Redundant Arrays of Inexpensive Disks

Annalise Tarhan

March 11, 2021

1 Use the simulator to perform some basic RAID mapping tests. Run with different levels and see if you can figure out the mappings of a set of requests. For RAID-5, see if you can figure out the difference between left-symmetric and left-asymmetric layouts. Use some different random seeds to generate different problems than above.

RAID-0: maps to disk number (address %4)
RAID-1: same as RAID-0 but disks 1 and 2 are switched
RAID-4: maps to disk number (address %3)
RAID-5: maps to disk number (address %4)

Googling indicates that the difference between left-symmetric and left-asymmetric layouts deals with how data blocks are arranged around parity blocks. Asymmetric layouts always start at disk 0 and simply jump over the parity block, wrapping back around to disk 0 after the stripe is completed. Symmetric layouts, on the other hand, start the stripe after the parity block, then wrap back to disk 0 to finish the stripe, jumping to the next stripe when it reaches the parity block.

Figures below from http://www.accs.com/p_and_p/RAID/LinuxRAID.html

RAID-5 Segments
Left Asynchronous

Drive 0	Drive 1	Drive 2	Drive 3	Drive 4
0	1	2	3	Parity
4	5	6	Parity	7
8	9	Parity	10	11
12	Parity	13	14	15
Parity	16	17	18	19
20	21	22	23	Parity

RAID-5 Segments
Left Synchronous

Drive 0	Drive 1	Drive 2	Drive 3	Drive 4
0	1	2	3	Parity
5	6	7	Parity	4
10	11	Parity	8	9
15	Parity	12	13	14
Parity	16	17	18	19
20	21	22	23	Parity

2 Do the same as the first problem, but this time vary the chunk size. How does chunk size change the mappings?

Increasing the chunk size by a certain factor causes the block size to be divided by the same factor before the mapping is calculated.

3 Do the same as above, but use the -r flag to reverse the nature of each problem.

I'm not sure what there is to figure out, since the blockSize is given at the top and the block number is already printed out.

4 Now use the reverse flag but increase the size of each request. Try specifying sizes of 8k, 12k, and 16k, while varying the RAID level. What happens to the underlying I/O pattern when the size of the request increases? Make sure to try this with the sequential workload too; for what request sizes are RAID-4 and RAID-5 much more I/O efficient?

As the number of blocks requested increases, so does the number of disks accessed, up to the number of available disks. All of the accesses for a single request are at roughly the same offset. RAID-1 alternates accesses between the mirrored disks, RAID-4 skips the parity disk, and RAID-5 skips parity blocks when it encounters them. RAID-4 and RAID-5 really shine with large, sequential requests, since all disks can operate fully in parallel, almost as quickly as RAID-0.

5 Use the timing mode of the simulator to estimate the performance of 100 random reads to the RAID, while varying the RAID levels, using 4 disks.

RAID-0: 275-300
RAID-1: 275-300

RAID-4: 320-440
RAID-5: 275-300

6 Do the same as above, but increase the number of disks. How does the performance of each RAID level scale as the number of disks increases?

RAID-0: 140-160
RAID-1: 145-170
RAID-4: 150-200
RAID-5: 140-175

Doubling the number of disks roughly halved the time requests took.

7 Do the same as above, but use all writes instead of reads. How does the performance of each RAID level scale now? Can you do a rough estimate of the time it will take to complete the workload of 100 random writes?

4 Disks
RAID-0: 270-300
RAID-1: 500-550
RAID-4: 960-990
RAID-5: 510-560

8 Disks
RAID-0: 140-160
RAID-1: 260-300
RAID-4: 930-960
RAID-5: 290-350

RAID-0 and RAID-1 saw their times cut in half when the number of disks doubled, but RAID-4's hardly changed and RAID-5's only dropped by about a third.

8 Run the timing mode one last time, but this time with a sequential workload. How does the performance vary with RAID level, when doing reads versus writes? How about varying the size of each request? What size should you write to a RAID when using RAID-4 or RAID-5?

There was very little difference between time to read and time to write for sequential workloads for any RAID level. Doubling the size of each request increased the total time by about a third for RAID-0 and RAID-1, but only by a quarter for RAID-4 and RAID-5. Doubling the size of each request means twice as much data was written, but the total time didn't come anywhere close to doubling. So, when writing to a RAID, especially RAID-4 and RAID-5, it is more efficient to make requests large.