Understanding RAID for Firebird

Table of Contents

Introduction

RAID configurations are more or less now standard for enterprise disc access. And it has become common to use external NAS and SAN for data storage, in addition to the more familiar internal RAID arrays. In all cases the array of discs is set up according to one of the standard RAID configurations. Choosing the right configuration is important for Firebird. The wrong configuration can seriously impact upon database performance.

Intended Audience

This article is not intended to be an exhaustive study of RAID in general. And it should also be noted that this article does not consider RAID from the viewpoint of heavy sequential I/O. If that is your use case scenario then move along. What you learn here will not be applicable.

Above all, this article attempts to clearly explain *why* many experts argue[5] that mirrored RAID is preferable to parity RAID for database servers. If you deploy production Firebird servers that do a percentage of random writes to standard hard drives then this article is for you. It summarizes the main issues to consider in relation to choosing the right RAID level. It is assumed that the budget for the disc subsystem is in the region of CU 1K - CU 10K (where CU is a currency unit roughly equivalent to 1 USD, EUR or GBP). We are not really considering disc subsystems priced in the 100K plus bracket.

It should also be noted that we mainly discuss mechanical disc drives here. There is a section later that looks at the impact of Solid State disks upon RAID.

There is a glossary which gives a quick description of the common RAID levels and other terms used in this document.
See the references section for links to articles that go into RAID in greater depth.

RAID under the hood.

RAID in more detail

RAID[1][2][3][4] sets out to achieve three goals:

- Increase data throughput by sharing disc i/o across numerous discs.
- Protect against disc failure by duplicating data to two or more discs.
- Allow failed discs to be replaced without taking the system off-line.

Redundancy brings these benefits but it also brings some liabilities. By its very nature redundancy implies writing the same data to more than one disc. There is always a price to pay for disc redundancy, in the financial cost of more discs and in the time required for more writes. RAID systems achieve this redundancy in one of two ways:

1. For every write to disc A copy the same data to disc B. If disc A fails the data is still available on disc B. This is RAID 1, RAID 10 etc. Generically this can be called Mirrored RAID.
2. Write a block of data on one disc and a parity block on another. Randomize these blocks across all available discs. If a disc fails the data blocks and parity blocks on the surviving discs enable reconstruction of the data on the failed disc. This is basically RAIDs 5,6, 50 and 60. This RAID family can be called Parity RAID.

Ultimately the bottleneck for RAID is the cost of writing to disc. Each RAID strategy attempts to mitigate this. Choosing the right RAID requires understanding two things:

- Are writes sequential or random?
- How much time is spent writing to disc?

If writes are sequential a number of factors come into play. As the volume of data increases the speed of the disc subsystem becomes all important. However, for database applications the volumes are typically smaller than the disc block size and fairly random in distribution. In this scenario more time is spent locating the correct disc sector than actually changing the bits.

Calculating raw RAID performance in an OLTP environment

In general it is reasonable to say that disc access is primarily read and primarily sequential. A web server or video streamer, for example. Even standard desktop usage will be primarily sequential reads with occasional sequential writes, as files are saved to disc. In contrast a mail server, or a database server will be doing a lot of random i/o, especially if there are a large number of concurrent users. And in the case of a database server the writes will often be very small, especially if the data has been normalised to 5NF with integer based surrogate keys. Small writes mean that a page of data must still be written to disc, even if the page is incomplete. ie, even if only one byte must be written to disc, the time required to do so is the same as writing an entire page.

Firebird is highly optimised to support this kind of useage. A high number of concurrent users of a typical database application will see a high proportion of random, small writes to disc. This usage pattern is often

referred to as OLTP - On-Line Transaction Processing. The use of the disc subsytem is very different to that of a data warehouse, where sequential bulk data uploads will occur regularly and most activity will be reads of data. Likewise the usage pattern is unlike that of a file server where disc writes, although frequent, will be larger than that of a database write. When writes are larger they are usually more sequential, which reduces the load on the discs.

So, when we come to measure the performance of disc drives for OLTP we are not so much interested in data throughput as we are in the number of times we can access the disc in a given period. For this reason we use the concept of IOPS (input/output operations per second) to ascertain the performance of the disc subsystem used by a database server.

There is a simple formula to calculate IOPS. Add the average time it takes for the disc platter to spin to the average time it takes to line up the drive head to a disc sector. These values are usually in milliseconds. Divide that value into 1000 and we have the IOPS. The former value is usually expressed as 'Average Latency' and the second value as 'Average Seek Time'. Some vendors provide separate data for read and write seeks. Some don't seem to provide this data at all. In fact it doesn't matter too much if some manufacturers don't give out this data. It can be derived anyway. There is very little difference between brands of discs. The important factor is primarily disc speed. Average latency can be calculated from the rpm value for the drive. It is just

```
Avg Latency = (60 / RPMs / 2) * 1000
```

There is no way to theoretically calculate Average Seek but it is possible to make a guestimate based on what published data is available. Using those guestimates we can calculate the IOPS for a disc thus:

```
IOPS = 1000 / (avg latency + avg seek)
```

Here is a rough guide to IOPS for different disc speeds. Times are in milliseconds.

RPM	Avg Latency	Avg Read Seek	RIOPS	Avg Write Seek	WIOPS
5,400	5.56	9.40	66	10.50	62
7,200	4.17	8.50	79	9.50	73
10,000	3.00	3.80	147	4.40	135
15,000	2.00	3.50	182	4.00	167

It is evident from this data that disc speed is the determining factor in the number of times per second that the head can access the disc.

What does IOPS really mean?

IOPS is a theoretical value. It has no relation to actual data throughput. Indeed sustained data reads or writes will see a significant distortion of the IOPS data. However, for the purposes of this discussion IOPS is the ideal way to measure disc access. The physical limitations of disk latency and seek time are real. IOPS indicates the maximum number of times per second that a drive could randomly read or write to a disc.

The Write Penalty

The next stage in studying RAID performance for OLTP is to consider the Write Penalty. As previously noted, redundancy means writing data to more than one disc. Each RAID configuration has a write penalty. Here is a quick guide:

RAID Level	Min. No. Disks	Write Penalty	Comment
JBOD / RAID 0	1	1	One disc. One write. However it could be argued that for RAID 0 the penalty is 1 / No. of Disks.
RAID 1	2	2	In fact the write penalty is directly related to the number of disks in the mirror. If three disks are mirrored then the penalty is 3. For the purposes of this discussion we will assume just mirrored pairs.
RAID 5	3	4	Every write requires a read of the data block in the stripe, a read of the existing parity block then the actual write of the updated data block and parity block. It is not entirely clear from the available documentation whether the write penalty is one read for every disk in the array plus a data block write and a parity block write, or just a data block read, a parity block read and a data block write and a parity block write. We'll give RAID 5 the benefit of the doubt and go with the lower figure.
RAID 6	4	6	Similar to RAID 5 except that the extra parity block requires an additional read and write.

Comparing the IOPS of different RAID Levels

Having established the concept of IOPS and the Write Penalty we need to put these ideas together to get an idea of the kind of theoretical performance we can achieve from different RAID implementations. There is a simple formula to calculate the theoretical IOPS for a given RAID configuration:

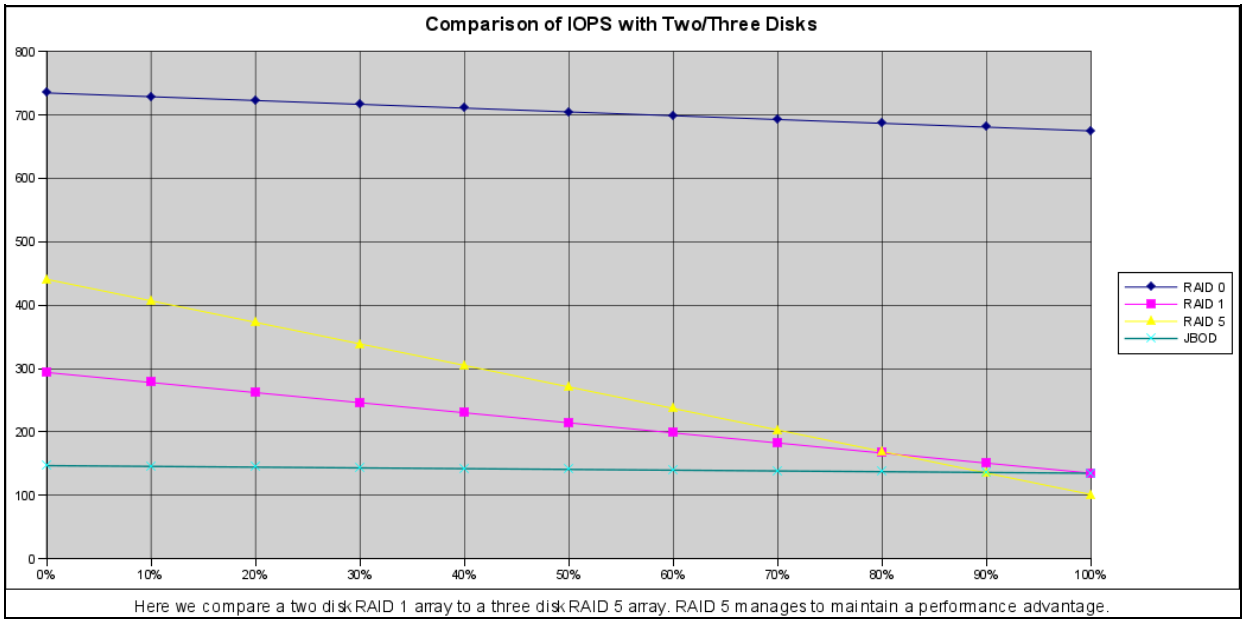
```
((DISK_WIOPS*NO_DISCS*%WRITES)/WRITE_PENALTY) + (DISK_RIOPS*NO_DISCS*%READS)
```

We can use this formula to model the impact of increased writes against reads, depending upon the RAID level chosen. Let's take a look at some graphs.

Note:

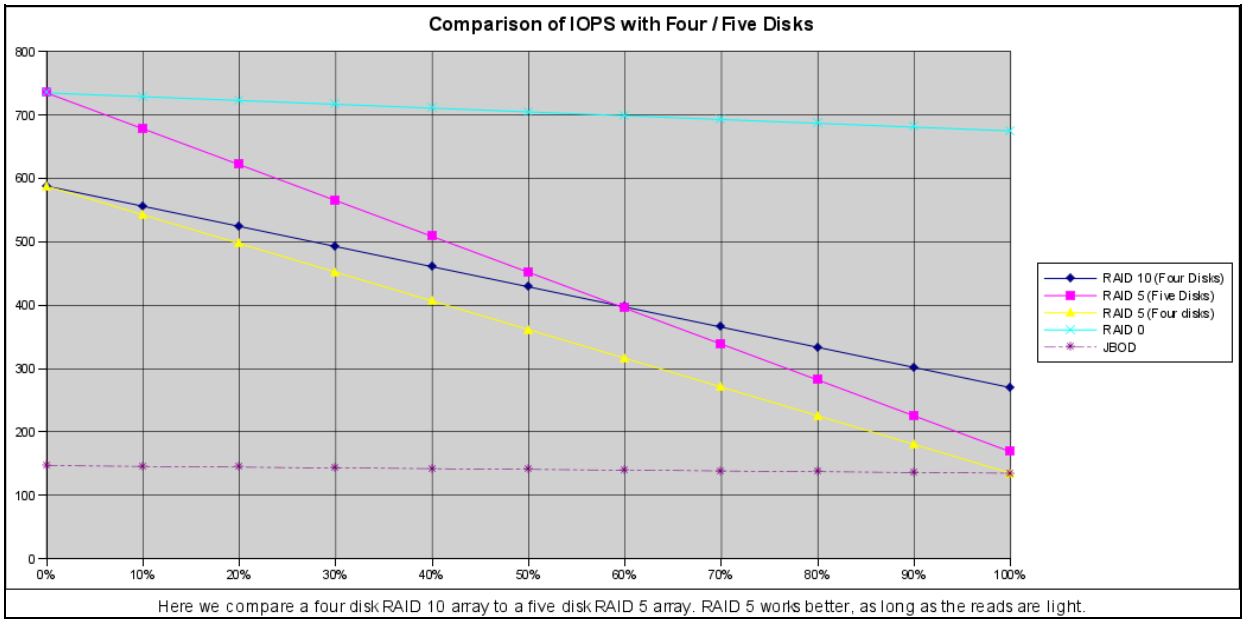
- These graphs use data derived from 10K rpm discs. They provide a reasonably good price/performance/capacity ratio.
- These graphs indicate RAID at its simplest. In reality Hardware RAID controllers mitigate the Write Penalty.
- JBOD shows us the performance of a single disc. The WIOPS only degrade slightly even as the proportion of writes increases to 100%.
- RAID 0 gives us an indication of the massive increase in RIOPS that RAID can help us achieve. Again, performance degrades gently as the proportion of writes increases.
- Both the JBOD and RAID 0 data is by way of comparison only. It is not suggested that they are viable storage solutions for a database server.

Using Two or Three Discs in a RAID



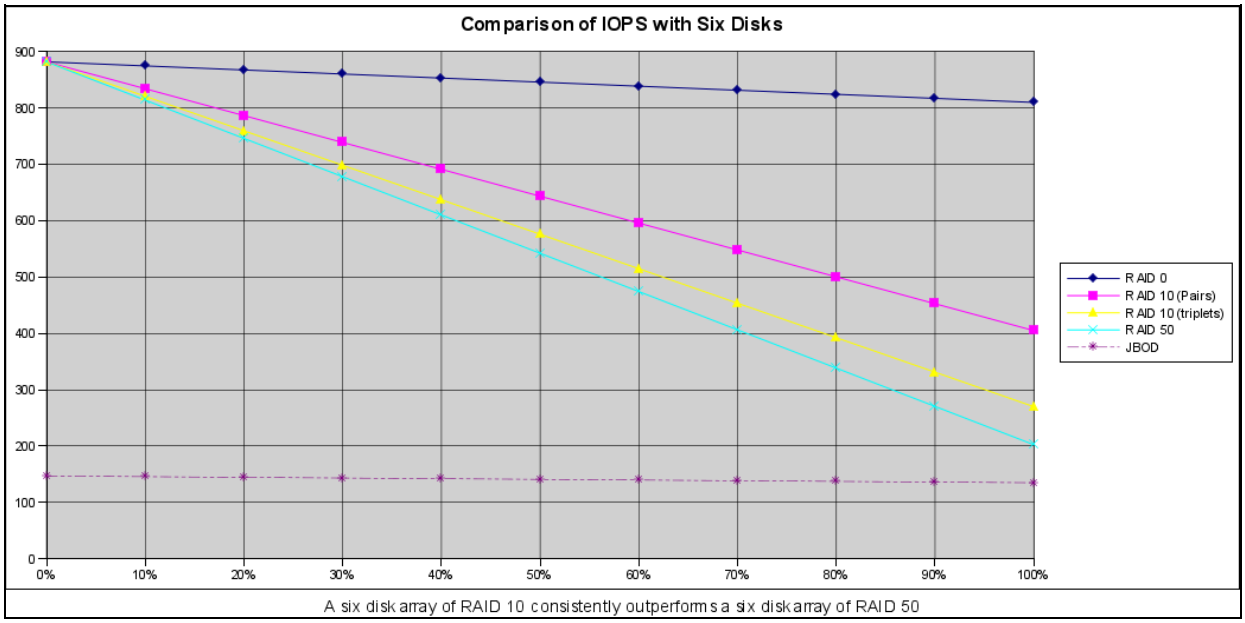
In this graph we compare the two entry level RAID arrays. A two disk RAID 1 with a three disc RAID 5. At this level the extra disk in RAID 5 pays for itself and the random write penalty does not kick in until we tip the %write balance to 80%. In general, for most usages where the reads outweigh the writes this three disk RAID 5 configuration is a good solution.

Four Disc RAID configurations



When we move to the next level we see the random write penalty start to affect RAID 5 heavily. In this configuration we have RAID 10 - two pairs of mirrored disks. Against that, we can see that a RAID 5 four disk array cannot compete with RAID 10. When another disc is added RAID 5 improves its random read performance. However, once the %writes mix hits around 60% the RAID 10 config performs better, despite one less disc.

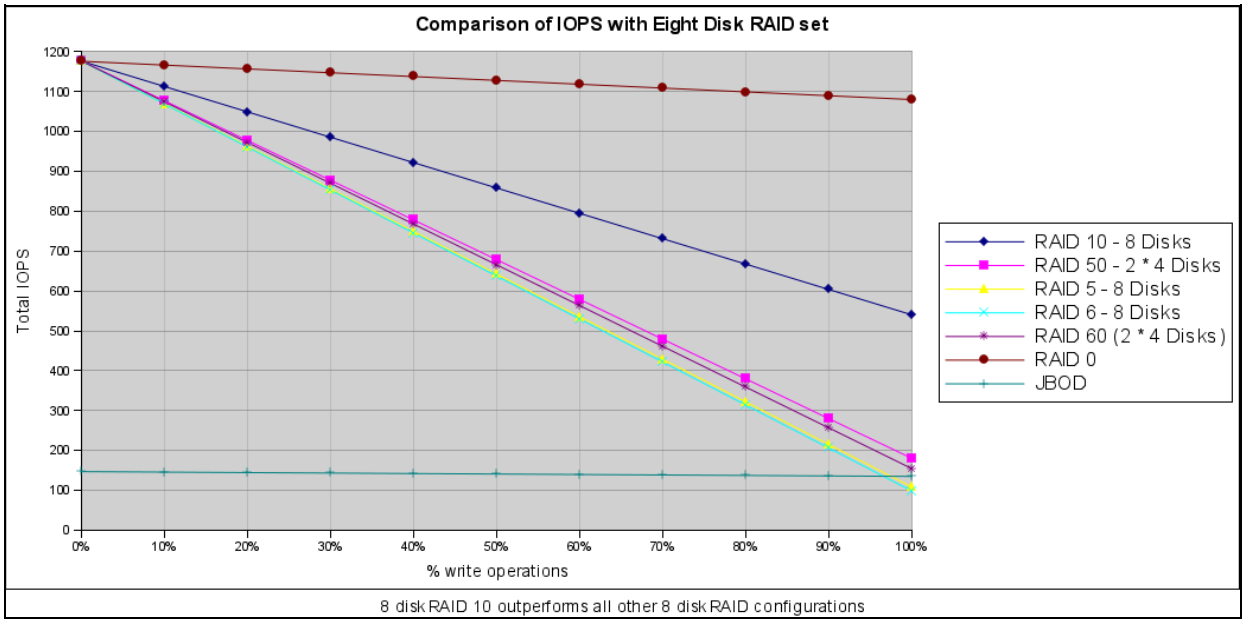
Six Disc RAID configurations



In this graph we look at three different configurations. One is standard RAID 10 of three pairs of mirrored discs. The second is a super redundant configuration of RAID 10 in two pairs of triplets. That means three copies of the data. And finally, a six disc configuration of RAID 50, ie a pair of RAID 5 arrays combined. The results are quite startling. Given the same number of discs, both RAID 10 configurations outperform any variant of RAID 5.

*Note:*This graph uses the standard random write penalty of 4 for a RAID 5/RAID 50 array.

Eight Disc RAID configurations

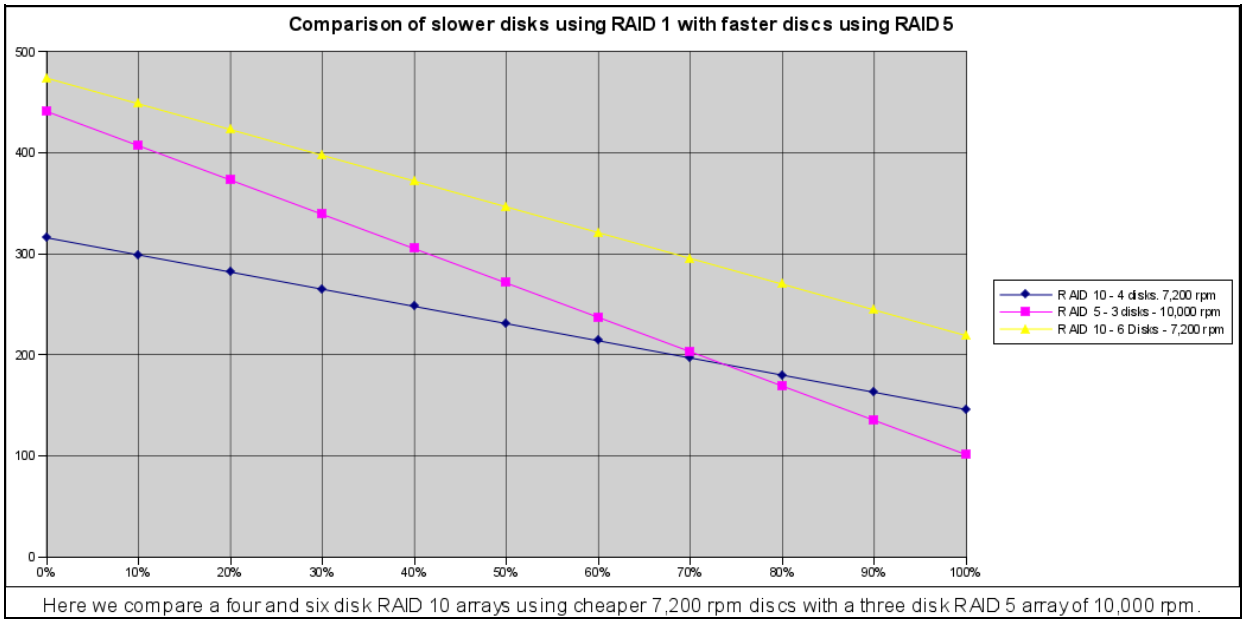


Here we show the impact of different eight disk arrays. Again, given the same number of discs, RAID 10 performs better than RAID 5/6/50/60.

*Note:*This graph uses a non-standard random write penalty based on number of discs + 2 for RAID 5 and +3 for RAID 6. This has the effect of pulling the graph lines for parity based RAID arrays closer together. However, even with the simpler calculation of the write penalty RAID 5 type array will still underperform against the RAID 1 family.

All these graphs indicate that, when disc numbers are identical that a mirrored RAID outperforms parity RAID for random I/O. This is consistent for reads and writes. When the parity based array is larger than the mirrored array the parity array will give better performance unless the read/write mix is heavily skewed to writes. On this basis alone a RAID 10 solution will always produce better performance than a Parity RAID solution and it will often be cheaper.

Can slower discs be better value than faster discs?



Here we compare a four and six disk RAID 10 arrays using cheaper 7,200 rpm discs with a three disk RAID 5 array of 10,000 rpm.

The next question to ask is - can a RAID 10 array of slow discs (7,200 rpm) outperform a smaller array of 10,000 rpm discs configured in RAID 5? In fact, if we throw six discs at the problem we do get better performance. Considering the price differential between 7.2K and 10K drives (approx 4 times) this starts to look very interesting. The next stage in this analysis would be to consider the additional cost of power consumption of the slower drives. It is interesting to note that the current generation of 'green' 7,200 rpm drives draw a lot less power than the 10,000 rpm drives, so a RAID 10 array of 6 slower drives may actually make technical and financial sense.

Solid State Drives (SSDs) - A game changer?

Until recently SSDs were a bit toy town. They had impressive speed but the disc capacity was poor and the price was excessively high. They also had a limit on the total number of writes possible over the lifetime of the disc. This made them a poor choice for a disc subsystem configured for database use. That however is starting to change. SSD cards with a capacity of around 120GB at a price of around 200 CU are now available. Even the slowest single disc of the new generation offers 15 times the WIOPS available from a mechanical eight disc RAID 10 configuration. And the limit on the number of writes has been raised significantly. Where disc I/O is a major issue the solution is SSD. None of that really negates the discussion in part 1 on choosing a RAID configuration. The same rules apply although they matter less because the bottleneck is unlikely to be the disc i/o, even with RAID 5.

Problems with SSD drives

- WIOPS deteriorate with use because FLASH memory must delete the memory block before writing to it. This is ameliorated in some drives by keeping a proportion of the disc free for writes and thus moving the data around. This also helps the levelling problem. Even so, SS disks still hugely outperform mechanical disks.
- The price of drives is still a hindrance to widespread use, although the price/capacity ratio is improving.
- The small capacity of drives is still a problem. Magnetic media continue to offer significantly larger capacities while their cost per MB remains low.
- SSD manufacturers use a huge amount of trickery to generate their incredible performance statistics. They are certainly not as good as they seem.

It is probably true that mechanical disk drives will go the way of tape drives. In a few years time SSD will be the norm. However we are not there yet. In the meantime we will see increasing numbers of mixed systems where SS disks will be used for performance critical applications while mechanical drives will be used for bulk storage.

Summary

These graphs demonstrate pure RAID performance without the influence of any kind of intelligent controller. There is no guarantee that this level of IOPS can be achieved in a similar real world system because of other physical limitations. But these figures give us a base-line for further analysis and above all they indicate the impact of the main physical limitation of RAID technology when running an OLTP database server.

- More discs always increases the theoretical number of Read IOPS.
- Write IOPS are always penalised by every RAID configuration.
- Mirrored RAID will give consistently better OLTP performance unless the disc subsystem uses just two or three discs.

Conclusion

Hopefully this article has explained cleary why mirrored RAID is superior to parity RAID for database servers. Unfortunately that is just one consideration when making a decision to build a RAID array. There are many aspects that are not discussed here, because they are not directly Firebird related. It is worth throwing out the following for consideration:

- Cheap firmware disc controllers should be avoided at all costs. The usually use a proprietary disc layout and when they fail the only way to recover data is with a similar firmware controller. If money for a proper h/w controller is not available use a software controller instead.
- Hardware controllers usually provide the best performance and appear to mitigate many of the failings of parity RAID.
- Don't buy a hardware controller without battery backed cache or similar.
- Restoration time from disk failure is considerably higher for parity based RAID than mirrored RAID. This will be particulary painful if the disc subsystem is already under-specified.
- Always have a secondary backup system. All RAID solutions can break catastrophically.[7][8]
- All disc subsystems have their limits. Even h/w raid controllers with battery backed cache have their limits - often just kicking the bottleneck down the road a bit.

Glossary and References

Glossary

RAID
: An acronym for 'Redundant Array of Inexpensive Discs'. The term has its origins in the mainframe days when storage was expensive and the possibility to use off the shelf disc drives was financially interesting, so long as the data integrity could be guaranteed.

RAID 0
: Two or more discs are combined to create a single large drive.

RAID 1
: Two discs are mirrored. If one fails the data is still fully available on the other disc.

RAID 1+0
: Two disc pairs are mirrored and the resulting mirrors are combined to make a larger drive. Usually known as RAID 10.

RAID 0+1
: Two disc pairs are combined and the the two resulting drives are then mirrored. Much less robust than RAID 10 due to the nesting of RAID 0 within RAID 1.

RAID 2
: Obsolete. See RAID 2 definition on wikipedia.

RAID 3
: Obsolete. See RAID 3 definition on wikipedia.

RAID 4
: Obsolete. See RAID 4 definition on wikipedia.

RAID 5
: Uses block-level striping of data across all discs. Parity data is evenly distributed across all disks.

RAID 50
: Sets of RAID5 arrays are combined to create a larger drive.

RAID 6
: Like RAID 5 but with two separate parity blocks.

RAID 60
: Similar to RAID50.

JBOD
: Just a Bunch Of Discs.

IOPS
: Input/Output Operations Per Second.

Striping
: Determines the size of each data segment that is read/written to. Makes better use of multiple discs in the array.

Parity
: Every time data is written a parity value is calculated to allow for data recovery in the case of disc failure. This parity value is written to a different disc.

Controller
: An array is managed by a disc controller. This can be implemented in hardware, software or firmware.

References

[1] The original paper explaining RAID. Written as long ago as 1988

[2] Wikipedia on RAID in general. A useful starting point to understand the wider aspects of RAID.

[3] Wikipedia on standard RAID levels

[4] Wikipedia on nested RAID levels

[5] The Battle Against Any Raid Five. Strong opinions backed up by good arguments.

[6] The Linux RAID wiki. All you ever wanted to know about software raid on Linux.

[7] Google Labs study on disk failures. Discs fail. Cheap, expensive, it doesn't matter. Read this and be prepared.

[8] Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? A companion article to the one from google labs. Required reading.