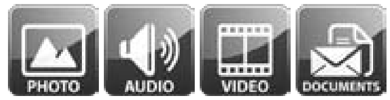
RAID 5

"Stripe w/ Redundancy"

Speed, Large Storage, & Protection



- 3 or more drives
- RAID array shows up as a single volume on the desktop.

Perfect For

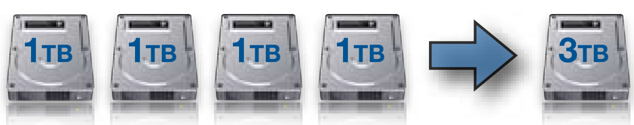
- Anyone wanting the speed and large capacity of striping, but with the protection of mirroring.
    - Store music, photos, videos
    - Anything that takes time to create
    - Anything that is difficult to replace
    - Anyone needing faster data rates

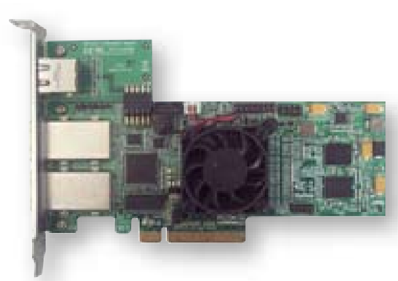
Speed & Protection

- Data striped for speed like a RAID 0, but a duplication or *parity* is built in to protect your data from a single drive failure.
- This results in fast performance comparable to a RAID 0, but with the added benefit of protection.

Optimized Storage Space

- The storage space resulting from a RAID 5 array is the sum of all the drives put together minus one drive worth of space.





RAID 5 — *How the Data is Distributed*

- RAID 5 incorporates striping of data just like in a RAID 0 array, however, in a RAID 5 there are redundant pieces of the data that are also distributed across the drives and are referred to as ***parity***. Having the parity blocks staggered across each drive allows any single drive in the RAID 5 array to fail without any data loss.

- In the illustration below there are 4 drives set up as a RAID 5. Each colored square is a “block” of data. All blocks are the same size.
    - The first data block of the mp3 file is represented by *a* the second is *b* and the third is *c*
        - Each of those three data blocks have a certain number of bits that are 1’s & 0’s.
        - Those values are then summed up as a value on the parity block represented by *a b c* on ***Drive 4.***
        - Like an algebraic equation, you can solve the equation when you have a missing piece, in this case a missing piece would occur when you have a failed drive.

Parity at the bit level

In a data block if the sums of 1’s & 0’s...
Equals an even number
==> Block has a value of 0

Equals an odd number
==> Block has a value of 1

Sum of the values makes up the value in the parity block

	a	*b*	*c*	*a b c*
1st bit	0 →	1 →	1 →	**0**
2nd bit	1	0	0	**1**
3rd bit	0	0	0	**0**
4th bit	1	1	1	**1**

Rebuilding

If any drive fails, you can solve for the missing value.

Before the failure, we know:
0+1+1=0

If drive 2 fails, we have:
0+x+1=0

We know we need the sum of *1+x+0* to equal an even number, so ***x=1***