RAID5 versus RAID10

First let's get on the same page so we're all talking about apples.

What is RAID?

RAID originally stood for Redundant Arrays of Inexpensive Disk and was an idea proposed in the early days of computing when storage was relatively expensive. It encompasses several schemes for building large unified storage from two or more smaller drives and for adding redundancy to the storage to improve the safety of the data it holds. Today, in recognition of the fact that permanent storage is no longer expensive relative to the cost of the other hardware making up a computing system, most vendors and pundits have taken to substituting “Independent” for “Inexpensive” in the expansion of the RAID acronym.

The two most basic forms of RAID are RAID0 and RAID1. RAID0 refers to a technique called “striping” where data is spread across multiple drives to form a logical or virtual drive or array that is larger than a single drive. The zero in the name indicates that there is no redundancy in a RAID0 array. If any of the component drives should fail data loss is complete and unrecoverable.

The other early and basic RAID form is known as RAID1 which identifies two or more drives that “mirror” each other so that the identical data is written redundantly to all of the drives in the set. A two drive mirror set or “mirrored pair” is most common and predates all other RAID forms. Since all of the data written to the set is completely mirrored on at least one other drive the loss of a single drive does not cause any data loss.

Mirroring (RAID1) was expensive when drives were expensive so other RAID forms were described by the inventors of the concept. RAID2, RAID3, and RAID4 were parity schemes that are no longer in active use today, so we will skip over them and get straight to the most ubiquitous form of RAID common today, RAID5.

What is RAID5?

RAID5 combines the concepts of RAID0 with parity for data redundancy. RAID5 uses ONLY ONE parity drive per stripe. RAID5 stripes can be assembled from three or more drives. Many RAID5 arrays are assembled from five drives (4 data and 1 parity though it is not a single drive that is holding all of the parity as in RAID 3 & 4 but read on). If your drive counts are different adjust the calculations appropriately. If you have 10 drives of say 20GB each making 200GB RAID5 will use 20% for parity (assuming you set it up as two 5 drive arrays) so you will have 160GB of usable storage.

Now since RAID10 (detailed below), like mirroring (RAID1), uses 1 (or more) mirror drive for each primary drive you are using 50% for redundancy. To get the same 160GB of usable storage you will need 8 pairs or 16 – 20GB drives, which is why RAID5 is so popular among storage vendors and accountants. This intro is just to put things into perspective.

RAID5 is physically a stripe set like RAID0 but with failed drive recovery included. RAID5 reserves one disk block out of each stripe block for parity data. The parity block contains an error correction code which can correct most errors in the RAID5 block, in effect it is used in

combination with the remaining data blocks to recreate any single missing block, gone missing because a drive has failed. The innovation of RAID5 over RAID3 & RAID4 is that the parity is distributed on a round robin basis so that there can be independent reading of different blocks from the several drives. This is why RAID5 became more popular than RAID3 & RAID4 which must synchronously read the same block from all drives together. So, if Drive2 in a RAID5 array fails blocks 1,2,4,5,6 & 7 are data blocks on this drive and blocks 3 and 8 are parity blocks on this drive. That means that the parity on Drive5 will be used to recreate the data block from Disk2 if block 1 is requested before a new drive replaces Drive2 or during the rebuilding of the new Drive2 replacement. Likewise the parity on Drive1 will be used to repair block 2 and the parity on Drive3 will repair block4, etc. For block 2 all the data is safely on the remaining drives but during the rebuilding of Drive2's replacement a new parity block will be calculated from the block 2 data and will be written to Drive 2.

Now when a disk block is read from the array the RAID software/firmware calculates which RAID block contains the disk block, which drive the disk block is on and which drive contains the parity block for that RAID block and reads ONLY that one data drive. It returns the data block. If you later modify the data block it recalculates the parity by subtracting the old block and adding in the new version then in two separate operations it writes the data block followed by the new parity block. To do this it must first read the parity block from whichever drive contains the parity for that stripe block and reread the unmodified data for the updated block from the original drive. This four step read-read-write-write operation is known as the RAID5 write penalty since these two writes are sequential and synchronous the write system call cannot return until the reread and both writes complete, for safety, so writing to RAID5 is up to 50% slower than RAID0 for an array of the same capacity. (Some software RAID5's avoid the re-read by keeping an unmodified copy of the original block in memory. In some application spaces this can require very large memory caches on the array which is usually far more expensive than mechanical or SSD drives.)

Now what is RAID10:

RAID10 is one of the combinations of RAID1 (mirroring) and RAID0 (striping) which are possible the other being RAID01. There used to be confusion about what RAID01 or RAID10 meant and different RAID vendors defined them differently. Several years ago I proposed the following standard language which seems to have taken hold. When N mirrored pairs are striped together this is called RAID10 or RAID 1+0 because the mirroring (RAID1) is applied before striping (RAID0). The other option is to create two stripe sets and mirror them one to the other, this is known as RAID01 or RAID 0+1 (because the RAID0 is applied first). In either a RAID01 or RAID10 system each and every disk block is completely duplicated on its drive's mirror. Performance-wise both RAID01 and RAID10 are functionally equivalent. The difference comes in during recovery where RAID01 suffers from some of the same problems I will describe affecting RAID5 while RAID10 does not.

Now if a drive in the RAID5 array fails, is removed, or is shut off data is returned by reading the blocks from the remaining drives in the array and calculating the missing data using the parity, assuming the defunct drive is not the parity block drive for that RAID block. Note that it takes 4 physical reads to replace the missing disk block (for a 5 drive array) for four out of every five disk blocks leading to a 64% performance degradation until the problem is discovered and a new drive can be mapped in to begin recovery. Performance is degraded further during recovery because all drives are being actively accessed in order to rebuild the replacement drive which will cause head contention (see below).

If a drive in the RAID10 array fails data is returned from its mirror drive in a single read with only minor (6.25% on average for a 4 pair array as a whole) performance reduction when two non-contiguous blocks are needed from the damaged pair (since the two blocks cannot be read in parallel from both drives) and none otherwise.

One begins to get an inkling of what is going on and why I dislike RAID5, but, as they say on late night infomercials, wait, there's more!

Yikes stripes!

When stripe sets like those that make up RAID0, RAID5, and RAID10 are formed there is a concept known as as the block size, stripe size, or stripe block. This is the amount of data that resides as an atomic unit of IO on each single drive. Files larger than the stripe block size are spread across multiple drives with sequential blocks of data the size of a stripe block written round robin across the drives making up the stripe set. Typically this stripe block size is significantly larger than the physical drives' IO block which is typically 512 bytes. In order to optimize file IO most stripe sets are created with block sizes likely to permit a large file to reside in a single stripe block, ie on a single drive. Popular stripe block sizes range from 256KB up to 2GB. Most storage administrators and the vendors providing such arrays prefer larger stripe block sizes and the most common stripe block size that I have come across recently is 256MB.

For filesystems used for typical flat files, this is fine. Most flat files are written or rewritten whole and most flat files are also read into memory in their entirety by the applications that maintain them. Unfortunately databases do not work that way. Databases in general, and relational databases in particular, read and write much smaller units of storage. Database records are typically very small, many less than 1KB. However, most databases do not read and write individual records but rather gather several records into fixed size storage blocks typically known as "pages". Page sizes in databases range from 1KB to perhaps 32KB and are rarely larger than 64KB. If a database's storage is kept on a stripe set with a logical stripe block size of say 256MB then there will be many many database pages stored on any given stripe block. As I said, databases do not read or write entire storage files, but rather only those pages that contain requested data. That means that as data is read from and written to disk the same stripe block will be read or written over and over again. As I have noted, when a RAID5 stripe block is written then the corresponding blocks on all of the other drives must be read in order to calculate a new parity value for the newly written block which then has to be written to the corresponding parity drive for that logical block. That means that all <N> drives in the stripe set, as noted above, are busy and cannot be used for reading other data blocks during the write. With a large stripe block size that same stripe block will be written over and over again as the database engine writes to various "pages" contained in that single block.

The larger the stripe block size the worse is the effect that this has on performance thereby multiplying the so called RAID5 penalty many times. Large stripe block sizes do not have as great an effect on the performance of RAID10 arrays because the multiple writes to the same stripe block affect performance only for that one drive pair, the other drive pairs in the stripe set are not affected at all. I will note that I always recommend that database administrators insist on the smallest stripe block size they can coerce their storage administrators to configure, tending to stripe blocks of less than 1MB and preferably less than 256KB, even

when RAID10 is selected. But with RAID5 this issue is critical to performance and unfortunately sites that accept their storage administrators' and storage vendor's insistence that RAID5 is safe and fast also accept the assurances from the same sources that bigger is better!

What's wrong besides a bit of performance that I didn't know I was missing?
OK, so that brings us to the final question which is: What is the **REAL** problem with RAID5? It does recover a failed drive right? So writes are slower, I don't do enough writing to worry about it and the cache helps a lot also, I've got LOTS of cache and I can buy more! (Do you catch the logical and financial fallacy of that last piece?)

The problem is that despite the improved reliability of modern drives and the improved error correction codes on most drives, and even despite the additional 8 bytes per block of error correction that EMC and some other vendors put on their more advanced disk subsystems (if you are lucky enough to use one of these), it is more than a little possible that a drive will become flaky and begin to return garbage. This is known as partial media failure. Up to a point the drive itself will correct single bit errors. Now SCSI, higher end SATA, and other intelligent drives reserve several hundred disk blocks to be remapped to replace fading sectors with unused ones when read errors cannot be corrected by rewriting the sector. However, if the drive is going these will not last very long and will run out and the drive does NOT report correctable errors back to the OS! Therefore you will not know the drive is becoming unstable until it is too late and there are no more replacement sectors and the drive begins to return garbage. [Note that the recently popular IDE/ATA/SATA drives do not (to my knowledge) include bad sector remapping in their hardware so garbage is returned that much sooner.] When a drive returns garbage, since RAID5 does not EVER check parity on read (RAID3 & RAID4 do BTW and both perform better for databases than RAID5 to boot) when you write the garbage sector back garbage parity will be calculated and your RAID5 integrity is lost! Similarly if a drive fails and one of the remaining drives is flaky the replacement will be rebuilt with garbage also propagating the problem to two blocks instead of just one.

Need more? During recovery, read performance for a RAID5 array is degraded by as much as 80%. Some advanced arrays let you configure the preference more toward recovery or toward performance. However, skewing priority towards performance during recover will increase recovery time and increase the likelihood of losing a second drive in the array before recovery completes. Losing a second drive in a RAID5 array will result in catastrophic unrecoverable 100% data loss. RAID10 on the other hand will only be recovering one drive out of 4 or more pairs with degraded performance ONLY of reads from the recovering pair making the performance hit to the array overall only about 20%! Plus there is no parity calculation time used during recovery – it is a straight data copy so recovery time is much shorter.

What was that about losing a second drive? Well, with RAID10 there is no danger unless the one mirror that is recovering also fails and that's 80% or more less likely to happen than that any other drive in a RAID5 array will fail! And since most multiple drive failures are caused by undetected manufacturing defects you can make even this possibility vanishingly small by making sure to mirror every drive with one from a different manufacturer's lot number.

I can hear you say, "This scenario does not seem likely!" Unfortunately it is all too likely. It happened to me and I have heard from several other DBAs and SAs who have had similar

experiences. My former employer lost 50 drives over two weeks when a batch of 200 IBM OEM drives began to fail. IBM discovered that the single lot of drives would have their spindle bearings freeze after so many hours of operation. Fortunately due in part to RAID10 on our database storage and in part to a herculean effort by DG techs and our own people over 2 weekends no data was irretrievably lost. HOWEVER, one RAID5 file system was a total loss after a second drive failed during recovery. Fortunately everything was backed up on tape, but that file system was down for several hours while the latest backup tapes were retrieved from off-site storage causing 1500 developers to twiddle their thumbs for most of a day. I have conservatively calculated the lost time cost at over $800,000. That one internal service outage of only a few hours cost more than the extra cost of using RAID10 for all of those filesystems arrays! If that file system had contained database storage, it would have contained transactions and mid-day data updates that would have been lost or would have had to be replayed leading to additional downtime and it would have been customers who were affected not employees and internal productivity. Given the nature of my then employer's business there might have been contractual penalties and lawsuits resulting from such an outage.

Conclusion? For safety and performance favor RAID10! The original reason for the RAID2 through RAID5 specifications was that the high cost of disks was making mirroring (ie RAID1), impractical for many organizations. That is no longer the case! Drives are commodity priced, even the biggest fastest SSD drives are cheaper in absolute inflated dollars or adjusted dollars than drives were then and the cost per MB is a tiny fraction of what it was. Does RAID5 make ANY sense anymore? Obviously I think not.

Speaking of SSD drives, their failure rates, including bit loss, are nearly identical to that of spindle drives. Also SSD memory cells have a lifetime limited by the number of writes cycles they are subjected to. For that reason modern SSD drives shuffle data to less used cells when it is written to spread the load and push the failure time away down the calendar. But, like the replacement sectors on magnetic drives eventually the cells will reach their end-of-life and begin to fail and return garbage. So everything I have said about the dangers to your data of using RAID5 does not change if you are now using SSD drives instead of magnetic platter drives.

To put things into perspective: If a drive costs $1000US (and most are far less expensive than that) then switching from a 4 pair RAID10 array to a 5 drive RAID5 array will save 3 drives or less than $3000US. What is the cost of overtime, wear and tear on the technicians, DBAs, managers, and customers of even a recovery scare? What is the cost of reduced performance and possibly reduced customer satisfaction and confidence? Finally what is the cost of lost business if data is unrecoverable? Of the legal ramifications if that data is required by government mandate? I maintain that the drives are FAR cheaper! Hence my mantra:

NO RAID5! NO RAID5! NO RAID5! NO RAID5! NO RAID5! NO RAID5!