Generic RAID Reassembly using Block-Level Entropy

Christian Zoubek, Sabine Seufert, Andreas Dewald

30.04.2016

Outline

Outline

What is RAID

Redundant Array of Independent (originally 'Inexpensive') Disks

- Several physical disks combined
  - Abstraction layer between hard disks and file system
  - One logical unit
- Depending on RAID it is able to
  - recover Data lost by hardware failure
  - speed up Data transfer
  - heavily increase capacity

Why recovery

Most server environments use RAID

Seizure does not guarantee knowledge about RAID parameters

- Undocumented RAID parameters
- Administrator not willing to cooperate
- Broken RAID controller

$\Rightarrow$ Some or all parameters missing

Missing parameters may lead to data loss

RAID parameters

RAID defined by several parameters

- RAID type/level (RAID 0, RAID 1, etc.)
- Stripe size
  - Size of each contiguous block
  - Common: 1KB - 1MB
- Disk count
- Stripemap
  - Order of disks
  - How data is distributed over disks

In detail

RAID 1

- All disks save the exact same data
- Redundancy by mirroring

$\rightarrow$ Recovery straightforward

RAID 0

- Data distributed over all disks
- No redundancy

$\rightarrow$ One broken disk equals to loss of all data

RAID 5 in detail

RAID 5

- Redundancy through parity
- Data and parity distribution over all disks

$\rightarrow$ Mix of failure safety and better performance

- Literature: Different Setups possible

RAID 5

Properties of common RAID 5 setups

- Parity distribution (describes shift of parity block after each row)
  - Left-sided (Parity block shifted from last disk to first)
  - Right-sided (Parity block shifted from first disk to last)
- Data distribution (describes location of first block of each row)
  - symmetric (First data block right to parity block)
  - asymmetric (First data block at first disk)

RAID 5 - examples

RAID 5 using 4 disks

0	1	2	P
3	4	P	5
6	P	7	8
P	9	10	11

left asymmetric

P	0	1	2
5	P	3	4
7	8	P	6
9	10	11	P

right symmetric

Outline

Algorithm

Distinguish between RAID 0/1/5 by utilizing their characteristics

- RAID 1 only has mirrored blocks
- RAID 5 uses parity block in each row

Declare counters for occurences of

- Mirrored blocks
- Parity blocks
- None of both

Comparison of counters lead to knowledge of RAID level

Interpretation

Possibility to detect missing RAID 5 disk

- Assumption: Some blocks on missing disk are empty
- Mirrored or parity blocks may be found (Y xor $0 = Y$)

	RAID-0	RAID-1	RAID-5c	RAID-5i
mirrored	low	high	low	mean
parity	low	low	high	mean
unassigned	high	low	low	high

Algorithm

Find possible sizes using entropy

- Calculate entropy of 512-byte blocks
  - Count encounters of each possible byte value
  - Probability distribution $\rightarrow H = -\sum_i p_i \times \log(p_i)$
- Find consecutive blocks with high entropy differences (Unusual within the same file)
- Validate finding by checking surroundings
- Mark edge as possible interesting address

Algorithm - continued

After finding some addresses of interest

- Calculate difference between two consecutive addresses
- Find best fitting stripe size
  - Start with greatest stripe size (we use 2MB)
  - Difference modulo stripe size
  - If zero, mark as possible stripe size

Example

1.75MB file over four disks, RAID 0

Address	Disk 0	Disk 1	Disk 2	Disk 3
...				
888273920	0	0	0	0
888274432	0	0	0	0
888274944	0	7.50199	7.56131	7.57583
888275456	0	7.53411	7.54758	7.54145
...				
888306176	0	7.46816	7.43265	7.48876
888306688	0	7.43318	7.59278	7.60496
888307200	6.14066	7.48741	7.58424	7.49408
888307712	7.64113	7.53735	7.59764	7.46034
...				
888732672	7.43689	7.55090	7.52364	7.54029
888733184	7.52416	7.54816	7.57045	7.53455
888733696	7.44034	7.54581	7.46290	0
888734208	7.47576	7.51771	7.57273	0
...				

Stripe: 888274944 - 888733696 (= 458752; Stripe: 64KB)

Disk order

Striped data blocks are written consecutively over the disks

- Empty blocks may indicate position within stripe
- Stripe with empty blocks and used blocks interesting

Algorithm

- Find begin/end of a file within a disk
    - Calculate entropy of blocks half the stripe size
    - Rising entropy: begin of a file
    - Falling entropy: end of a file

Disk order - Algorithm

Check other disks at same address

- All full with data: Discard
- One or more empty
  - If begin of a file; Empty blocks were written beforehand
  - else; empty blocks written after end of file

RAID 0 almost finished

- Only disk order to recover
- Rebuild order by resolving findings

RAID 5 uses parity block

- Disk order not that easy to tell (parity block)
- Derive a disk order for each row in stripe map

RAID 5 - extension

RAID 5 usually uses map with n rows (n = # disks)

- Find distribution of parity across disks
  - Fact: The more random data the higher the entropy
  - Assumption: Parity most often the most random block each row
  - → Derive parity map by comparing entropies of each row
- Find correct row to address: $\left(\frac{a}{s}\right) mod(n)$
  - a = address on disk
  - s = stripe size
  - n = number of disks

RAID 5 - Stripe map

Use parity map and row-wise disk order to set properties

- Find parity block of each row
- Check blocks written previous to parity block by the same disk

  - Always first block → right symmetric
  - Always last block → left symmetric
  - Ascending order → right asymmetric
  - Descending order → left asymmetric

Outline

Data set

Different RAID setups for data storage

- Low entropy data (text files)
- High entropy data (picture files)
- RAID 0 and RAID 5
- Varying stripe sizes: 16,64,256,1024 [KB]
- File systems: Ext4 and NTFS

Furthermore

- Six Ubuntu installations (3 $\times$ RAID 0, 3 $\times$ RAID 5)
- Several Software RAIDS (mdadm)

$\Rightarrow$ 38 RAIDs + Software RAIDs

Stripesize

Optimal threshold for entropy differences dependent on

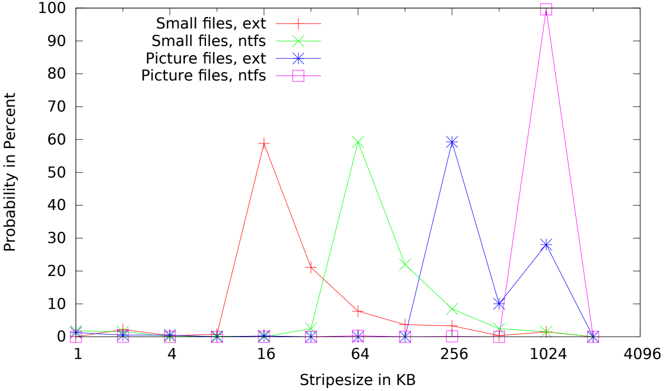
- File system
- Types of file
- Stripe size

Observations

- NTFS using picture files stable in almost every combination
- Large stripe sizes prefer large entropy differences
- Best fitting in all cases: 0.3 (lower bound) - 7.3 (upper bound)

Stripesize

Some results for different stripe sizes and data



Stripe map - Parity distribution

Using picture files

Disk 0	Disk 1	Disk 2	Disk 3
0	4958	0	0
0	0	5002	0
0	0	0	4911
4922	0	0	0

Different small files

Disk 0	Disk 1	Disk 2	Disk 3
485	480	497	3805
469	512	3808	478
499	3785	490	498
3800	518	442	510

Summary

Stripe size calculation

- fixed entropy threshold (0.3 and 7.3)
- worked in every case

Stripe map

- Parity distribution worked in every RAID 5 case
- Finding disk order worked in every case but one
  - RAID 0, small files, great stripe size
  - Only part of the disk order was recovered

Outline

Conclusion

Automated reassembly of RAID systems is possible, yet has its limits

- Will not work on encrypted disks
- Disk with only small files lack enough information
- Nested RAIDs?

Last slide

Thank you for your attention.
Questions?

Slides and OpenSource tool:
https://www1.cs.fau.de/content/forensic-raid-recovery