

# RAID

## Introducción

- Problema de rendimiento: los procesadores mejoran la velocidad un 50% al año pero los discos un 10%
- Solución: distribuir entre un array de discos los datos por tiras (***stripes***) de tal forma que se puedan hacer lecturas/escrituras simultáneas
- Nuevo problema: los arrays así planteados son muy sensibles a fallos. Si falla un disco, falla todo el array
- Solución: Utilizar **redundancia**. Resultado: RAID (Redundant Array of Inexpensive/Independent Disks)
- **Sobrecarga**:  $n^{\circ}$  bytes redundantes /  $n^{\circ}$  de bytes de datos

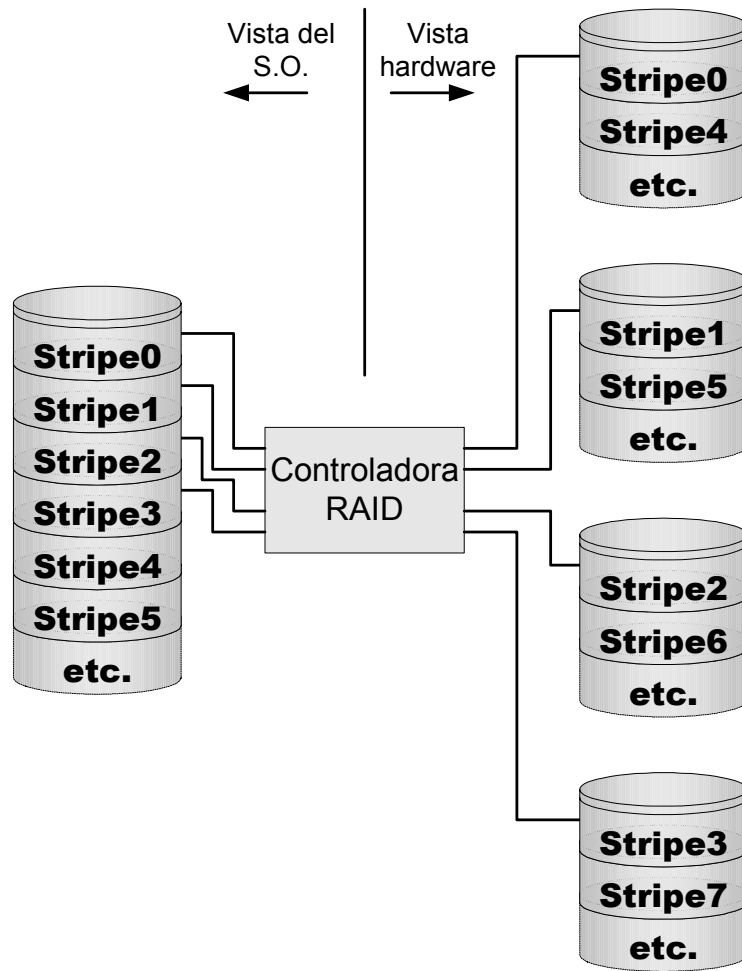
# RAID

## Striping

- Objetivo: distribuir los datos en varios discos de forma transparente para que parezca un único disco rápido
- Tipos:
  - Grado fino: cada unidad de datos entrelazada es pequeña.  
Ventajas: todas las peticiones de I/O usan todos los discos y tienen altas tasas de transferencia (MB/s)  
Desventajas: No puede haber peticiones simultáneas y se pierde mucho tiempo posicionando
  - Grado grueso: cada unidad de datos entrelazada es grande  
Ventajas: las peticiones de I/O pequeñas usan pocos discos y puede haber varias peticiones a la vez  
Desventajas: las peticiones de I/O pequeñas tienen tasas de transferencia menores

# RAID

## RAID 0: No redundante



- No es redundante. Sólo usa *striping*
- Es el más barato (sobrecarga 0 porque no se utiliza espacio para información redundante)

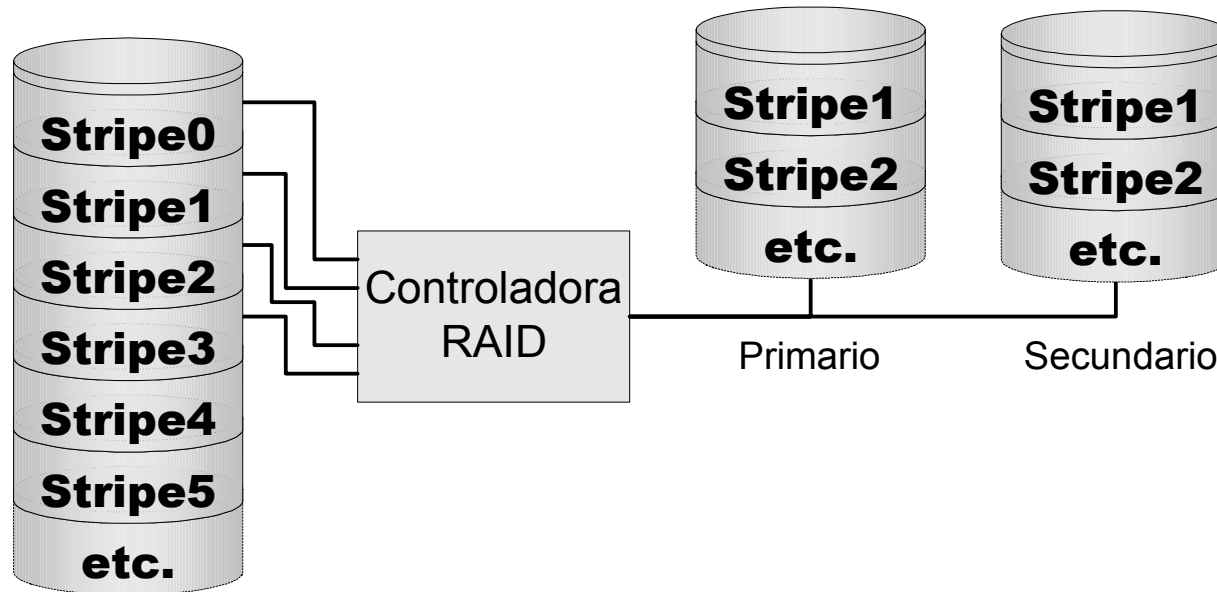
$$MTTF_{array} = \frac{MTTF_{disco}}{N}$$

Con  $N=n^{\circ}$  de discos  
y todos los discos iguales

# RAID

## RAID 1: En espejo (*mirror*)

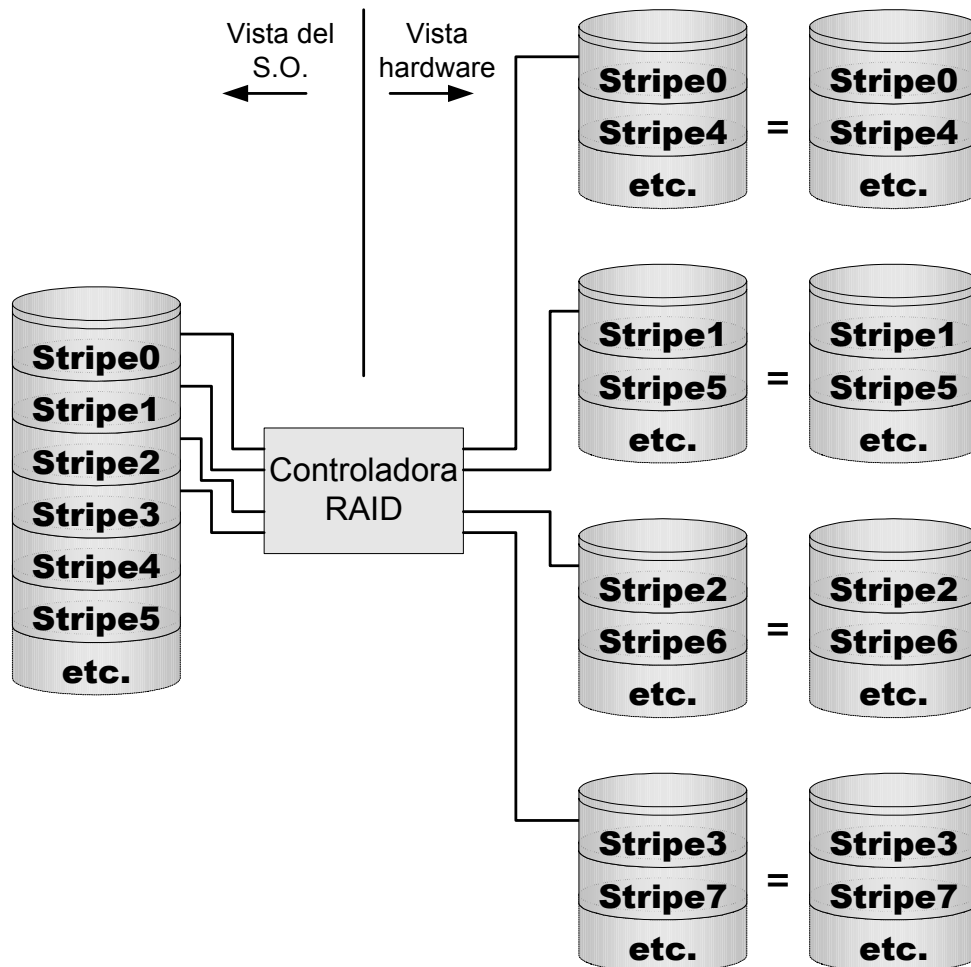
- Por cada disco de datos, uno de copia



- Máximo rendimiento de lectura
- Máxima sobrecarga (100%)

# RAID

## RAID 1 extendido a N discos



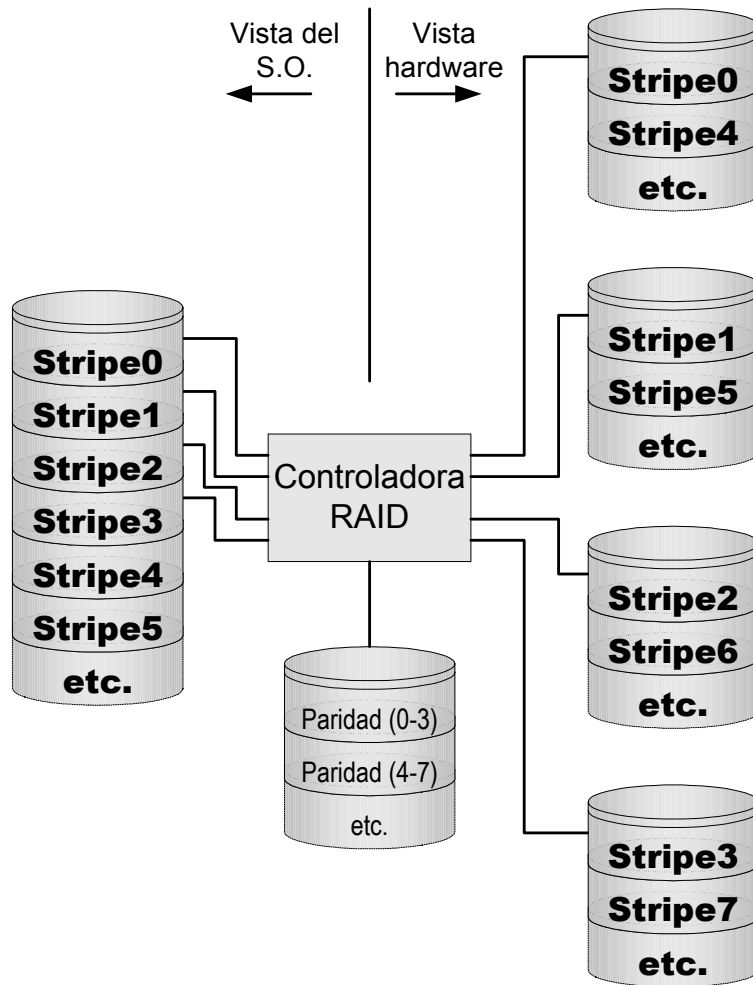
Se hace un RAID 0 y se duplica cada uno de sus discos con un RAID 1

$$MTTF_{array} = \frac{MTTF_{disco}^2}{2N * MTTR_{disco}}$$

Con  $N=n^0$  de discos de datos (en la fig., 4)

# RAID

## RAID 3 y RAID 4



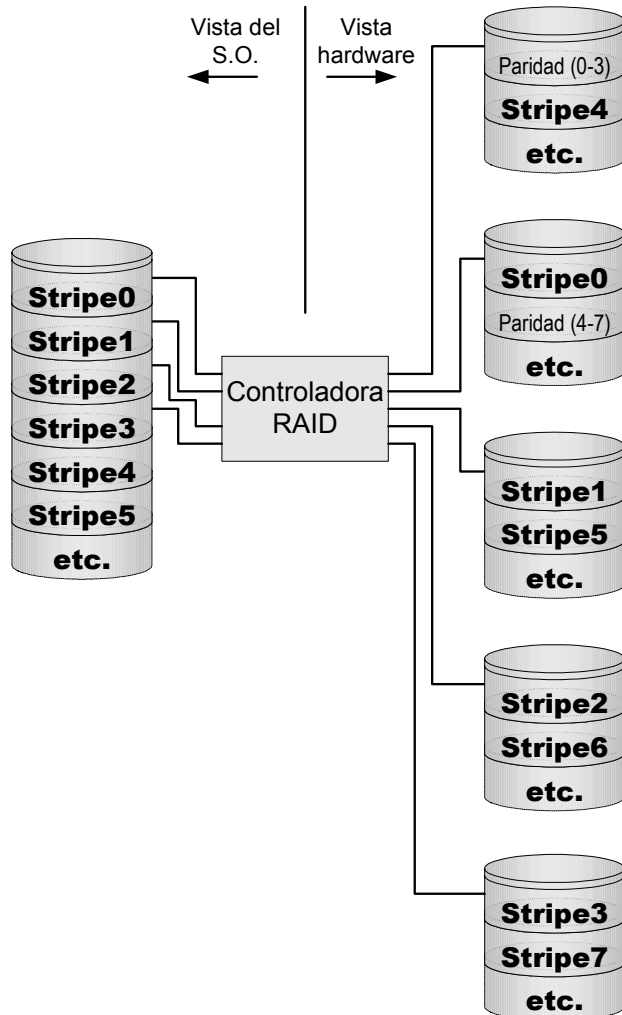
- Utilizar N discos de datos y 1 disco de paridad con la paridad de las tiras
- RAID 3: tamaño tira = 1 bit
- RAID 4: tamaño tira = 1 bloque
- Problema: el disco de paridad se convierte en cuello de botella

$$MTTF_{array} = \frac{MTTF_{disco}^2}{N * (N + 1) * MTTR_{disco}}$$

Con  $N=n^0$  de discos de datos  
(en la fig., 4)

# RAID

## RAID 5: Paridad entrelazada a nivel de bloque y distribuida



Como el RAID 4 pero en lugar de tener todos los bloques de paridad en un disco, se distribuyen entre todos los discos

$$MTTF_{array} = \frac{MTTF_{disco}^2}{N * (N + 1) * MTTR_{disco}}$$

Con  $N=n^0$  de discos de datos  
(en la fig., 4)

# RAID

## Otros problemas

- En las fórmulas anteriores se supuso que los fallos de los discos eran independientes. En la realidad puede no ser así: discos del mismo fabricante, misma remesa tienden a fallar a la vez
- Cuando falla un disco, en algunas configuraciones de RAID (sobre todo en RAID-5) baja mucho el rendimiento. Solución: tener un *hot-spare*, es decir, un disco preparado para que cuando un disco falle, otro lo sustituya inmediatamente
- No se tarda lo mismo en todos los tipos de RAID para recomponer el array cuando hay un fallo



# RAID

## Implementaciones

- En la práctica se usan:
  - RAID 0 y 3: Edición de vídeo, edición de imágenes, computación científica, aplicaciones que requieran mucha productividad
  - RAID 1: Sistemas transaccionales financieros, aplicaciones que requieran alta fiabilidad
  - RAID 5: Servidores (de ficheros, de aplicaciones, de bases de datos, web...)
- Las implementaciones pueden ser hardware o software. Sólo algunos niveles (0 y 1) se hacen software y, además, tienen peor rendimiento