The end of RAID 5

Jörg Möllenkamp
Sun Microsystems

This is a thought game ...

It´s based on some articles written by Robin Harris of storagemojo.com

RAID 5 will stop working in 2009

Out of problems inherent
to rotating rust (aka harddiscs)

At the end it´s yet another reason to want ZFS.

Okay, one trend in the business ist the usage of cheap SATA disks

It´s really cheaper,
but has it´s disadvantages ...

From the specification of a Seagate harddisk

Specifications	160GB[1]
Interface Options	
Model Number	ST3160811AS[2] ST3160211AS[2]
Performance	
Cache Options (Mbytes)	2, 8
Available Transfer Rates (Mbytes/sec)	100, 300
Average Latency (msec)	4.16
Spindle Speed (RPM)	7200
Configuration/Organization	
Bytes per Sector	512
Reliability/Data Integrity	
Contact Start-Stops	50,000
Nonrecoverable Read Errors per Bits Read	1 per 10^{14}
Annualized Failure Rate (AFR) (%)	0.34
Limited Warranty (years)	5

It doesn´t look different at other vendors

Statisticially every 10^{14} bits you will read a corrupt bit fron the harddisk without detection of the error correction of the harddisk

Such errors are inevitable ...

By the way ...

To get todays huge data density at high speeds you needa special mechanism to read the data from the rotating rust

PRML

Partial Response Maximum Likelihood

„maximum likelihood“ ....
... hearth warming isn´t it ? ;)

The harddisk electronics don´t
measure the exact signal levels of the data.
Based on **statistical** methods the **most probable**
bit pattern is generated from the analog signal.

This works extremly well ...
but not everytime

Back to the
10^{14} bit

10^{14} bit

13

$1.25 * 10^{13}$ Byte

13

$1.25 * 10^{13}$ Byte

12.207.031.250 KiB

11.920.928 MiB

11.641 GiB

11,3 TiB

Let´s simplify things by assuming that this are 12 Terabyte ...

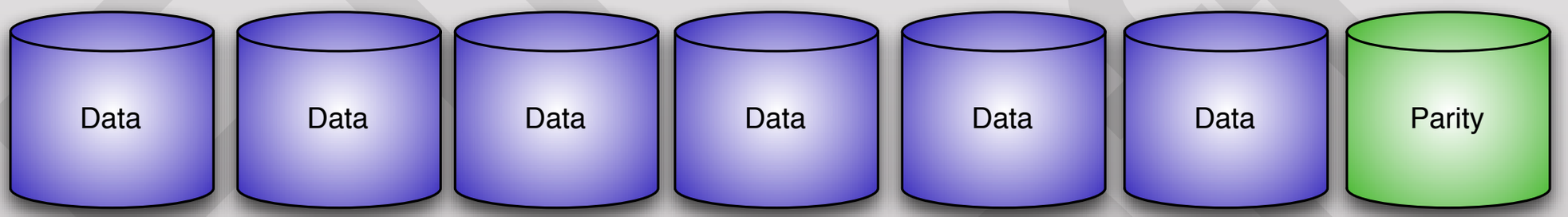
Keep this 12 Terabyte in mind!

Assumption

We will see 2 TB and bigger disk in 2009

Okay, let´s start with some basics of RAID5

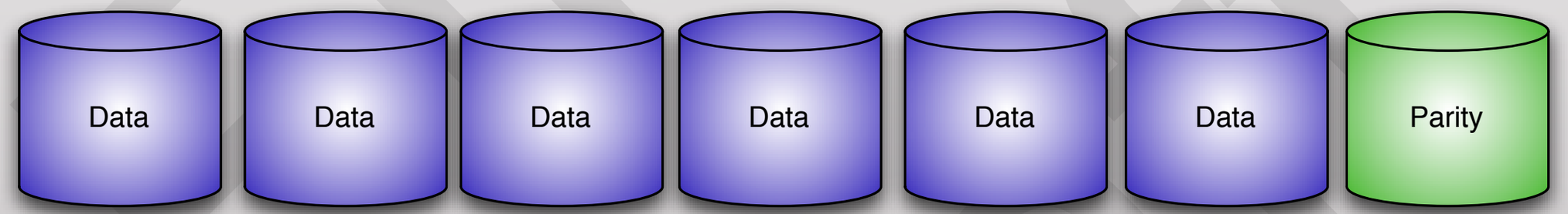
Okay, let´s look at a single stripe with 6 disks and 1 disk for the parity.



Side note

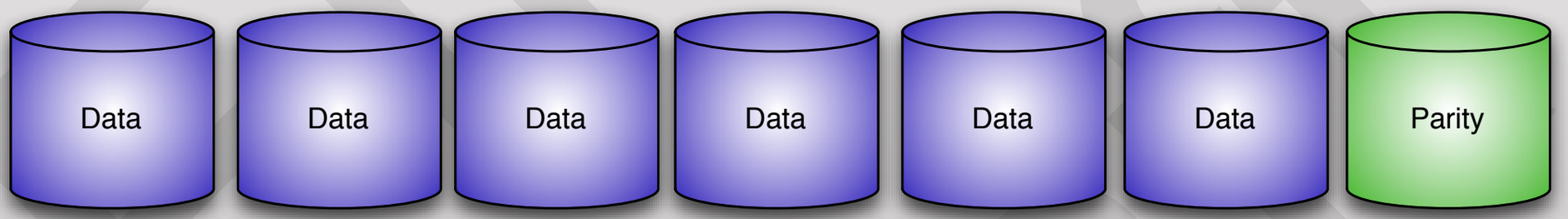
I know, that RAID5 has no dedicated parity disk, but this is much easier to understand and much easier to put into to some figures.

As long all disks are well and okay, you doesn´t have this problem, and you would be able to use the parity for scrubbing

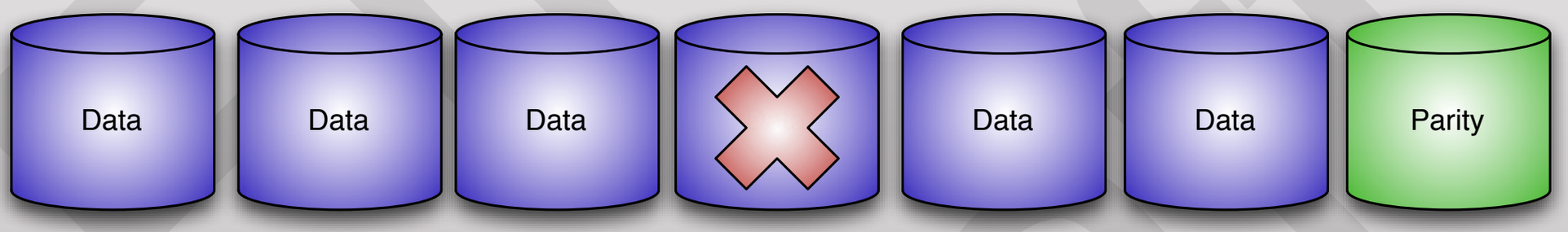


(although i would prefer a stronger information redundancy by the one introduced by XOR)

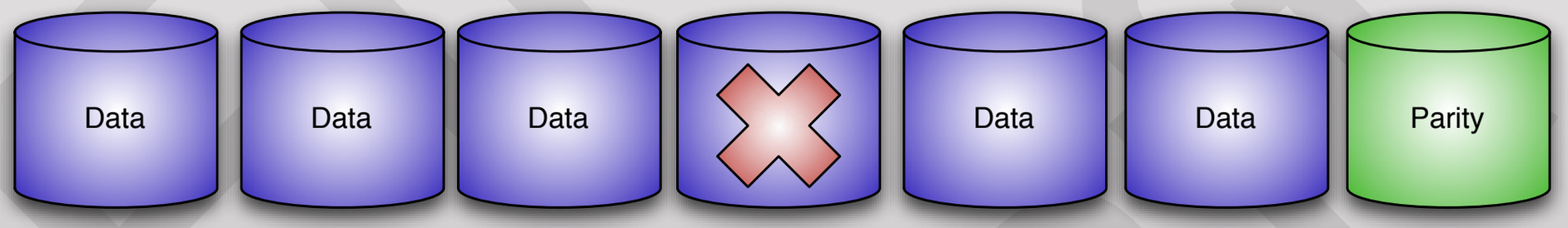
Okay, the rationale behind RAID 5 is the deliberate generation of information redundancy to survive the losses of informations by a disk failure



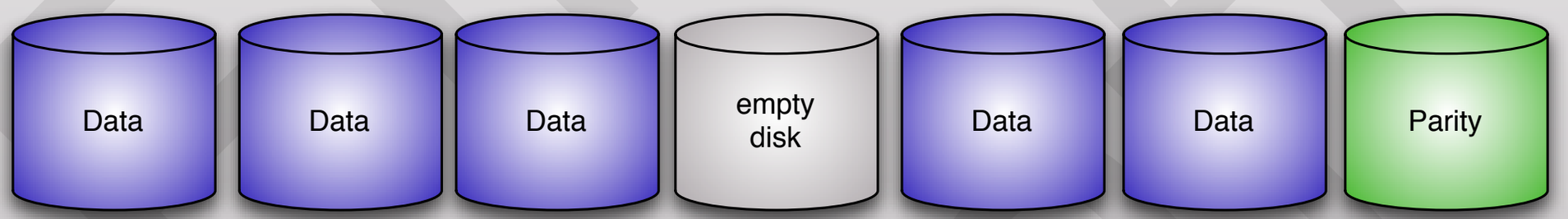
Let´s assume a disk failure ...



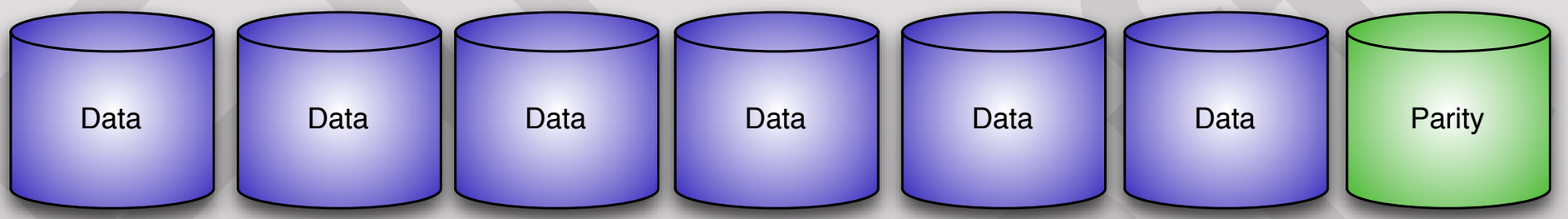
You lost the data redundancy ...



The failed disks is subsituted by an intact, but empty disk.

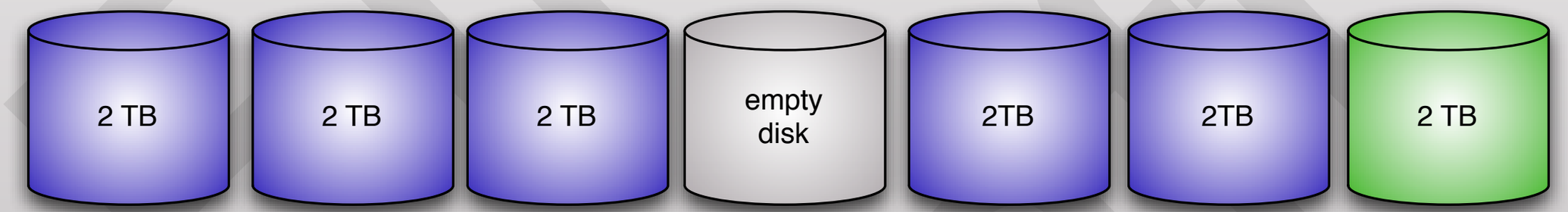


You get your data back with an algorithm to calculate the missing data out of the parity and the surviving data

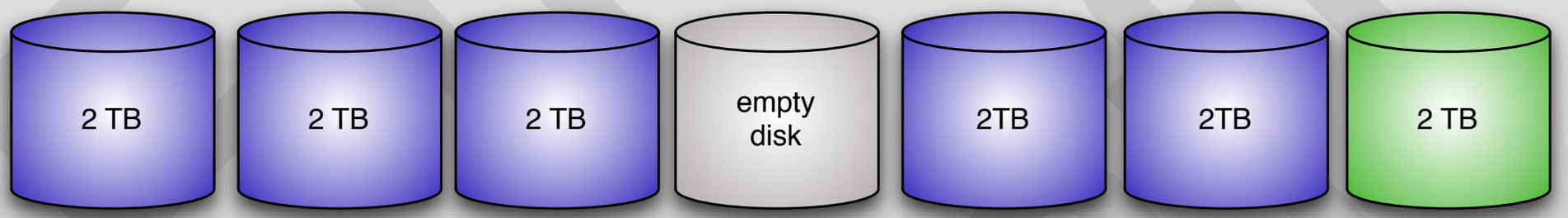


And that´s the problem ...

You have to read all the data on all surviving disks to recover the data.

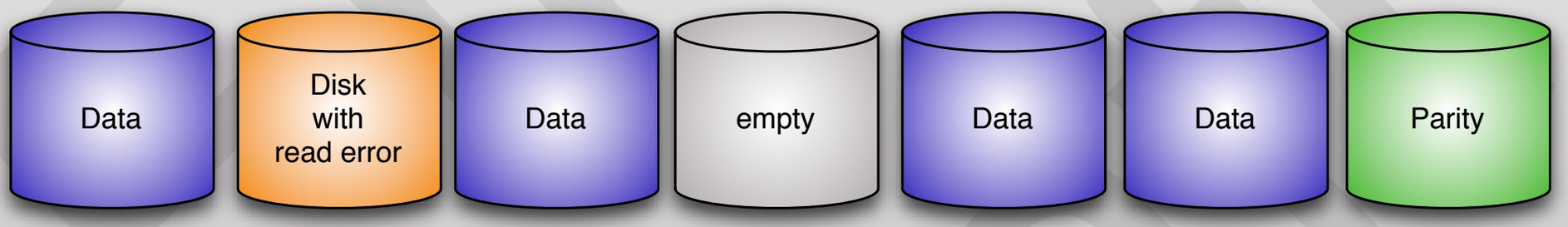


Okay, that´s 6 * 2 TB = 12 TB

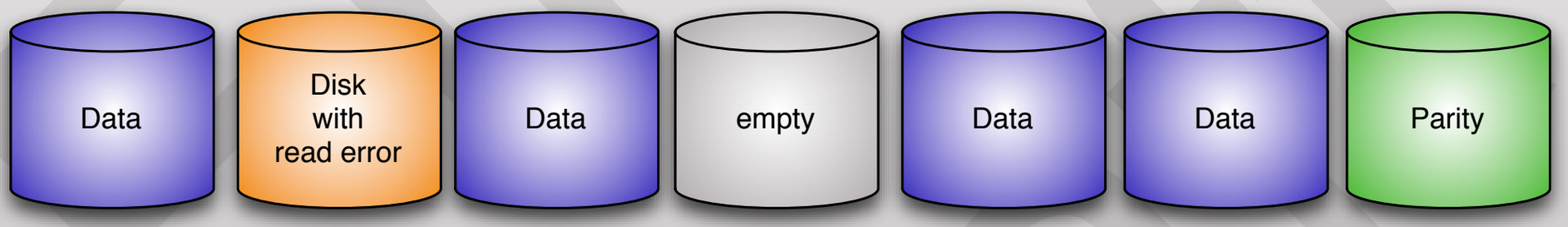


Do you remember
Every 12 TB ....

While resyncing your RAID5 you´ve read at least one wrong bit from your disks giving you wrong data.

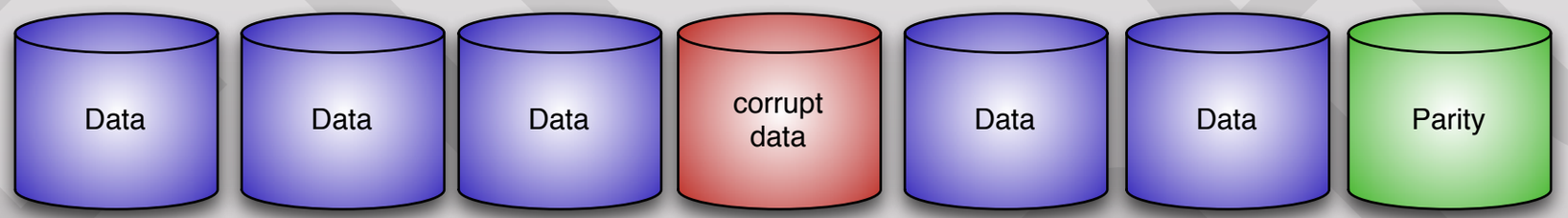


So the reconstruction algorithm works with incorrect data.



The RAID-Controller can´t detect it.
As the controller works on blocklevel
without knowledge about the data,
it can´t tell correct from incorrect data,
empty blocks from filled blocks

With a bit error while resyncing:
The reconstructed data on the new disk is incorrect.



And as a nice addon:
A time bomb

The data is still correct on the rotating rust.

Now let´s assume another drive failure.

The RAID5 gets reconstructed again.

The disk with the bit error in the last resync may deliver the correct bits now.

But the actual data of the disk failed before was calculated on the still present and unchanged parity and the data with the bit error.

At the next resync the same you need the same bit error to calculate at least the data of this failed disk correctly.

So even without an
bit error you create another
disk with incorrect data.

The overarching challenge:
To assure, that you´ve read
the surviving information correctly
or deliver means detect
bit errors.

Especially while resyncing you need to validate the correctness of the data you´ve read from the rotating rust.

In the moment of a failure,
the parity isn´t useful for validating
the correctness

You´ve lost your data redundancy
by the failure

So how detect such a problem ?
Majority vote reads ?

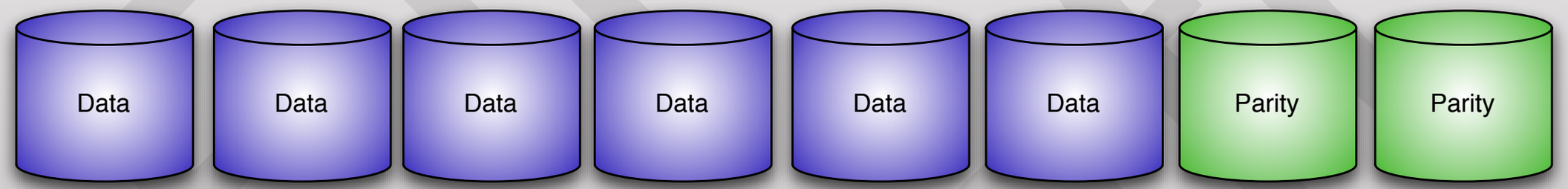
And what´s with persistent problems?
When all reads deliver wrong data ...

You need more data redundancy

And now ... ?

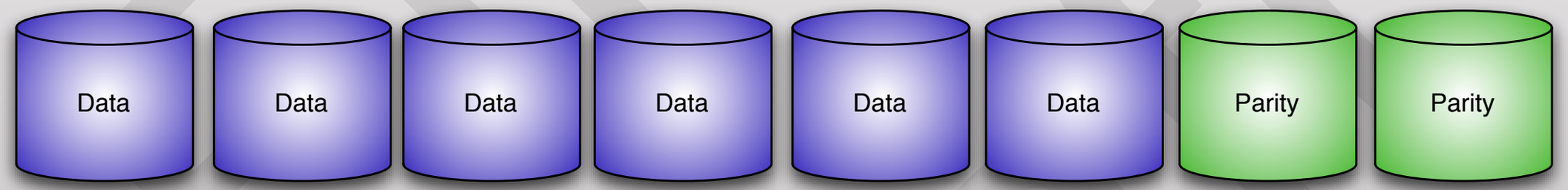
RAID6

You can use the additional data redundancy
to check the validity of your reconstruction

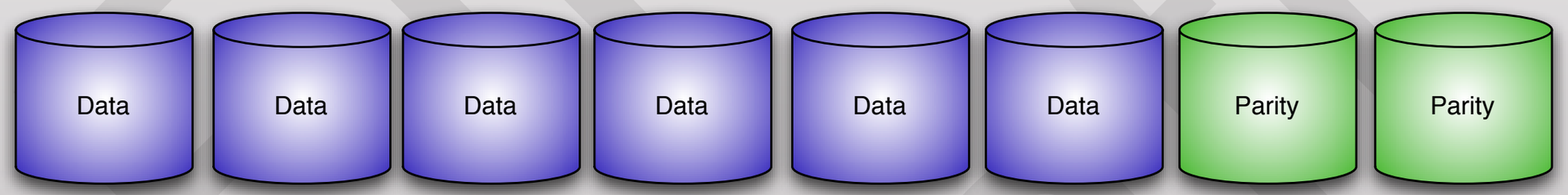


but ...

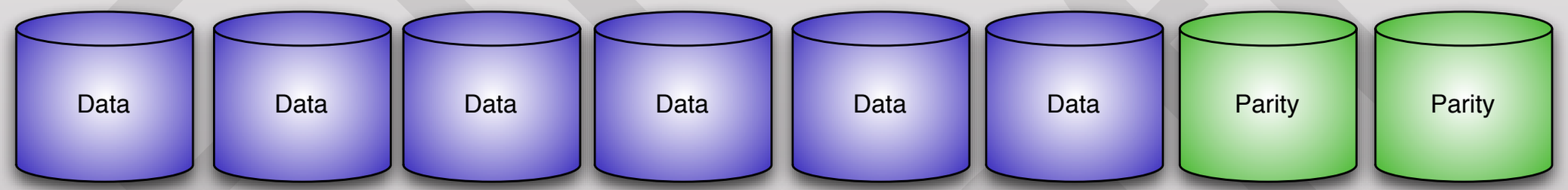
more overhead



And the overarching problems of
bit errors isn´t solved by this problem ...



The parity information is only used for resyncing (and scrubbing), but not at every read ...



Alternative: Checksums

like implemented in ZFS

With strong checksums
i can validate the correctness of
the data i´ve read

The checksum is an
information redundancy, too ...

I can use it to check
the correctness of the reconstruction ...

ZFS uses this for a neat trick
combinatorical resync

When the
checksum on disk
and the
checksum calculated from the read data
doesn´t match
the system fetches the parity.

With checksum and parity,
you can determine the block with the incorrect data
and correct it ...

But there are still good reasons for RAID6...

A double failure of two disks
in a RAIDZ sends
you to the tapes as well ..

There is a reason for double parity RAIDZ
(obviously called RAIDZ2)

At the end this poses an interesting question:
Can you afford not to use checksums
for exisiting and upcoming large disks?