# Manual de supervivencia para DCE

**Atención:** El temario que en este manual se trata corresponde a los temas 1, 2, y 3 de la asignatura de *Diseño, configuración y evaluación de los sistemas informáticos* que se impartió en el curso 2020-2021. Tanto los ejercicios resueltos que en este documento se hallan, como las explicaciones teóricas, son una combinación de las diapositivas de la asignatura con los apuntes que realicé durante el curso. Es por ello, que es posible que algún ejercicio no esté bien resuelto (aunque creo que sí), pero con todo, este manual me ayudó a aprobar la asignatura de DCE, y espero que también te ayude a tí.

# TEMA 1

## COMPARACIÓN DE PRESTACIONES

Comparar prestaciones entre dos sistemas puede permitir saber cuál es más rápido (rendimiento), cuál es más caro (coste) y cuál tiene una relación rendimiento/coste más favorable para nosotros a la hora de elegir uno de los dos.

### *COMPARACIÓN DE RENDIMIENTO*

$$S = \frac{Tiempo\ de\ ejecución_M}{Tiempo\ de\ ejecución_m} = \frac{Rendimiento_M}{Rendimiento_m}$$

### *EJEMPLO DE COMPARACIÓN DE RENDIMIENTO*

Un programa determinado se ejecuta en 45 segundos en el computador A, y el mismo programa se ejecuta en 36 segundos en el computador B.

$$S = \frac{Tiempo\ de\ ejecución_M}{Tiempo\ de\ ejecución_m} = \frac{45}{36} = 1.25 = 1 + 0.25$$

Con estos datos podemos decir que el computador B es 1.25 veces más rápido que el computador A. Podemos decir también que el computador B es un 25% más rápido que el computador A.

### *COMPARACIÓN DE COSTES*

$$\Delta C = \frac{Coste_M}{Coste_m}$$

### *EJEMPLO DE COMPARACIÓN DE COSTES*

El computador A cuesta 649 €, y el computador B cuesta 729 €.

$$\Delta C = \frac{Coste_M}{Coste_m} = \frac{729}{649} = 1.12$$

Con estos datos podemos decir que el computador B es 1.12 veces más caro que el computador A. Podemos decir también que el computador B es un 12% más caro que el computador A.

*COMPARACIÓN DE RENDIMIENTO/COSTE*

$$\frac{Rendimiento}{Coste} = \frac{1}{Tiempo\ de\ ejecución * Coste}$$

*EJEMPLO DE COMPARACIÓN RENDIMIENTO/COSTE*

El computador A cuesta 649 € y es capaz de ejecutar un programa determinado en 45 segundos, y el computador B cuesta 729 € pero es capaz de ejecutar el mismo programa en 36 segundos.

$$\frac{Rendimiento_A}{Coste_A} = \frac{1}{45*649} = 3.42 * 10^{-5}$$

$$\frac{Rendimiento_B}{Coste_B} = \frac{1}{36*729} = 3.81 * 10^{-5}$$

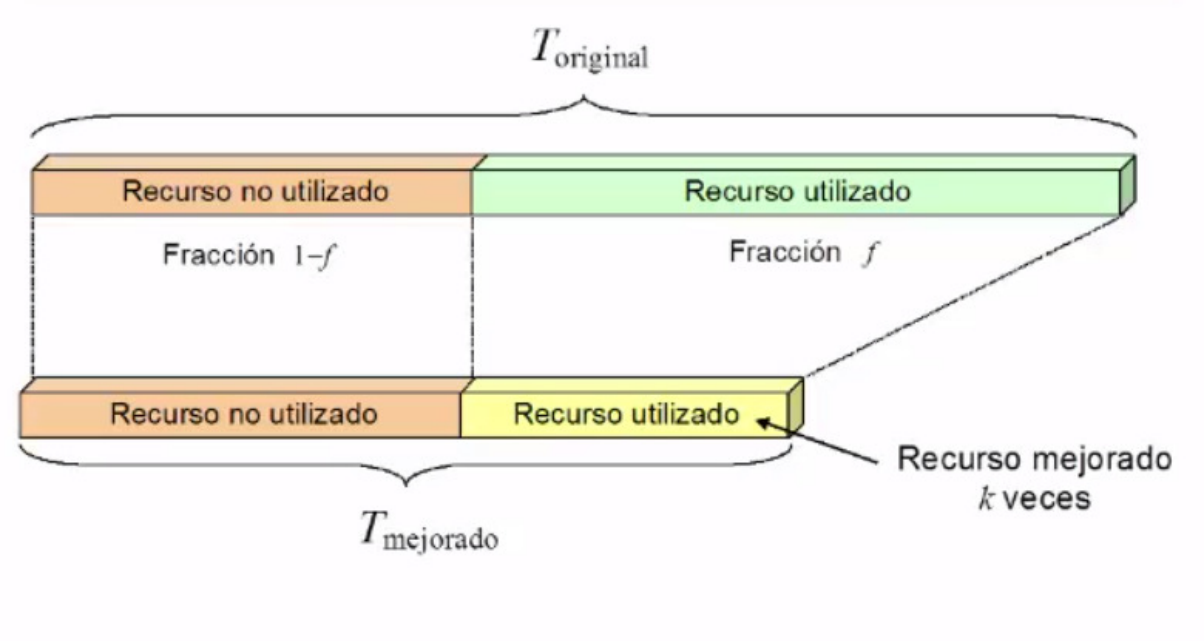
En este caso, el computador B presenta una relación rendimiento/coste ligeramente más alta que el computador A, por lo que sería más conveniente adquirir el computador B.

**Apunte:** Hay que evitar decir que el computador A es más lento que el computador B, el computador A es más barato que el computador B, o el computador A es menos conveniente que el computador B.

## LEY DE AMDAHL

**Definición:** Más allá de cierto punto, el hecho de agregar más procesadores a un sistema de procesamiento paralelo no produce una mejora significativa de la velocidad.

**¿Para qué se usa?:** Para calcular la mejora temporal de un sistema cuando se le realiza una o N mejoras a sus componentes con respecto al tiempo original.




Fórmula para calcular la mejora temporal:

$$T_{mejorado} = T_{original}((1 - f) + \frac{f}{k})$$

Fórmula para calcular la mejora de velocidad (*Speed up*) con la ley de Amdahl:

$$S = \frac{T_{original}}{T_{mejorado}} = \frac{1}{(1-f) + \frac{f}{k}}$$

Anotaciones:

Si $f$=0 entonces A=1
Si $f$=1 entonces A=$k$

*EJEMPLO DE UTILIZACIÓN DE LA LEY DE AMDAHL*

La utilización de un procesador es del 60%. ¿En cuánto aumentará el rendimiento del sistema si se duplica la velocidad del procesador? ($k$=2)

$$S = \frac{1}{(1-0.60) + \frac{0.60}{2}} = 1.43 \rightarrow$$

El rendimiento del sistema aumentará en un 43%

*CÁLCULO DE LA ACELERACIÓN MÁXIMA*

$$\lim_{k \to \infty} S = \frac{1}{(1-f)} = \frac{1}{(1-0.60)} = 2.5$$

*RESUMEN LEY DE AMDAHL*

| Para una sola mejora: | Para $n$ mejoras (sin solapamiento): |
|---|---|
| $S = \frac{1}{(1-f) + \frac{f}{k}}$ | $S = \frac{1}{(1 - \sum_{i=1}^{n} f_i) + \sum_{i=1}^{n} (\frac{f_i}{k_i})}$ |

*RENDIMIENTO CON VARIOS PROCESADORES*

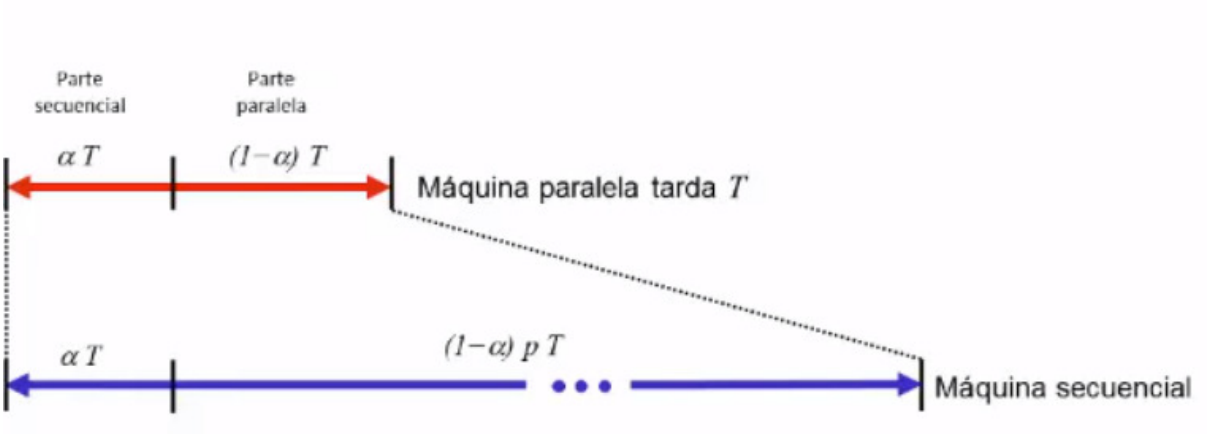
En un sistema con p procesadores, el tiempo de ejecución de la aplicación que se planea paralelizar dispone de una fracción secuencial del 10% del tiempo total, y de una fracción paralelizable del 90%.
**¿Cuál es la aceleración obtenida con 15 procesadores (p=15)?**
**¿Cuál es la aceleración máxima posible?**

| Aceleración con 15 procesadores (p=15) | Aceleración máxima posible |
|---|---|
| $S = \frac{1}{f_{secuencial} + \frac{f_{paralelizable}}{p}}$ <br> $S = \frac{1}{0.10 + \frac{0.90}{15}} = 6.25$ | $\lim_{p \to \infty} S = \frac{1}{f_{secuencial}}$ <br> $\lim_{p \to \infty} S = \frac{1}{0.10} = 10$ |

## LEY DE GUSTAFSON

Definición: La cantidad de trabajo que se puede hacer en paralelo varía linealmente con el número de procesadores

**¿Para qué se usa?:** Para calcular la aceleración proporcional que tiene un modelo con p procesadores frente a uno con un solo procesador.




Fórmula para calcular la ley de Gustafson

$$S(p) = \frac{\alpha T + (1-\alpha) * pT}{T} = p - \alpha(p-1)$$

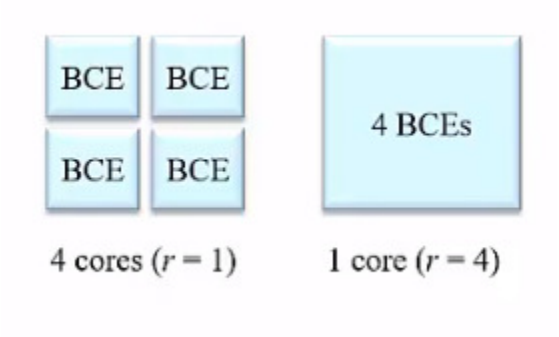
Cabe destacar el hecho que, teniendo en cuenta la relación utilizada en esta fórmula, si la parte secuencial tiende a 0, A(p) = p.

## LEY DE AMDAHL EN CHIPS MULTICORE

Es necesario tener en cuenta una serie de factores hipotéticos (los mismos que se usarán en los enunciados) para poder realizar las siguientes fórmulas correctamente:

1. Los cores tienen memoria caché L1.
2. Un chip contiene n BCEs (o cores) elementales de rendimiento normalizado a 1.
3. Se pueden combinar r BCEs para obtener un rendimiento$perf(r)$, pero al combinar los cores su rendimiento ya no será 1, sino que será cuadrático al número de cores que se haya utilizado.

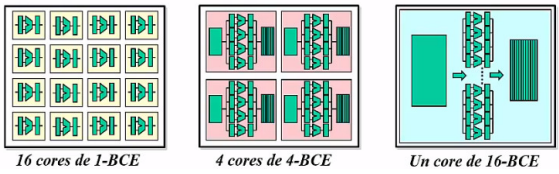
   Es decir: $perf(r) = \sqrt{r}$




### *CHIPS SIMÉTRICOS*

**Propiedades:**

- ➔ Todos los cores del chip tienen el mismo coste.
- ➔ Con 16 BCEs existen diferentes combinaciones:
  - ◆ 16 cores de 1 BCE cada uno.
  - ◆ 8 cores de 2 BCEs cada uno.
  - ◆ 4 cores de 4 BCEs cada uno.
  - ⧫ 1 core de 16 BCEs.




### *LEY DE AMDAHL CON CHIPS SIMÉTRICOS*

- ➔ Para la fracción secuencial se usará un core de rendimiento perf(r)
- ➔ Para la fracción paralela se usarán n/r cores de rendimiento perf(r)

$$S_{SIM} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{\frac{n}{r} * perf(r)}}$$

*EJEMPLO DE APLICACIÓN DE LA LEY DE AMDAHL PARA CHIPS SIMÉTRICOS*

En un chip con una capacidad de 16 cores básicos (n=16), se decide agrupar los BCEs en 4 chips de 4 BCEs cada uno (r=4).

$$S_{SIM} = \frac{1}{\frac{1-f}{\sqrt{4}} + \frac{f}{\frac{16}{4} * \sqrt{4}}}$$

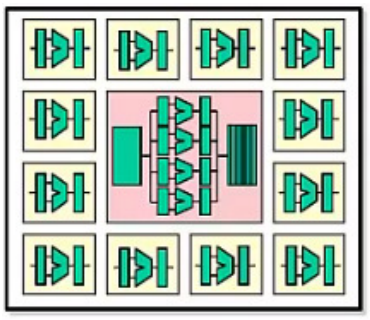
**Anotaciones:**

Si f es 1, entonces la mejor combinación para los chips simétricos sería r=1 y n=16.

*CHIPS ASIMÉTRICOS*

**Propiedades:**

- ➔ Combinan parte de los n BCEs para hacer un core más potente.
- ➔ Un chip tiene 1 core que combina r BCEs y los n-r BCEs restantes son cores elementales.
- ➔ El chip tiene 1+n-r cores de dos capacidades de procesamiento distintas.



*LEY DE AMDAHL CON CHIPS ASIMÉTRICOS*

- ➔ Para la fracción secuencial, se utilizará un core con rendimiento perf(r).
- ➔ Para la fracción paralelizable, se utilizará un core con perf(r) y n-r cores de rendimiento 1.

$$S_{ASIM} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}}$$

*EJEMPLO DE APLICACIÓN DE LA LEY DE AMDAHL PARA CHIPS ASIMÉTRICOS*

En un chip con una capacidad de 16 cores básicos (n=16), se decide agrupar los BCEs en 4 chips de 4 BCEs cada uno (r=4).

$$S_{ASIM} = \frac{1}{\frac{1-f}{\sqrt{4}} + \frac{f}{\sqrt{4}+16-4}}$$

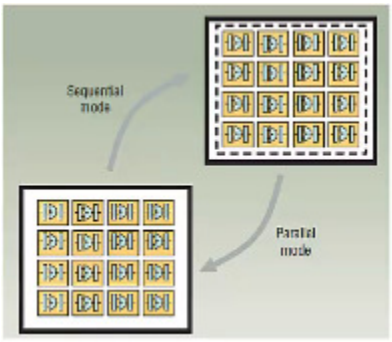
**Anotaciones:**

Una configuración asimétrica será en todo caso más rápida que una simétrica.

## CHIPS DINÁMICOS

**Propiedades:**

- ➔ Aprovechan un core potente de r BCEs en la fracción secuencial.
- ➔ Emplean todos los n cores en la fracción paralela.
- ➔ Programables para ser configurados de manera dinámica mediante carga.




### *LEY DE AMDAHL CON CHIPS DINÁMICOS*

- ➔ Para la fracción secuencial, se utilizará un core con rendimiento perf(r).
- ➔ Para la fracción paralelizable, se utilizarán n cores de rendimiento 1.

$$S_{DIN} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$

### *EJEMPLO DE APLICACIÓN DE LA LEY DE AMDAHL PARA CHIPS DINÁMICOS*

En un chip con una capacidad de 16 cores básicos (n=16), se decide agrupar los BCEs en 4 chips de 4 BCEs cada uno (r=4).

$$S_{DIN} = \frac{1}{\frac{1-f}{\sqrt{4}} + \frac{f}{16}}$$

**Anotaciones:**

Una configuración dinámica será en todo caso más rápida que una asimétrica y, consecuentemente, que una simétrica.

Es conveniente que el valor de r sea lo más alto posible.

# CONFIABILIDAD DE SISTEMAS TOLERANTES A FALLOS

**Sistema tolerante a fallos:** Es un sistema que es capaz de continuar con su función habitual independientemente de los fallos que se puedan producir tanto a nivel de software como de hardware.

**Confiabilidad:** Propiedad que permite a los usuarios del sistema depositar una confianza justificada en este.

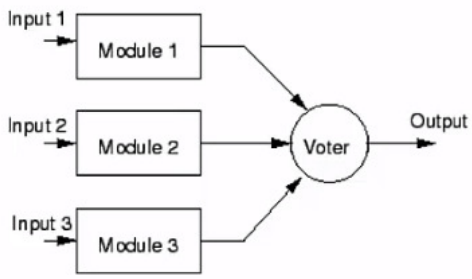
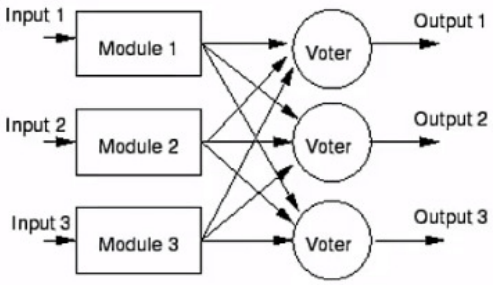
## *FACTORES DE LA CONFIABILIDAD*

1. Fiabilidad: El sistema funciona sin interrupciones.
2. **Disponibilidad:** El sistema está disponible el máximo tiempo posible.
3. **Mantenibilidad:** El sistema es fácilmente reparable.
4. **Seguridad-Inocuidad:** El sistema no provoca averías catastróficas (pone en juego vidas humanas).
5. **Seguridad-Confidencialidad:** El sistema impide el acceso no autorizado.
6. **Seguridad-Integridad:** El sistema impide la corrupción de la información.

*EJEMPLOS DE SISTEMAS TOLERANTES A FALLOS*

*SISTEMAS N-MODULAR REDUNDANTES*

**N-version programming:** Diseñar y codificar N versiones de un módulo software por N equipos de programadores distintos, que además deben ser independientes entre ellos. Se puede implementar en hardware y software.

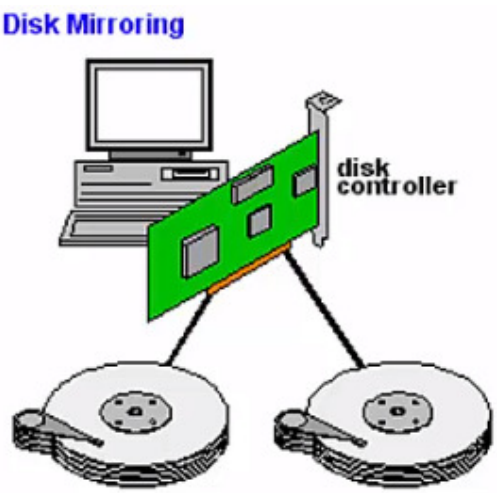
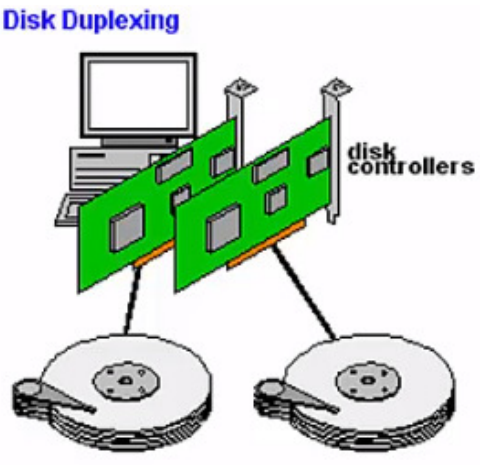
| Tolerancia al fallo de un módulo | Tolerancia al fallo de un módulo y de un detector de mayoría |
|---|---|
|  |  |



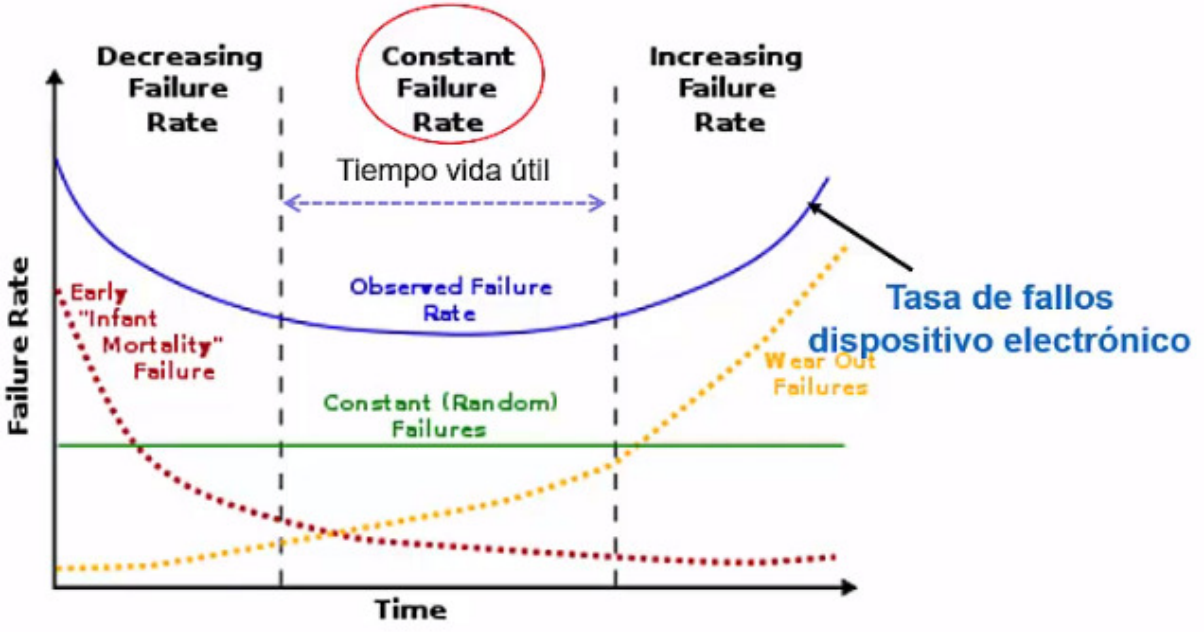


*DISCOS EN ESPEJO*

Únicamente redundante en hardware ya que se duplican recursos. Requiere que las operaciones de escritura se realicen simultáneamente en ambos discos.

| Tolerancia al fallo de un disco | Tolerancia al fallo de un disco y una controladora. |
|---|---|
|  |  |

*FIABILIDAD DE LOS DISPOSITIVOS ELECTRÓNICOS*




*MODELADO DE LA FIABILIDAD*

Considerando el tiempo T como el tiempo que transcurre hasta el próximo fallo o avería, siempre que T sea una variable aleatoria que se distribuye exponencialmente (es decir, que T se encuentr en el periodo de vida útil del sistema), su función de distribución será:

$$F(t) = P[T \leq t] = 1 - e^{-\lambda t}$$

Anotaciones:
λ= fallos/hora
$t$ = Tiempo (h)

Consecuentemente, con esta función también obtenemos la función de fiabilidad, que expresa la **probabilidad de que el sistema funcione correctamente más allá de un tiempo t.**

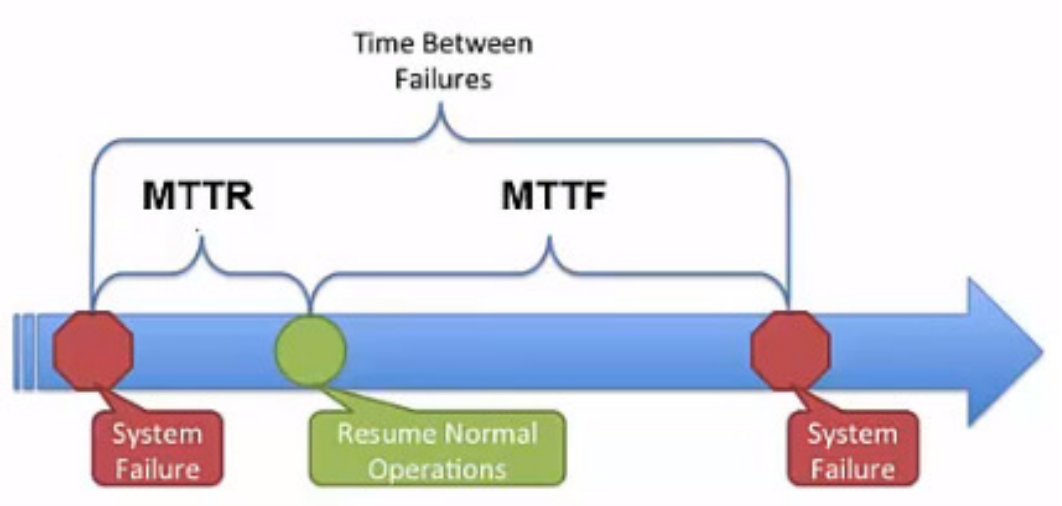
$$R(t) = P[T > t] = 1 - F(t) = e^{-\lambda t}$$

*TIEMPOS MEDIOS PARA POBLACIONES*

**MTTF:** (Mean Time To Failure) Tiempo medio en que trabajará un sistema antes de que se estropee por primera vez. **Durante el periodo de vida útil de un sistema, su MTTF es 1/λ.**

**MTTR:** (Mean Time To Repair) Tiempo medio en reparar un sistema.

**MTBF:** (Mean Time Between Failures) Tiempo medio entre dos fallos consecutivos. Su valor siempre será **MTBF = MTTF+MTTR**. Si un sistema **no es reparable** entonces **MTTR = 0**, por lo que **MTBF=MTTR**.




*DISPONIBILIDAD DE UN SISTEMA*

Es el porcentaje del tiempo promedio en que un sistema está operativo.

$$A = \frac{MTTF}{MTTF+MTTR} = \frac{MTTF}{MTBF}$$

# RESIDUOS INFORMÁTICOS E IMPACTO AMBIENTAL

**Huella de carbono:** La totalidad de gases de efecto invernadero emitidos por el efecto directo o indirecto de un individuo.

Los servidores y equipos se usan entre un 8 y un 15% del tiempo en que están encendidos, y el hardware x86 consume un 60-90% de energía en estado ocioso. Anualmente, un servidor emite 4 toneladas de $CO_2$ al año. La meta actual es optimizar al máximo el consumo eléctrico.

**Mochila ecológica:** Cantidad de materiales, energía, y recursos usados en la elaboración de un producto a lo largo de su ciclo de vida.

La fabricación de un ordenador consume 240 kg de combustibles, 22kg de productos químicos y 1500 litros de agua dulce, de manera que la mochila ecológica de un ordenador pesa 1500 kg, mientras que la de un smartphone pesa 75 kg en residuos.

Además de esto, la rápida obsolescencia de los sistemas genera que los residuos sobrepasen los 50 millones de toneladas al año.

Cuando los sistemas dejan de ser útiles, se convierten en **basura electrónica**. Su mala gestión es muy tóxica para el medio ambiente ya que el Plomo (Pb), el Mercurio (Hg), el Cadmio (Cd), el Selenio (Se) y el Arsénico (As) contaminan el aire, la tierra y el agua.

La basura tecnológica de EEUU, Europa y Japón (en gran medida), es enviada a países pobres. En China se vierte el 70% del material tecnológico mundial.

## PROBLEMAS

Si pulsas sobre el problema, te llevará a la parte de teoría donde se explica.

| PROBLEMA 1.1 | Una mejora en un sitio web ha permitido rebajar de 17 a 9 segundos el tiempo medio de descarga de sus páginas. Si la mejora ha consistido en hacer 3 veces más rápido el subsistema de discos que almacena las páginas del servidor, ¿Cuánto tiempo se dedicaba a acceder a los discos antes de realizar la mejora? |
|---|---|

En este caso nos están preguntando por el valor f de la ley de Amdahl, concretamente piden el producto del valor f por los segundos del tiempo medio original del problema. De este modo adaptaremos un poco la ley de Amdahl para que pueda cumplir nuestras necesidades. Para ello tendremos en cuenta que el valor de S es 17/9, es decir 1.89.

$$S = \frac{1}{(1-f)+\frac{f}{k}} \rightarrow f = \frac{k(S-1)}{S(k-1)} \rightarrow f = \frac{3(1.89-1)}{1.89(3-1)} = 0.70$$

Una vez realizado el cálculo, podemos deducir que el porcentaje de consumo del sitio web a la hora de descargar sus páginas era del 70%. Podemos proceder a resolver el ejercicio multiplicando $17 * 0.7$cuyo resultado es 11.89 segundos, valor que podemos redondear a 12 segundos.

| PROBLEMA 1.2 | El simulador de combates de SHINRA tarda 100 segundos en ejecutar un programa de inmersión completa para el entrenamiento de sus operarios SOLDADO. El programa dedica el 30% en hacer operaciones de aritmética entera, el 60% en hacer operaciones de aritmética en coma flotante, y el resto se emplea en operaciones de entrada/salida. Calcule el tiempo de ejecución si las operaciones aritméticas enteras y reales se aceleran de manera simultánea 2 y 3 veces respectivamente. |
|---|---|

Dado que ambos procesos de cálculo no se solapan entre ellos, es posible aplicar la ley de Amdahl para n mejoras por lo que

$$S = \frac{1}{(1-(0.30+0.60))+(\frac{0.30}{2}+\frac{0.60}{3})} = \frac{1}{(1-0.90)+0.35} = 2.22$$

Dado que el enunciado pide el resultado en formato temporal, solo resta dividir el tiempo original entre S, es decir $T_{mejorado} = \frac{100}{2.22} = 45$segundos

| PROBLEMA 1.3 | Una aplicación informática se ejecuta en un computador durante un total de 70 segundos. Mediante el uso de un monitor de actividad, se ha podido saber que el 85% del tiempo se utiliza en la tarjeta de red, mientras que el resto del tiempo se hace uso del procesador. Se pide:<br>1. Calcular el incremento de prestaciones si se mejora en 8 veces la velocidad de la tarjeta de red.<br>2. Determinar en cuánto hay que mejorar el rendimiento del procesador si se quiere ejecutar la aplicación en 25 segundos. |
|---|---|

El primer paso es recopilar datos, f=0.85 para la tarjeta de red, y f=0.15 para el procesador.

En estos casos es conveniente primero verificar si es posible lo que nos están pidiendo, porque aplicar el límite de la ley de Amdahl siempre será más rápido que hacer los cálculos necesarios para encontrar un valor k que se ajuste a lo que nos pide el enunciado.

De modo que procederemos a aplicar el límite a la ley de Amdahl.

$$\lim_{k \to \infty} S = \frac{1}{(1-f)} = \frac{1}{(1-0.15)} = 1.17$$

Dado que la mejora máxima del procesador tan solo permitiría que la ejecución de la aplicación durase $\frac{70}{1.17} = 59.82$segundos, podemos afirmar que el apartado 2 es imposible mejorando únicamente el procesador. En cuanto al apartado 1 tan solo tenemos que aplicar la ley de Amdahl

$$S = \frac{1}{(1-0.85) + \frac{0.85}{8}} = 3.9$$

Pudiendo afirmar que con la mejora mencionada en la tarjeta de red, el incremento de prestaciones mejoraría en un 290%.

| PROBLEMA 1.4 | Han Solo planea modificar el halcón milenario para poder continuar traficando objetos valiosos sin que la nueva orden lo atrape. Indíquese qué opción de modificación de un carguero ligero corelliano YT-1300 de las dos que se enumeran resultará más ventajosa:<br>1. Cambio del motor central (250 créditos). Esta modificación permite que el 75% de los elementos de la nave funcionen dos veces más rápidamente.<br>2. Adición de otro reactor de propulsión (150 créditos). Añadir otro reactor de propulsión mejorará hasta 3 veces el rendimiento del 40% de los elementos de la nave. |
|---|---|

Para la opción 1 tenemos que f=0.75 y k=2.
Para la opción 2 tenemos que f=0.40 y k=3.

$$S_{Motor} = \frac{1}{(1-0.75) + \frac{0.75}{2}} = 1.6$$

$$S_{Reactor} = \frac{1}{(1-0.40) + \frac{0.40}{3}} = 1.36$$

Contando con que la opción 1 cuesta 250 créditos y la opción 2 cuesta 150 créditos, es el momento de realizar el cálculo de la relación rendimiento/coste con los datos obtenidos.

$$\frac{Rendimiento_{Motor}}{Coste_{Motor}} = \frac{1}{1.6*250} = 2.5 * 10^{-3}$$

$$\frac{Rendimiento_{Reactor}}{Coste_{Reactor}} = \frac{1}{1.36*150} = 4.9 * 10^{-3}$$

Dado que la relación rendimiento/coste de la adición de un nuevo reactor es más ventajosa que la del cambio del motor, lo conveniente sería añadir otro reactor de propulsión.

| PROBLEMA 1.5 | Un programa de predicción meteorológica tarda 84 minutos en ejecutarse en un supercomputador diseñado al efecto. Sin embargo, esta cantidad de tiempo origina muchos problemas para los estudios de los meteorólogos. El responsable del equipo informático quiere reducir este tiempo sustituyendo la memoria principal por una más rápida, para lo cual existen dos modelos alternativos:<br>1. Modelo Skynet (900€), que disminuye el tiempo de ejecución hasta los 71 minutos.<br>2. Modelo GLaDOS (1300€), que rebaja este tiempo de ejecución hasta los 63 minutos.<br>Determine cual de estas dos opciones es la mejor opción |
|---|---|

Este ejercicio es incluso más fácil que el anterior porque no es necesario ni siquiera aplicar la ley de Amdahl, con la información que tenemos nos sobra para poder realizar directamente el cálculo de la relación rendimiento/coste.

$$\frac{Rendimiento_{Skynet}}{Coste_{Skynet}} = \frac{1}{71*900} = 1.56 * 10^{-5}$$

$$\frac{Rendimiento_{GLaDOS}}{Coste_{GLaDOS}} = \frac{1}{63*1300} = 1.22 * 10^{-5}$$

Dado que la relación rendimiento/coste del modelo Skynet es más favorable, será este la opción más ventajosa.

| PROBLEMA 1.6 | Indique la expresión de la aceleración proporcional para un sistema con p procesadores que ejecuta una aplicación que tiene una fracción secuencial α = 0,1 |
|---|---|

Aplicando la ley de Gustafson para este ejercicio tenemos que

$$S(p) = \frac{\alpha T+(1-\alpha)*pT}{T} = p - \alpha(p-1) \rightarrow S(p) = p - 0.1(p -$$

| PROBLEMA 1.7 | Una aplicación dedicada a detectar replicantes tarda 2 minutos en ejecutarse en un procesador monocore. A fin de reducir el tiempo de ejecución se quiere sustituir este chip por otro multicore con espacio suficiente para 8 cores elementales o BCEs). Un estudio ha mostrado que dicha aplicación se puede paralelizar en un 90 %. Calcule el tiempo que tardaría en ejecutarse esta aplicación si el chip se configura de manera simétrica con las siguientes agrupaciones de cores:<br>➔ 8 cores elementales (r = 1)<br>➔ 4 cores dobles (r = 2)<br>➔ 2 cores cuádruples (r = 4) |
|---|---|

Para facilitar el resultado, normalizaremos los minutos a segundos, sabiendo que 2 minutos son 120 segundos.
Podemos identificar que el valor f del enunciado es 0.90, y el valor n es 8, por lo que aplicando la ley de Amdahl para los chips simétricos los resultados serían los siguientes:

| 8 cores elementales | 4 cores dobles | 2 cores cuádruples |
|---|---|---|
| $S_S = \frac{1}{\frac{1-0.9}{1}+\frac{0.9}{8}} = 4.7$ | $S_S = \frac{1}{\frac{1-0.9}{\sqrt{2}}+\frac{0.9}{\frac{8}{2}*\sqrt{2}}} = 4.35$ | $S_S = \frac{1}{\frac{1-0.9}{\sqrt{4}}+\frac{0.9}{\frac{8}{4}*\sqrt{4}}} = 3.64$ |
| $\frac{120}{4.7} = 25.5\ segundos$ | $\frac{120}{4.35} = 27.57\ segundos$ | $\frac{120}{3.64} = 33\ segundos$ |

En base a los datos obtenidos, podemos afirmar que se obtiene un tiempo mejor cuando se utilizan 8 cores elementales.

PROBLEMA 1.8

Una aplicación dedicada a la investigación sobre el prototipo de Virus T de Umbrella Corporation tarda una hora en ejecutarse. Se pretende acelerar esta ejecución mediante la inclusión en el sistema informático de un chip multicore con 6 cores elementales o BCEs. Se sabe que la aplicación se puede paralelizar en un 90 %. El chip multicore permite las siguientes tres configuraciones:

- ➔ Chip simétrico con 6 cores elementales (r=1)
- ➔ Chip asimétrico con un core cuádruple (r=4) y dos cores elementales
- ➔ Chip dinámico con un core doble (r=2) para la fracción secuencial

Calcule qué opción es más ventajosa. ¿Qué habría pasado si la fracción paralelizable fuera del 70%?

Para facilitar el resultado, normalizaremos las horas a minutos, sabiendo que una hora son 60 minutos.
Podemos identificar que el valor f del enunciado es 0.90, y el valor n es 8, por lo que aplicando la ley de Amdahl para los chips simétricos los resultados serían los siguientes:

| Simétrico con r=1, f=0.9 | Asimétrico con r=4, f=0.9 | Dinámico con r=2, f=0.9 |
|---|---|---|
| $S_S = \frac{1}{\frac{1-0.9}{1}+\frac{0.9}{6}} = 4$ | $S_A = \frac{1}{\frac{1-0.9}{\sqrt{4}}+\frac{0.9}{\sqrt{4}+6-4}} = 3.63$ | $S_D = \frac{1}{\frac{1-0.9}{\sqrt{2}}+\frac{0.9}{6}} = 4.5$ |
| $\frac{60}{4} = 15\ minutos$ | $\frac{60}{3.63} = 16.5\ minutos$ | $\frac{60}{4.5} = 13.24\ minutos$ |

Como cabía esperar, la opción más ventajosa es la del chip dinámico, pero es interesante el hecho de que el chip simétrico sea más ventajoso que el asimétrico. Esto ocurre porque el valor r del simétrico es 1 y aprovecha todo su potencial dado que además, la fracción paralelizable es grande.

En el caso de que la fracción paralelizable fuese del 70%, los valores serían los siguientes:

| Simétrico con r=1, f=0.7 | Asimétrico con r=4, f=0.7 | Dinámico con r=2, f=0.7 |
|---|---|---|
| $S_S = \frac{1}{\frac{1-0.7}{1}+\frac{0.7}{6}} = 2.4$ | $S_A = \frac{1}{\frac{1-0.7}{\sqrt{4}}+\frac{0.7}{\sqrt{4}+6-4}} = 3.0$ | $S_D = \frac{1}{\frac{1-0.7}{\sqrt{2}}+\frac{0.7}{6}} = 3.04$ |
| $\frac{60}{4} = 25\ minutos$ | $\frac{60}{3.63} = 19.5\ minutos$ | $\frac{60}{4.5} = 19.72\ minutos$ |

En este caso, si bien con la configuración anterior el chip simétrico tenía alguna oportunidad contra el asimétrico, cuando la fracción paralelizable disminuye, el chip simétrico pierde todas sus cualidades. No obstante en esta configuración sale ganando el chip asimétrico contra el dinámico.

| PROBLEMA 1.9 | Se estima que el ventilador de un procesador del último modelo de droide personal tiene un MTBF de 50000 horas. Calcule la probabilidad de que el ventilador sufra una avería durante los 3 primeros años de funcionamiento. ¿Cuál es la probabilidad de que el ventilador siga funcionando correctamente transcurridos estos 3 años? |
|---|---|

Teniendo en cuenta que los 3 primeros años pertenecen a la vida útil del sistema, y que el enunciado da a entender que transcurridas las 50000 horas el sistema dejará de funcionar y no será reparable, MTTF=MTBF, además, podemos asumir que su MTTF = 1/λ. Teniendo eso claro, podemos despejar el valor de λcomo

$$\lambda = \frac{1}{MTTF} = \frac{1}{50000} = 2 * 10^{-5}$$

Una vez tenemos el valor de λ, podemos utilizarlo para calcular la probabilidad de que falle durante su vida útil con

$$F(t) = P[T \leq t] = 1 - e^{-0.2*5} = 0.63$$

La probabilidad de que el sistema sufra una avería durante los 3 primeros años de funcionamiento es del 63%.

La probabilidad de que el ventilador siga funcionando correctamente transcurridos los 3 años es de 1-F(t), es decir, de un 37%

# TEMA 2

## ANÁLISIS DE PROGRAMAS EN SISTEMAS UNIX

La monitorización (o profiling), es una técnica para obtener información sobre la ejecución de los programas. Responde a las preguntas:

- ¿Dónde pasa la mayor parte del tiempo de ejecución?
- ¿Cuántas veces se ejecuta una línea de programa?
- ¿Dónde está Wally?
- ¿Cuántas veces se llama un procedimiento y desde donde?
- ¿Qué funciones se llaman desde un determinado procedimiento?

En Unix existen dos herramientas destinadas a la monitorización, las órdenes **gprof** (orientada al análisis de procedimientos), y **gcov** (orientada al análisis de líneas y bloques de instrucciones).

Las etapas a seguir serán:

1. Compilar el programa habilitando la recogida de información.
2. Ejecutar el programa instrumentado (ejecución lenta).
3. Analizar la información contenida en el fichero de comportamiento.

## MEDIDAS DE LA ACTIVIDAD

### *TÉCNICAS DE MEDIDA*

**Detección de eventos:**

- Registra el estado del sistema.
- Registra los cambios de estado que provocan los eventos.
- El volumen de la información recogida depende de la frecuencia de los eventos.
- Una gran parte de los eventos pueden ser detectados por software.
- Ejemplo de eventos:
  - Inicio/fin de ejecución de un programa.
  - Activación de señales RD y WR en memoria.
  - Acierto/fallo en memoria caché.
  - Abrir/cerrar un fichero

**Muestreo:**

- Análisis estadístico de datos más fácil de utilizar.
- El volumen de la información recogida y de su precisión dependen del tiempo de ejecución T.

*MONITORES SOFTWARE*

**Propiedades:**

- ➔ Son los más usados.
- ➔ Se activan cuando se ejecutan las instrucciones, lo que genera una sobrecarga.
- ➔ Se pueden implementar añadiendo el monitor al código del programa, añadiendo el programa al monitor o modificando el sistema operativo.

*CÁLCULO DE LA SOBRECARGA*

La ejecución de las instrucciones del monitor se lleva a cabo en el procesador del sistema monitorizado. La sobrecarga se puede calcular con la siguiente fórmula.

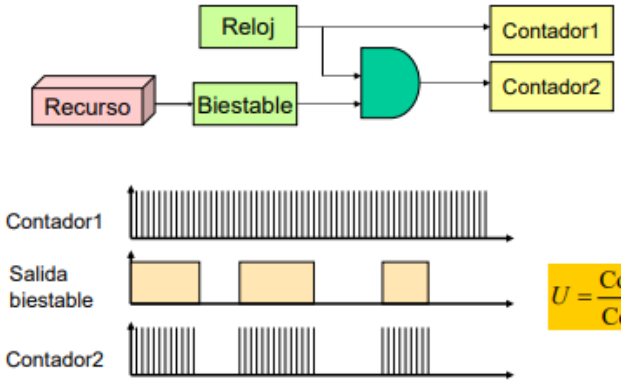
$$Sobrecarga = \frac{Tiempo\ de\ ejecución\ del\ monitor}{Intervalo\ de\ medida}$$

Sabiendo esto, si el monitor se activa cada 5 s y cada activación del mismo usa el procesador durante 6 ms, su sobrecarga se calculará de la siguiente manera:

$$Sobrecarga = \frac{6*10^{-3}}{5} = 0.0012 = 0.12\%$$

*MONITORES HARDWARE*

- ➔ Instrumentos independientes del sistema conectados a este mediante sondas electromagnéticas.
- ➔ Ventajas:
  - ◆ No usan recursos del sistema monitorizado.
  - ◆ Son muy rápidos.
- ➔ Desventajas:
  - ◆ Los sistemas no facilitan la instalación de sondas.
  - ◆ El personal que los usa debe de ser especializado.
  - ◆ Hay magnitudes no accesibles por hardware.
  - ◆ Se pueden producir posibles perturbaciones electromagnéticas.



### *MONITORES HÍBRIDOS*

Disponen de una parte de hardware y una de software, donde la parte de software añade instrucciones especiales de I/O al sistema operativo y actúa como una sonda que recoge la información y la envía a la parte de hardware y la parte hardware actúa como un dispositivo de I/O que guarda, analiza y procesa la información enviada por la parte de software.

## MONITORIZACIÓN EN UNIX

### *HERRAMIENTAS DE MEDIDA (COMANDOS)*

- ➔ time: Mide el tiempo de ejecución de un programa.
  - ◆ real: Tiempo total usado por el sistema
  - ◆ user: Tiempo de CPU ejecutando en modo usuario
  - ◆ sys: Tiempo de CPU en modo supervisor
- ➔ who: Quién está conectado al sistema.
- ➔ w: Quién está conectado al sistema y qué hace.
- ➔ uptime: Tiempo que lleva el sistema en marcha y la carga media que soporta.
- ➔ ps: Información sobre el estado de los procesos del sistema.
- ➔ top: Carga media, procesos y consumo de memoria.
- ➔ mpstat: Información relativa a estadísticas de uso de los procesadores.
- ➔ vmstat: Paginación, swapping, interrupciones y cpu.
- ➔ df: Espacio utilizado por los archivos del sistema en el disco.
- ➔ du: Espacio utilizado por los archivos.
- ➔ hdparm: Parámetros del disco duro.
- ➔ cat /proc
  - ◆ /cpuinfo
  - ◆ /meminfo
  - ◆ /interrupts
  - ◆ /devices

*MONITOR gprof*

**Propiedades:**

- ➔ **Técnica de medida** mediante muestreo estadístico (cada 10 ms) y ejecución de llamadas al sistema.
- ➔ Ordenación según el consumo de CPU
- ➔ No adecuado para funciones recursivas
- ➔ Tipos de tiempos mostrados:
  - ◆ total/s per call: Tiempo que tarda en ejecutarse una función.
  - ◆ self/s per call: Tiempo que tarda en ejecutarse el código propio de una función sin depender las llamadas a otras funciones.

Suponiendo que nuestro programa se llame prog.c, su utilización sería la siguiente:

| | |
|---|---|
| gcc prog.c -o prog **-pg -g -a** | Instrumentación en la compilación |
| ./prog | Ejecución del programa que generará un fichero de información llamado gmon.out |
| gprof prog > prog.gprof | Visualización de la información referida a la ejecución del programa |

*VISUALIZACIÓN DE SALIDA DE gprof SOBRE CÓDIGO*

```
#include <stdio.h>
#include <time.h>
#include <math.h>
double a=3.14,b=6.34,c=-3.03;
long y;

void main()
{
  producto(); producto(); producto();
  division(); division();
  atangente();
}

producto()
{for (i=0; i<50000000; y++) c=a*b;}

division()
{for (i=0; i<30000000; y++) c=a/b;}

atangente()
{for (i=0; i<30000000; y++) c=atan(a);}
```

flat profile

```
Each sample counts as 0.01 seconds
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 62.79     11.12    11.12        2  5560.00  5560.00  division
 20.33     14.72     3.60        1  3600.00  3600.00  atangente
 16.88     17.71     2.99        3   996.67   996.67  producto
```

Tiempo total

call profile

Código propio

```
index % time    self  children    called     name

[1]    100.0    0.00   17.71                 main [1]
               11.12    0.00       2/2       division [2]
                3.60    0.00       1/1       atangente [3]
                2.99    0.00       3/3       producto [4]
-----------------------------------------------------------
               11.12    0.00       2/2       main [1]
[2]     62.8   11.12    0.00       2         division [2]
-----------------------------------------------------------
                3.60    0.00       1/1       main [1]
[3]     20.3    3.60    0.00       1         atangente [3]
-----------------------------------------------------------
                2.99    0.00       3/3       main [1]
[4]     16.9    2.99    0.00       3         producto [4]
-----------------------------------------------------------
```

*OPCIONES DISPONIBLES EN gprof*

**Opciones de compilación (**gcc**):**

- ➔ Información sobre funciones: -pg
- ➔ Información línea a línea del código fuente: -g
- ➔ Información por bloques de instrucciones: -a

**Opciones de obtención de información (**gprof**):**

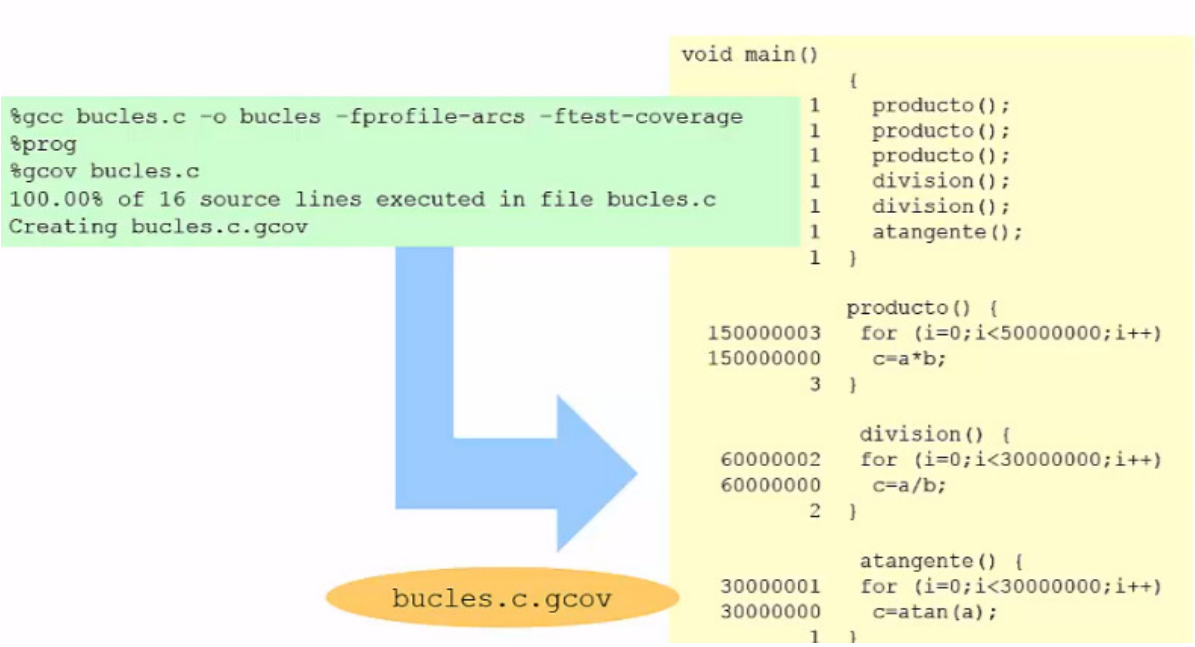
- ➔ Predeterminadas: -p (flat profile) -q (call profile)
- ➔ Informe sin explicaciones: -b
- ➔ Informe por línea de código: -l
- ➔ Informe por bloque: -A

*MONITOR gcov*

Aporta información sobre el número de veces que se ejecuta una línea de código.

| | |
|---|---|
| gcc prog.c -o prog **-fprofile-arcs -ftest-coverage** | Instrumentación en la compilación |
| ./prog | Ejecución del programa que generará varios archivos |
| gcov prog.c | Visualización de la información referida a la ejecución del programa (genera archivo prog.c.gcov) |

*VISUALIZACIÓN DE SALIDA DE gcov SOBRE CÓDIGO*




*OPCIONES DISPONIBLES EN gcov*

- ➔ Frecuencias de salto en instrucciones de salto: -b
- ➔ Informe por funciones: -f

*MONITOR sar*

Se basa en dos órdenes complementarias:

- ➔ sadc: Recoge los datos estadísticos y construye un registro en formato binario.
- ➔ sar: Lee los datos binarios que recoge sadc y los traduce a un formato legible por el ojo humano en formato texto.

El monitor utiliza un fichero histórico de datos por cada día y programa la ejecución de sadc un número de veces al día con la utilidad "cron" del sistema Unix (Por ejemplo, cada 5 minutos). Cada ejecución de sadc añadirá un registro binario con los datos recogidos al fichero histórico del día.

*EJEMPLO DE FICHERO HISTÓRICO DE sar*

```
-rw-r--r--    1   root   root    3049952   Oct  2   23:55    sa02
```

**Observaciones:**

- ➔ El fichero histórico de ese día ocupa 3049952 bytes (unos 3 MiB).
- ➔ Dada la hora del registro, podemos deducir que se ejecuta cada 5 minutos. Por lo que:
  - ◆ Cada hora se recogen 60/5 = 12 muestras.
  - ◆ Al día se recogen 24*12 = 288 muestras.
- ➔ Por lo tanto, dividiendo lo que ocupa el histórico por las muestras, podemos deducir que cada registro ocupa unos 10.3 KB.

*OPCIONES DISPONIBLES EN sar*

- ➔ Utilización del procesador: -u
- ➔ Paginación de la memoria virtual: -B
- ➔ Creación de procesos: -c
- ➔ Transferencias con I/O: -b
- ➔ Transferencias para cada disco: -d
- ➔ Sistema de interrupciones: -I (i mayúscula)
- ➔ Conexión de red: -n
- ➔ Carga media del sistema: -q
- ➔ Sistema de memoria: -r
- ➔ Cambios de contexto: -w
- ➔ Intercambio (swapping): -W
- ➔ Estadísticas sobre un proceso: x PID

*EJEMPLOS DE EJECUCIÓN DE sar*

| | |
|---|---|
| sar 2 30 | Ejecución interactiva |
| sar -A | Información recogida el día de hoy |
| sar -d -s 10:00 -e 12:00 | Información recogida sobre el día de hoy desde las 10:00 hasta las 12:00 |
| sar -f /var/log/sa/sa02 | Información recogida relativa a los procesadores del segundo día del mes |
| sar -b -f /var/log/sa/sa06 | Información recogida relativa a I/O del sexto día del mes |
| sar -d -s 10:00 -e 12:00 -f /var/log/sa/sa04 | Información recogida relativa a los discos el cuarto día del mes entre las 10:00 y las 12:00 |

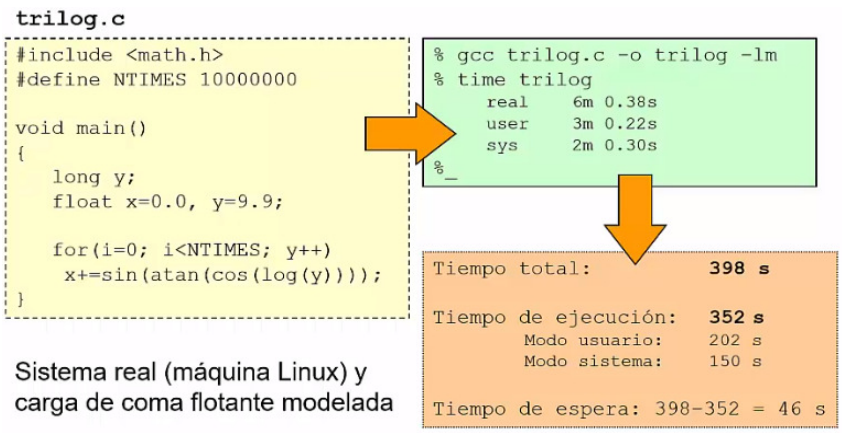
# TEMA 3

Estudio de referenciación (benchmarking): Todas aquellas técnicas utilizadas en la comparación del rendimiento de diferentes sistemas informáticos. Es necesario que todos los sistemas que se comparen estén sometidos a las mismas condiciones de carga.

## MEDIDAS DE RENDIMIENTO

- **Tiempo de ejecución:** Aquel sistema que ejecute la máxima cantidad de trabajo en el menor tiempo posible, será el sistema más rápido.
- **Repetibilidad:** Todas y cada una de las medidas de rendimiento han de ser reproducibles indicando las condiciones en que se han obtenido.
    - Evitar confundir el rendimiento del procesador con el del sistema. Hay que tener en cuenta muchos más componentes ya que habitualmente, el procesador no es la parte limitante del sistema.
    - Es importante también tener en cuenta los elementos hardware de cada equipo, su procesador, placa base, discos...
    - El sistema operativo también condiciona al rendimiento, ya que en un mismo hardware, un sistema operativo puede rendir más que otro.
    - El sistema de memoria también tendrá una gran influencia sobre los resultados de la medida (Configuraciones de las memorias caché y virtual).
    - Por último, el efecto del compilador sobre el código ejecutable no será inocuo a la hora de medir el rendimiento, ya que las optimizaciones de este afectarán a la eficiencia del código.
        - j=log(5.0); por for(i=1;i<=5000;i++) j=log(5.0);
            - Si bien esta optimización hace el código del programa más eficiente, no favorece a la repetibilidad.

### *EJEMPLO DE MEDIDA DE RENDIMIENTO*




Sistema real (máquina Linux) y carga de coma flotante modelada

*EJEMPLO DE RENDIMIENTOS SIMILARES*

| Programa | A | B | Diferencias (A-B) |
|---|---|---|---|
| P1 | 5.4 | 19.1 | -13.7 |
| P2 | 16.6 | 3.5 | 13.1 |
| P3 | 0.6 | 3.4 | -2.8 |
| P4 | 1.4 | 2.5 | -1.1 |
| P5 | 0.6 | 3.6 | -3.0 |
| P6 | 7.3 | 1.7 | 5.6 |

Aunque a simple vista parezca que los rendimientos en los dos sistemas son diferentes, es necesario aplicar fórmulas estadísticas para poder corroborar si son realmente significativas las diferencias. Para ello será necesario calcular el intervalo de confianza entre las dos medidas:

$$\bar{x} \pm t_{1-\frac{\alpha}{2},n-1} \frac{s}{\sqrt{n}}$$

$\bar{x}$, media aritmética
s, desviación típica
t, distribución t Student
$\alpha$, nivel de significación
n-1, grados de libertad de la distribución

Si el nivel de confianza es del 95% ($\alpha = 0.05$)

- ➔ Si incluye el cero, no hay diferencias significativas
- ➔ Si no incluye el cero, las máquinas tienen rendimientos significativamente diferentes.

*CÁLCULO DE RELEVANCIA ENTRE LAS DIFERENCIAS DE LOS RENDIMIENTOS*

En este caso n, el número de diferencias, es 6.

**Cálculo de la media aritmética:**

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{6}(-13.7 + 13.1 + \ldots + 5.6) = -0.32$$

**Cálculo de la desviación típica:**

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} = \sqrt{\frac{1}{5}((-13.7 + 0.32)^2 + \ldots + (5.6 + 0.32))} = 9.03$$

Y por último necesitaremos el valor de la distribución t para un valor de confianza del 95%

$$t_{1-\frac{\alpha}{2},n-1} = t_{1-\frac{0.05}{2},6-1} = t_{0.975,5} = 2.57$$

**(consultar en tablas)**

Con todos los valores calculados, ahora sí que podemos calcular el intervalo de confianza con la fórmula anteriormente descrita, es decir:

$$\overline{x} \pm t_{1-\frac{\alpha}{2},n-1} \frac{s}{\sqrt{n}} = -0.32 \pm 2.57 * \frac{9.03}{6} = -0.32 \pm 9.47$$

Una vez lista la fórmula, tan solo queda sumar y restar 9.47 a -0.32, dándonos a entender que tenemos un intervalo de confianza situado entre los valores [-9.79, 9.15].

Dado que el intervalo de confianza sí que incluye el cero, podemos afirmar con una confianza del 95% que las diferencias **no son estadísticamente significativas**.

## ÍNDICES CLÁSICOS DE RENDIMIENTO

*CPI*

- ➔ Valor que interesa minimizar.
- ➔ El valor mínimo teórico es 1.
- ➔ Depende de la organización y la arquitectura.
- ➔ Ignora el tiempo imprevisible que hace falta para sincronizar procesador y memoria caché (ciclos de espera, fallos...).

$$CPI = \frac{Ciclos\ de\ CPU\ usados}{Instrucciones\ ejecutadas} = \frac{T_{ejecución} * Frecuencia\ de\ re}{Instrucciones\ ejecutada}$$

Cabe recordar, además, que la frecuencia de reloj se puede calcular también como $\frac{1}{T_{ciclo}}$

*EJEMPLO DE RELACIÓN ENTRE TIEMPO Y FRECUENCIA*

¿Cuál es la duración del tiempo de ciclo de un reloj a 500 MHz?

$$Ciclo\ de\ reloj = \frac{1}{500\ MHz} = \frac{1}{5*10^6 Hz} = 2 * 10^{-9} s = 2ns$$

*FORMA ALTERNATIVA DE CALCULAR EL CPI DE UN PROGRAMA*

Cada instrucción necesita un determinado número de ciclos, y por tanto, el valor del CPI depende de las instrucciones ejecutadas por el programa:

| | | |
|---|---|---|
| Instrucciones de escritura (store) | 12% | 2 ciclos |
| Instrucciones de lectura (load) | 21% | 2 ciclos |
| Instrucciones de la ALU | 43% | 1 ciclo |
| Instrucciones de salto | 24% | 2 ciclos |

Sabiendo esto, podemos calcular el CPI realizando el siguiente cálculo

$$CPI = \sum_{i=1}^{n} nºciclos_i * f_i$$

De este modo, podemos calcular que el CPI del programa mostrado en la tabla superior es de 1.57.

Por último, el tiempo de ejecución de un programa se puede calcular como:

$$Tiempo\ de\ ejecución = \frac{Instrucciones\ ejecutadas * CPI}{Frecuencia\ de\ reloj}$$

*MIPS*

- ➔ Depende del juego de instrucciones y los MIPS medidos varian entre programas en el mismo computador.
- ➔ Se diferencian entre nativos, que son MIPS teóricos, y relativos, que son MIPS referidos a una máquina de referencia
- ➔

El cálculo de los MIPS (nativos) es el siguiente:

$$MIPS_N = \frac{Instrucciones\ ejecutadas}{Tiempo\ de\ ejecución * 10^6} = \frac{Frecuencia\ de\ reloj\ de\ CPU}{CPI * 10^6}$$

Del mismo modo, los MIPS (relativos) se calcularán de la siguiente manera:

$$MIPS_R = \left(\frac{Tiempo\ de\ referencia}{Tiempo\ de\ ejecución}\right) * MIPS_{referencia}$$

*MFLOPS*

- ➔ Mide el rendimiento del procesador en operaciones en coma flotante en entornos de grandes computadores.
- ➔ Basados en operaciones y no en instrucciones
- ➔ El juego de instrucciones en coma flotante varía de entre una arquitectura a otra, y dentro de una misma arquitectura, cada instrucción tiene un tiempo distinto que puede variar según los operandos

$$MFLOPS_{Nat} = \frac{Operaciones\ de\ coma\ flotante\ ejecutadas}{Tiempo\ de\ ejecución * 10^6}$$

*MFLOPS NORMALIZADOS*

- ➔ Consideran la complejidad de las operaciones en coma flotante
  - ◆ Suma, resta, multiplicación, comparación, negación: poco coste
  - ◆ División, raíz cuadrada: coste medio.
  - ◆ Trigonometría: coste alto.
- ➔ Ejemplo de normalización de operaciones en coma flotante:
  - ◆ ADD, SUB, COMPARE, MULT = 1 operación normalizada
  - ◆ DIVIDE, SQRT = 4 operaciones normalizadas
  - ◆ EXP, SIN, ATAN, ... = 8 operaciones normalizadas

*CÁLCULO DE LOS MFLOPS DE UN PROGRAMA*

El programa Spice tarda 94 segundos en ejecutarse. Contiene 109970178 operaciones en coma flotante, de las cuales 15682333 son divisiones, el resto tiene una complejidad similar a la de la suma.

$$MFLOPS_{Nat} = \frac{109970178}{94 * 10^6} = 1.2$$

$$MFLOPS_{Nor} = \frac{94287845 + 15682333 * 4}{94 * 10^6} = 1.7$$

## PRUEBAS DE RENDIMIENTO

Para realizar las pruebas de rendimiento se utilizan un conjunto de programas **idéntico** a la carga real teniendo en cuenta que

- ➔ Se usan los mismos programas y de la misma manera
- ➔ Se sabe la frecuencia de uso
- ➔ Análisis WAW (Workload Analysis with Weights)
- ➔ Interesa el tiempo total de ejecución.

Por otra parte, se pueden realizar también con un conjunto de programas **similar** a la carga real teniendo en cuenta que:

- ➔ Se usan programas parecidos a los de la carga real.
- ➔ No se sabe la frecuencia de uso.
- ➔ Análisis SERPOP (Sample Estimation of Relative Performance of Programs)
- ➔ Interesa la aceleración relativa (speedup)

*REPRESENTACIÓN DEL RENDIMIENTO*

El rendimiento es una variable multidimensional, para representarla sería necesario utilizar múltiples índices, pero comparar el rendimiento entre dos sistemas diferentes es más sencillo si solo se utiliza un valor numérico (a maximizar o minimizar). De ese modo, se decide concentrar todas las variables en una sola utilizando la mejor variable que represente el rendimiento y asegurándose que es válida. El método habitual para hacer esto, es el de la síntesis, es decir: **el uso de medias.**

*MEDIA ARITMÉTICA*

| **Media aritmética para n valores con el mismo peso** | **Media aritmética ponderada para n valores con pesos distintos** |
|---|---|
| $x_a = \frac{1}{n}\sum_{i=1}^{n} x_i$ | $x_a = \frac{1}{n}\sum_{i=1}^{n} x_i * w_i$, con $\sum_{i=1}^{n} w_i = 1$ |
| No se debe utilizar la media aritmética con valores normalizados. Si se quiere normalizar, se debe normalizar solo el resultado final, no cada $x_i$ | |

*MEDIA ARMÓNICA*

| **Media aritmética para n valores con el mismo peso** | **Media aritmética ponderada para n valores con pesos distintos** |
|---|---|
| $x_h = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$ | $x_h = \frac{n}{\sum_{i=1}^{n} \frac{w_i}{x_i}}$,con $\sum_{i=1}^{n} w_i = 1$ |
| Se suele utilizar para índices distintos del tiempo de respuesta (WAW) | |