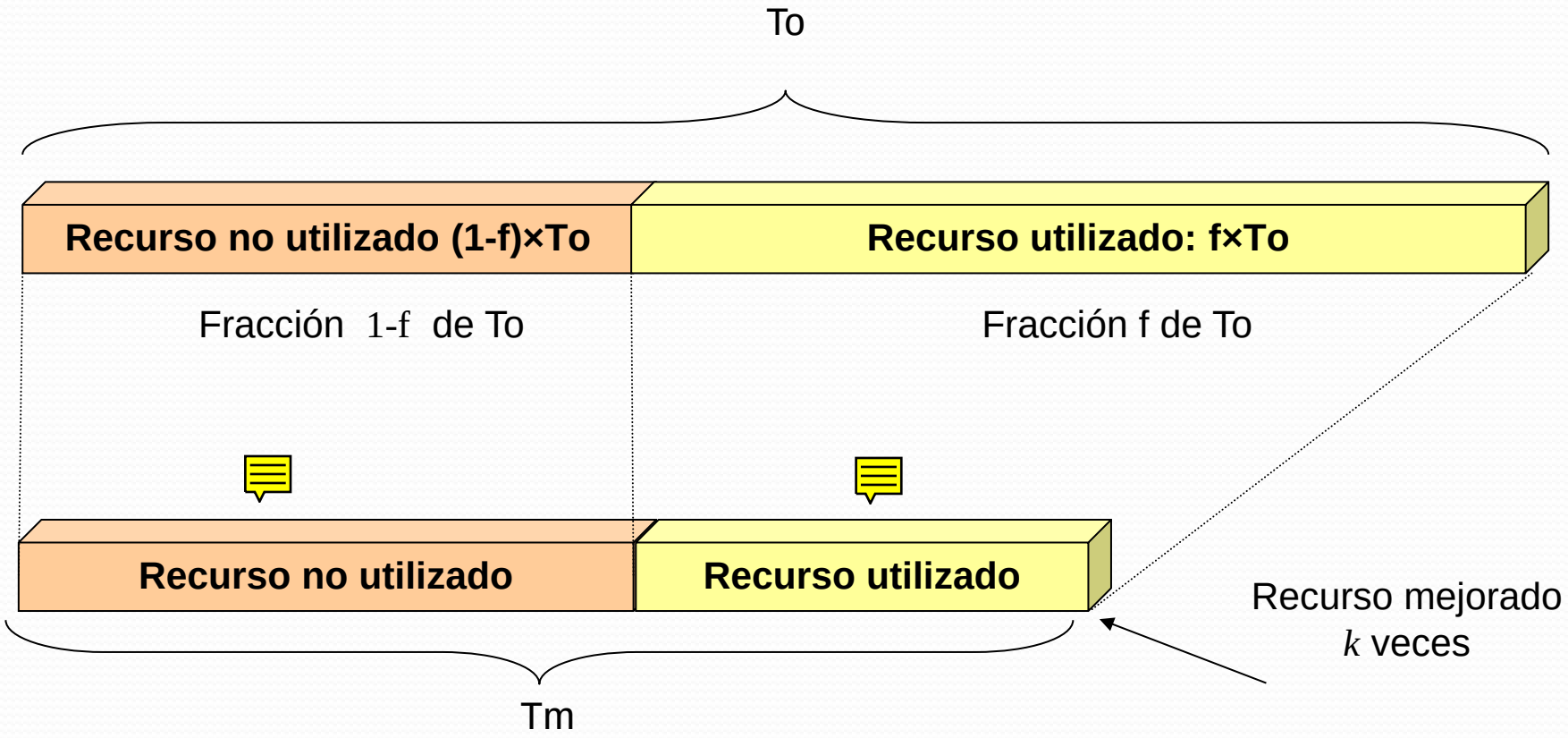
1.4. Introducción a la mejora del tiempo de respuesta de un servidor

Ley de Amdahl

Mejora del tiempo de ejecución de un proceso

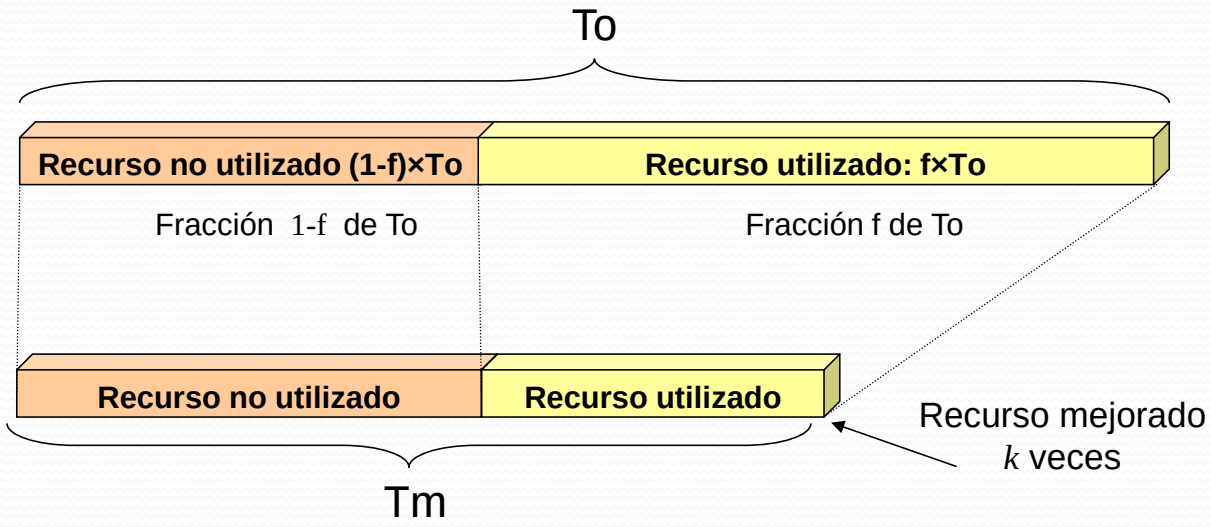
- Una de las formas más habituales de mejorar el tiempo de respuesta de un servidor consiste en reemplazar un componente del mismo por otro más rápido.
- El caso más sencillo para evaluar la mejora conseguida consiste en suponer que en el servidor sólo se ejecuta un único proceso monohebra (=no hay acceso simultáneo a dos o más recursos del sistema).
- En ese caso, supongamos que:
  - El servidor tarda un tiempo $T_{original} \equiv To$ en ejecutar dicho proceso.
  - Mejoramos el servidor reemplazando uno de sus componentes por otro k veces "más rápido" (suponiendo k>1), es decir, un (k-1)×100% "más rápido".
  - Este componente se utilizaba durante una fracción f del tiempo To (f=1 significaría que se usaba el 100% del tiempo).
- ¿Cuál es la ganancia en velocidad (*speedup*) en la ejecución de ese proceso que conseguimos con el cambio?

Tiempo original (To) vs tiempo mejorado (Tm)



Ley de Amdahl

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



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

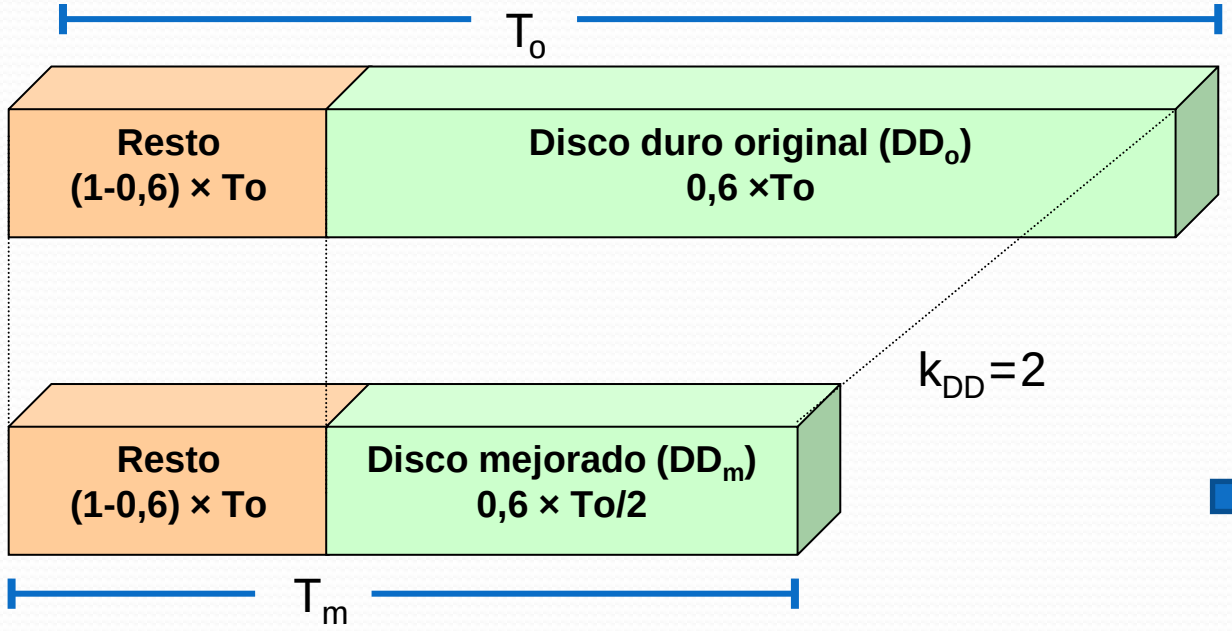
$$S \equiv S_{original}(mejorado) = \frac{v_m}{v_o} = \frac{T_o}{T_m} = \frac{T_o}{(1 - f) \times T_o + \frac{f \times T_o}{k}} = \frac{1}{1 - f + {}^{f}/_{k}}$$

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

- Supongamos que la utilización de un disco duro durante la ejecución de un programa monohebra en un determinado servidor es del 60%. ¿Cuál sería la ganancia en velocidad del servidor al ejecutar dicho programa si se duplicara la velocidad del disco?



$$S = \frac{v_m}{v_o} = \frac{T_o}{T_m} = \frac{T_o}{(1-0{,}6) \times T_o + \frac{0{,}6 \times T_o}{2}} =$$

$$\frac{1}{1 - 0{,}6 + \frac{0{,}6}{2}} = 1{,}43$$

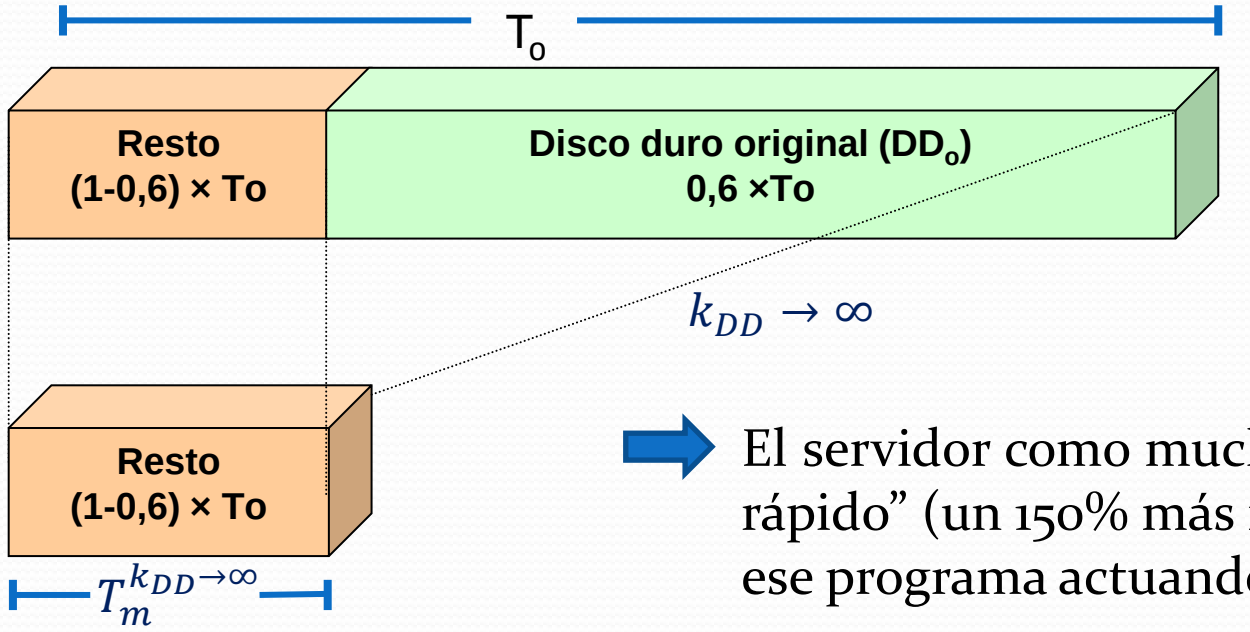
El servidor es ahora 1,43 veces *"más rápido"* (un 43% más rápido) que antes.

Otra forma: Usamos la Ley de Amdahl:

$$S = \frac{1}{1 - f_{DD_O}^{T_o} + {f_{DD_O}^{T_o}}/{k_{DD}}} = \frac{1}{1 - 0{,}6 + {0{,}6}/{2}} = 1{,}43$$

Ejemplo de aplicación de la Ley de Amdahl (cont.)

- Para el mismo ejercicio anterior, ¿cuál es la ganancia máxima que se podría conseguir actuando sólo sobre el disco?



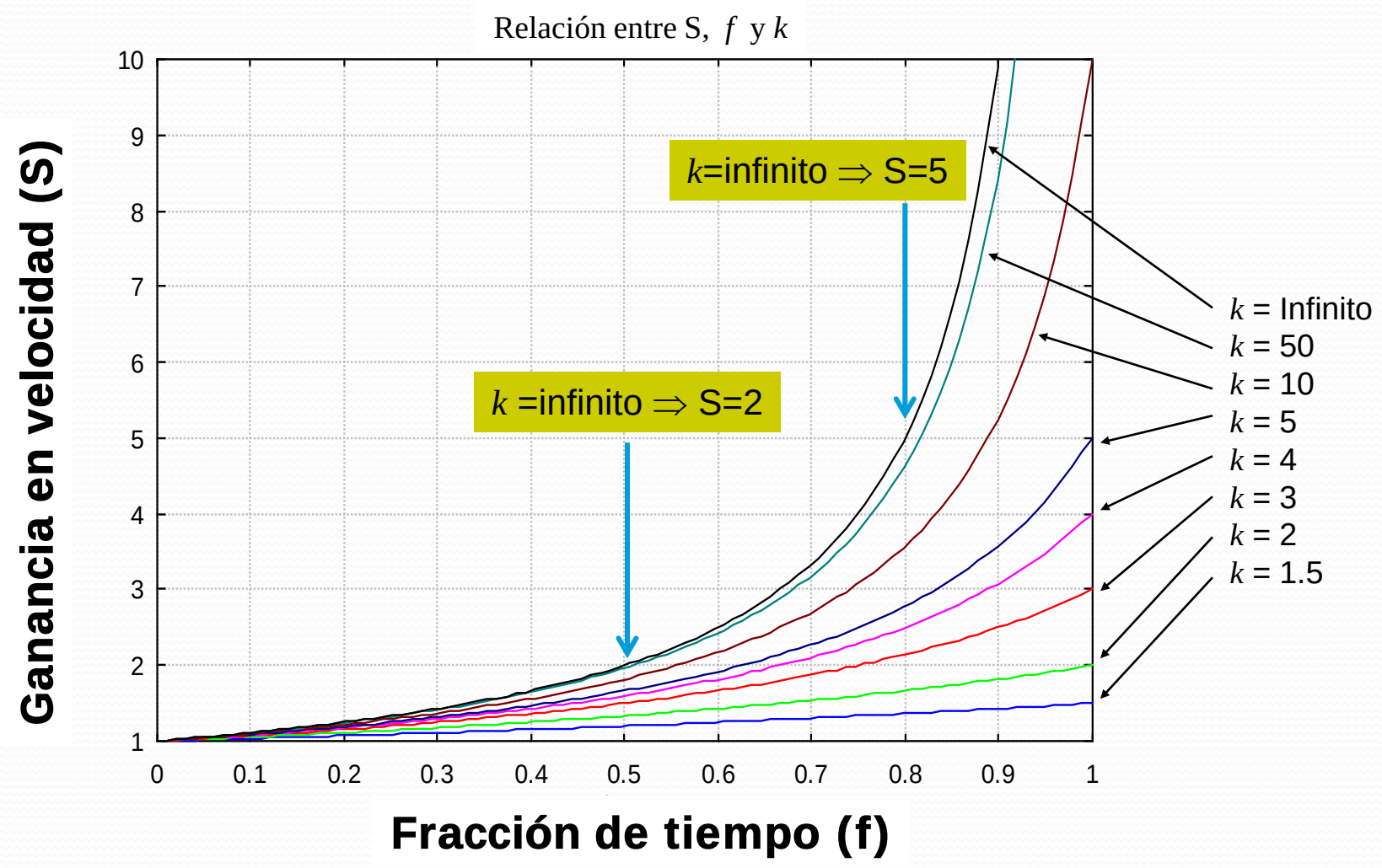
$$S_{max} = \lim_{k_{DD}\to\infty} S = \frac{T_o}{T_m^{k_{DD}\to\infty}} = \frac{T_o}{(1-0{,}6)\times T_o} = \frac{1}{1-0{,}6} = 2{,}5$$

➡ El servidor como mucho podría llegar a ser 2,5 veces "más rápido" (un 150% más rápido) que antes en la ejecución de ese programa actuando sólo sobre el disco.

Otra forma: Usamos la Ley de Amdahl:

$$S_{max} = \lim_{k_{DD}\to\infty} S = \lim_{k_{DD}\to\infty} \frac{1}{1 - f_{DD_O}^{T_o} + {f_{DD_O}^{T_o}}/{k_{DD}}} = \lim_{k_{DD}\to\infty} \frac{1}{1-0{,}6+{0{,}6}/{k_{DD}}} = \frac{1}{1-0{,}6} = 2{,}5$$

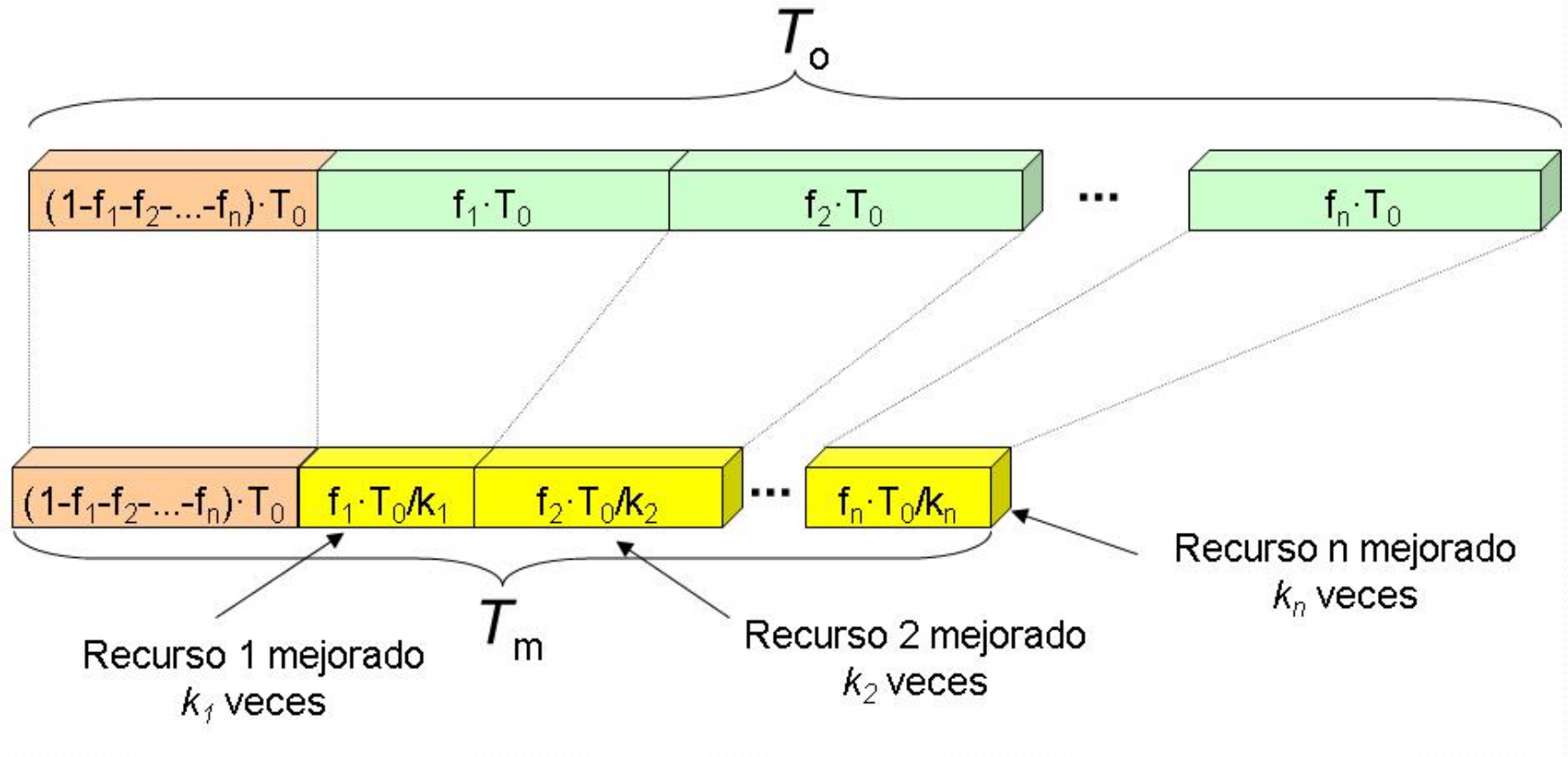
Análisis: Relación entre S, f y k



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?**