SANE SAN

The Random Acronym Seminar (or RAS for short…)

James Morle, Scale Abilities, Ltd.

1.0 Introduction

This paper talks about storage within SANs. It is easier than ever to implement a badly laid out SAN, with the levels of abstraction and data sharing made possible through this technology. We will look at ways this can be simplified though careful planning, and discover why its a good idea to lie to your boss. Before that, though, it's worthwhile taking a journey into the past to find out why all this stuff exists, and why there are so many acronyms in the storage industry.

2.0 The Dawn of Time

Time began in 1833, when Charles Babbage decided that a new Analytical Engine, which accepted **input** in the form of punch cards and produced **output** in the form of a result, was a really great idea. Babbage unwittingly invented I/O back then, and for sure wasn't aware of the complexity and acronym madness to come.

The important principal here, though, was that the Analytical Engine encapsulates what computing is all about; **putting data in, and taking data out**. This principal is often forgotten in the complex state of modern computing, but still applies more than ever. Remember this; it's the only reason you have a job.

Ultimately, the time it takes to get an answer back from a request is the truest measure of response time, so we have the most fundamental picture of I/O completed:

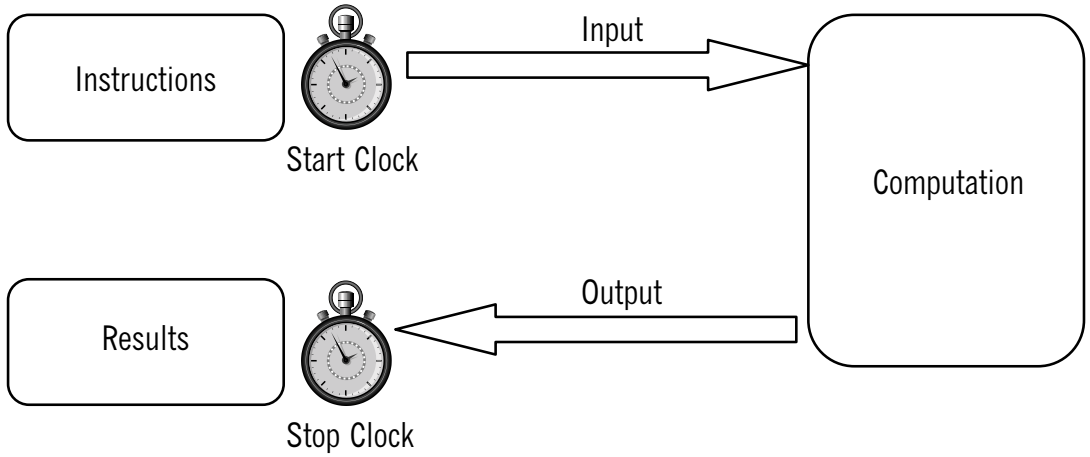

FIGURE 1. Primary I/O

3.0 The Industrial Revolution

Eventually, in the early 1950s, persistent magnetic storage was born, enabling new dimensions in I/O, most notably that of **data-based computing**. Initially tape-based,

the random access characteristics of the disk drive soon won over in the late 1950s, early 1960s, and things haven't changed radically since! Yes, things have got cheaper, faster, smaller, but nothing truly radical has changed within the humble disk drive. Of course, an acronym was devised for this new type of storage: Random Access Method of Accounting and Control (RAMAC)! The mainframers really went to town, though, and coined the first generic term for disk storage - Direct Access Storage Devices, or DASD (*dasdee*), a term still very much in use in IBM circles.

This shift into data-based computing is where the real fun stuff started. We could store information in an automated way, and retrieve it programmatically. One obvious first candidate for this new technology was, of course, junk mail, the forefather to spam! It wasn't all bad, though - Other more useful usage of this technology included the birth of airline reservation systems and automated banking.

With this persistent storage, we introduce further elements into I/O model:

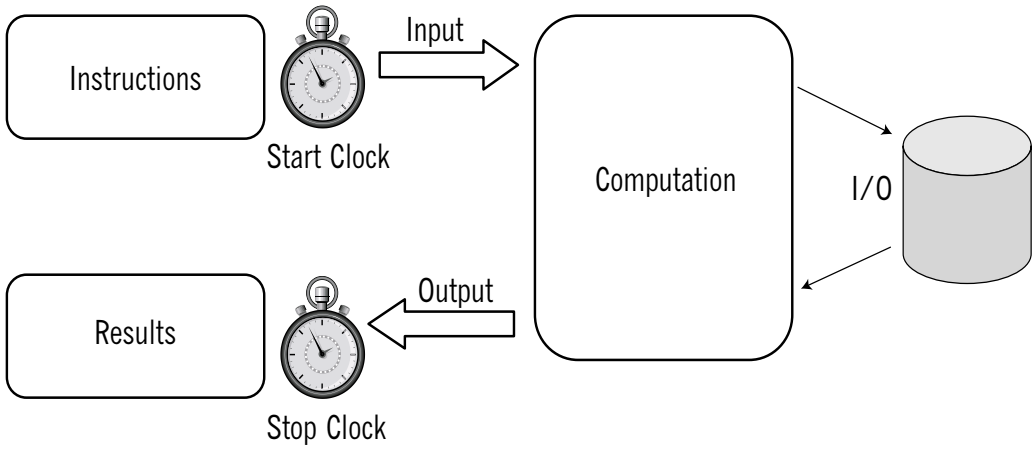

FIGURE 2. Secondary I/O

Note here that the stopwatches don't move. The only thing that matters is *still* the response time to the end user, and the resulting output. During these early days of the industrial revolution, most of the work was batch-oriented, and so the response time was more closely linked to *throughput* rather than latency.

Back to the acronym madness. The disks in these systems were subsequently to be known as JBOD - Just a Bunch Of Disks. This term is used to describe an array of disks that offer *no protection* against failure. In 1978, the first work started into the research that resulting in the formal definition of the Redundant Array of Inexpensive Disks (RAID) in 1987. The definition was provided by Gibson, Katz and Patterson at the University of California at Berkeley in a ground-breaking whitepaper. Note, though, that the real purpose of this research was to produce reliability in disk storage that was not available before this point without 1:1 mirroring.

4.0 Commercial UNIX Computing

It is probably fair to say that most large database developments now take place on UNIX platforms, so it's time to switch gears. UNIX databases were, of course, started

on JBOD-only systems, mostly where the disk drives were internal to the host system. Reliability was provided solely by robust (hopefully) backup regimes, and mostly worked for the quantities of data found on such systems in the early 90s. Database sizes were typically in the tens or hundreds of megabytes at this stage, with tens or hundreds of users!

Although OLTP had been available for some time in the mainframe environment, the UNIX database environment quickly became synonymous with OLTP. In the OLTP world, latency is the dominant factor in response time, and so the latency associated with the I/O system became very critical. This led to carefully thought out I/O layouts, and could be very predictable, as long as the system was not overloaded[1]. It was mostly practical to have sufficient disk spindles for the I/O load, because the capacity of the drives was in the 1-2GB range at this time.

As time went on, and the client-server revolution took off, testosterone production increased to new levels. By 1993, I was working on a 3,000 user, 75GB database, running on 3 nodes of the brand-new Oracle Parallel Server! This clustering brought new challenges to I/O systems. Suddenly, the disks needed to be accessible by more than one host, and the concept of the disk array was launched upon the unsuspecting open systems world.

Access to these disks was through the woefully restrictive SCSI-2 channels available at the time. Oh, another acronym crept in there - Small Computer System Interface. Funny, having six cabinets, each 2 metres in height, it didn't seem that small a computer system at that time.

5.0 The Information Age

The next step in computing was the emergence of the Information Age[2]. In addition to the widespread adoption of eCommerce and other online services, the price of disk storage started dropping at alarming rates. This prompted existing companies to start trying unheard of craziness, such as keeping many months of Call Data Records online in a database. As an implication, availability, performance and recoverability needed a more reliable solution.

6.0 The Storage Network

The demands of the Information Age pretty much lead directly to the concept of the Storage Network. This term refers to the connection of storage to a host, or a number of hosts, by means of open networking standards, rather than electrical protocols such as SCSI. This is a good opportunity to introduce the next set of acronyms: SAN and NAS, the real subjects of this paper.

1. At which point queuing theory accurately describes the ensuing long response times.
2. Term courtesy of Larry Ellison

6.1 NAS - The Definition

Network Attached Storage. In a nutshell, this is a layer 7 (mostly) implementation of the Storage Network. What does this mean? It means that the two communicating nodes of the network do so at a high level, using the NFS protocols. Picture the storage device as another UNIX server with exported NFS volumes, and the server as an NFS client that mounts these volumes.

In reality, things are a good deal more complex than this under the covers. Taking the Network Appliance Filers as an example, facilities are present to take instant snapshots of whole filesystems, dynamically grow the filesystem, and provides full fault-tolerance. Of course, though these devices use standard networking technologies, such as Gigabit Ethernet, they will always exist on their own switched network in a storage system with *any* reasonable performance requirement.

Multiple hosts can access the storage device concurrently, and backup/restore can be performed from a different host than the original database server.

6.2 SAN - The Definition

Storage Area Network. SAN's operate at a lower level than the NAS solution. Using Fibre Channel for connectivity, SANs use the SCSI protocol over the fabric. Note, this does not imply any of the limitations of SCSI, only the adoption of the protocol. SANs present virtual disks to the various hosts within the network, and the host is free to use that device as if it were a local drive.

Again, things are complicated under the covers. Devices such as the EMC Symmetrix provide snapshot support, redundancy, hotspot identification and data relocation. Multiple hosts are given selective access to the disk drives as required.

6.3 SAN vs NAS

There are fierce battles between the NAS and SAN evangelists, and we're not going to get into all that here. Instead, a few statements about current suitability might be more appropriate.

If sharing files is the requirement, NAS is a clear winner. It can provide a low maintenance, high performance solution for this problem, and does not even require a private network for this facility. SAN has no capability for sharing files, unless the hosts are in a cluster, and a clustered filesystem is used.

For database work, SAN currently has the edge, in my opinion. There are a few points here:

1. SAN has a lower latency for I/O, and a lower server overhead. This is because of the higher level protocols used for the I/O, but is nowhere near as high as might be expected. A well configured NAS system easily outperforms a low performing SAN solution. Also, more on this subject in a moment.

2. Fibre Channel is a more robust networking technology that Gigabit Ethernet. Things like guaranteed delivery, larger frame sizes, and the direct support for the SCSI protocol give it the edge here.
3. SANs leave more control with the administrator. NAS tries to do too much magic performance stuff behind the scenes.

Going back to the first point, the latency for I/O is being comprehensively addressed by a brand new acronym, DAFS: Direct Access Filesystem. Using similar technology to the the reflective memory technology used in Digital/Compaq's cluster systems, DAFS allows processes on the server to issue I/O to the NAS device using user-mode memory copying. Not only does this dramatically reduce the code path on the server, but it also eliminates a context switch to kernel mode for each I/O. Combined with an ultra-low latency, high bandwidth interconnect such as InfiniBand, the overhead of NAS could be significantly lower than SAN.

For the rest of this paper, we will concentrate on SAN. Many of the points discussed apply equally to both types of interface anyway.

6.4 The Tendency towards Anarchy

This seems like as good a place as any to preach a little about lying to your boss. Everyone does it anyway, and at least there's a good reason behind this one. In fact, this lie was pretty much invented by IBM in the early days, so it can't be a bad thing, can it?

The thing that makes this lie necessary is that disk drives have just got too big for their boots. Just look at this:

	Seagate Barracuda 4LP	**Seagate Cheetah 73LP**
Capacity	2.16GB	73.4GB
Rotation Speed	7200rpm	10000rpm
Rotational Delay (avg)	4.1ms	3ms
Time to read 32KB	6ms	3ms
Seek Time (avg)	9.4ms	4.9ms
Total time for single I/O	19.5ms	10.9ms
I/Os per second (conservative)	51	92
I/Os per 100GB	2550	126

OK, it's an extreme example, but still a valid one. We have compared a drive from about 7 years ago, with the fastest mainstream drive of current times. The capacity is 34 times that of the elderly drive, and yet the I/O capability is only 1.8x. That's not a great thing for database workloads - it means we cannot do as much I/O per GB as we could 7 years ago. This is the first piece of evidence in our favour. Note: I am not advocating buying 7 year old drives!

Consider this also. For random I/O, the length of each seek becomes the dominant factor. If the subsequent accesses are close to each other, this latency can be minimised. For example, in the case of the Cheetah drive, a single track seek is 0.6ms, whereas a full bore seek is greater than 5ms. On the outer edges of the disk, more data is stored in

each track (due to Zone Bit Recording). This means that the outer tracks of the disk are faster than the inner one from both on a seek and transfer time basis. Therefore, it would be optimal to place database files on the outer tracks of the disk.

So here's the lie: **Always lie about available space in your SAN**. More specifically, especially if you cannot tell a lie, talk about the I/O capacity of the SAN, not the storage capacity!

Luckily, because a disk is circular in shape, the majority of data is stored on the outer edges anyway, so the lie can be minimised a little! In fact, most of the benefit of using outer tracks is found be using only the first 50% of the drive. Still, always use the I/O capacity as your primary metric when determining the capability of your SAN.

This concept of using the correct metric when assessing the capability of your SAN was indeed conceived by IBM, back in the early days of mainframes. It is more important than ever today.

6.4.1 The Implications of SAN

The entry-level SAN is not for the faint at heart. Essentially a very complex computer system in its own right, potentially with more memory than any single host in your network, the storage system is rarely cost-effective unless populated with a large number of drives and shared among many servers. These servers could be presented, either implicitly or explicitly, with logical drives that map to the same physical drive. For that matter, a single server could see the same physical drive as many unrelated logical drives! Take this example of a small section of SAN storage device:

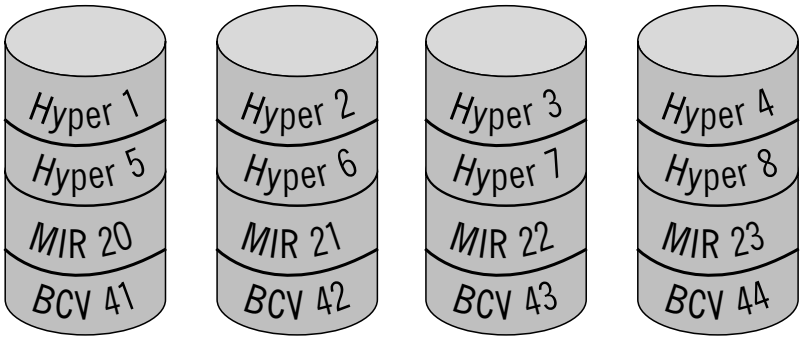

FIGURE 3. Subset of Logical Drives in Typical SAN

In this example, the numbers denote the logical volume number, and the labels indicate the function. MIR is the mirror of a primary device, and BCV is a dynamic 3rd mirror which is broken off for backup purposes. Taking the first physical disk, it is hosting two logical volumes, one mirror, and one BCV. If Hyper 1 were used on its own, it would be a nice, high performance disk storage device. It would provide reliable response times, too.

The first problem comes when Hyper 5 is used; if it's used by the same application it is fairly predictable, if it's not, all bets are off! If the other application just happens to place a busy file on that logical volume, response times to hyper 1 will be affected in an unpredictable manner. Any database-level statistics will not reflect I/O from the other application, as will any system statistics if the other database resides on another server.

Problems are compounded when the MIR 20 volume and BCV 41 are in use. These logical volumes are mirrors of an effectively random other volume! While EMC, or whomever is providing the SAN storage, may be trying to prevent this problem, without physically checking, how do you **know**?

Time for more complexity...

What if the volumes in question are part of a stripe set? It does not matter whether that set is formed by EMC or by a Veritas Volume Manager - how do you know when a stripe member is colliding with a member from another set or even another system? The answer is: With great difficulty. Scripts and some of the newer tools can help you find out what is going on, but is it not better to adopt a logical strategy from the beginning?

6.5 Strategies For SANE SAN

The answer to the problem of colliding logical volumes is to plan ahead.

The first thing to do, regardless of platform or claims by the vendor, is to completely forget the existence of a cache in the SAN. Just forget it, it doesn't exist. No arguments.

Oh, I suppose I could provide a rationale too. This is I/O, the slowest part of the system, and the sword by which it is most easy to die by. The drives exist behind the cache, so they must be used for something, right? Oracle has already 'stolen' the nice cache hits with its own buffer cache anyway, leaving the SAN cache with the foulest smelling random I/O known to man, and a high probability of physical I/O. So - if we plan diligently for a robust and dependable physical I/O later, any gains from front-end caches are 'extra bonus'.

Once you have successfully ignored the cache, the next thing you may want to do is to stretch police tape around some of the drives, and declare them exclusive for this particular database. That's the only way to avoid collisions by other databases, though it kind of goes against the 'spirit of SAN'. Let's invent a new term for this policy: Micro-SAN.

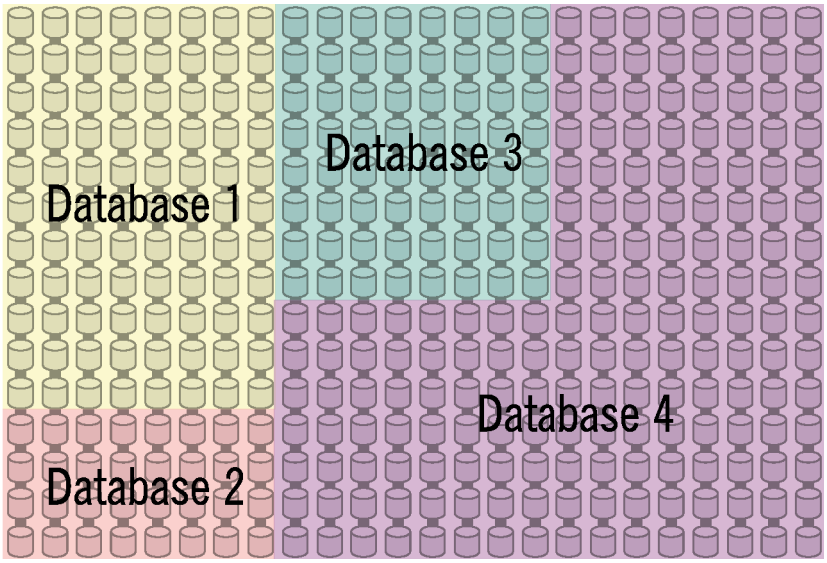

FIGURE 4. MicroSAN

In each microSAN, the drives (and to some degree the internal I/O channels) are made private to specific databases. This has advantages and disadvantages. The major advantage is obvious - discrete drives means predictable I/O . The major disadvantage is that we are not maximising the I/O capability of the SAN device. Let's take a quick look at that.

In this case where microSAN is not adopted, smaller logical volumes could be created on each physical drive, and each database spread across the whole disk array. This allows the total aggregate I/O capability of all the drives to be shared equally among all the databases. I can see the advantage of that, but the downside is that we immediately lose any element of predictable, bounded response times. For this reason alone, I would recommend that applications with mission critical response times should be built on a MicroSAN layout.

Once the MicroSAN is defined, a standard layout for each physical drive needs to be adopted. This makes it easier for administration later, because all of them will look the same. We can invent another acronym for this SLP: Standard Logical Partitioning. Conformance to the law of wasted space is important here, depending on the application of the database:

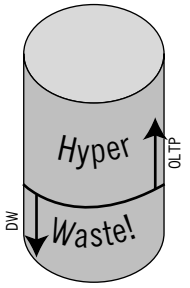

FIGURE 5. Choosing a Standard Logical Partitioning

I am proposing a single logical volume for each physical drive, with a variable amount of wasted space on each drive. Remember, we are losing some capacity by this action, but actually gaining I/O capability. The amount of wastage is application dependent, where OLTP systems will benefit from a greater amount of wasted space on each drive, and Data Warehouses probably don't benefit much at all.

One note of caution when defining the SLP(!). A Fibre Channel connection can only support a maximum of 253 devices, and so some volumes may need to be restricted to certain channels back to the host. Just like the good old days of SCSI, only sixteen times more devices per channel.

Once the SLP is determined, that space can be allocated to the various types of disk object required. By disk object, I refer to both sides of a mirror, any BCV drives, whatever. Here's some nice rules to adhere to:

1. Both sides of a mirrored pair must be hosted by logical partitions of **identical** performance. If this is not the case, one side of the mirror will be effectively degrading the other.

2. If BCV/Snapshot volumes are attached and synchronized to the primary for long periods, it too should exist on a logical volume of identical performance, unless the volume is highly dominated by reads (which don't occur against the BCV volume anyway).
3. If BCV is used for backup, and more importantly **restore**, then it should have the best possible performance. Often, the recovery of a database will have to occur back to the BCV, which then has to be copied over to the primaries. If the BCV is hosted on the worst performing volume in the system, recovery is going to be directly impeded by this!

So, here's the kind of layout that works well for a busy OLTP system:

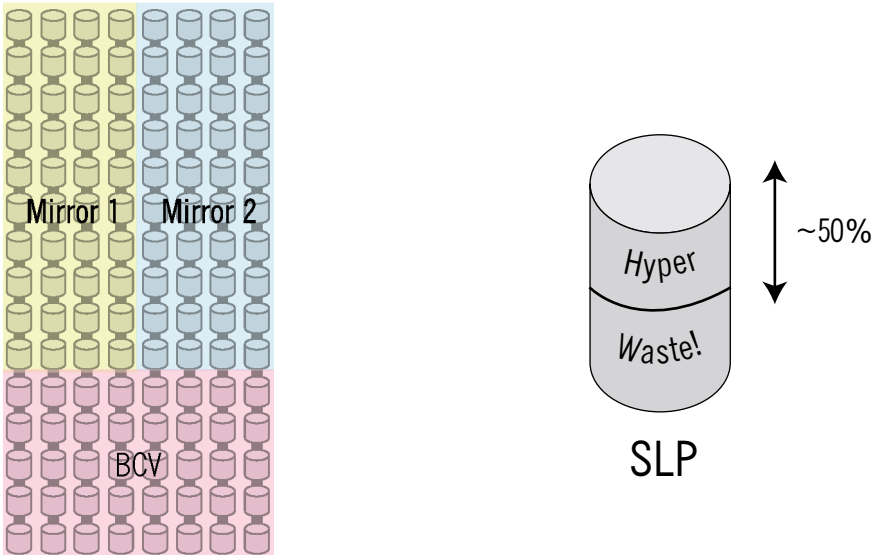

FIGURE 6. OLTP MicroSAN

A problem sometimes occurs when trying to implement such a layout. The internal software of the storage device has to consider many issues related to ensuring data is protected. It may also default to a pseudo-random layout automatically. Therefore, it can often be an uphill struggle to get this implementation the way you might want it. It's worth persisting, especially when we start looking at naming conventions. The MicroSAN above is guaranteed to support 6,000 physical I/Os/s, assuming 100% writes, up to 12,000 I/Os/s for 100% reads. This also assumes, of course, that the distribution of I/O across the available drives is well distributed!

Naming is another vital aspect of a well planned MicroSAN. In the example above, the host can only see a single device for any set of Mirror1, Mirror2 and BCV. With the Mirror1 and Mirror2 volumes being configured symmetrically, we know that any naming convention that includes some kind of location indication will work well for both

sides of the mirror, so we can effectively forget the fact that there are two physical devices for each volume presented to the host.

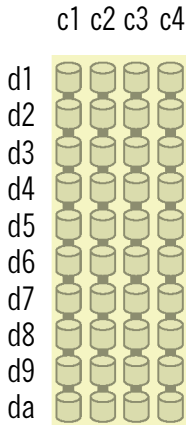

FIGURE 7. Location-based Naming

Using location-based naming, we are now in a position where we can use standard old UNIX tools to monitor the MicroSAN. We could build a Veritas stripe across all the disks in the c1 group, for example, and still be able to immediately determine whether there were hotspots in this stripe. A simple `sar -d` is all that is required, and will show the relative loading of c1d1, c1d2, c1d3, etc.

Once this is done, the MicroSAN is a real treat to manage. No special tools are required to determine exactly what is occurring within the I/O system.

Performance of the I/O subsystem can be a real headache without this kind of planning, and I strongly recommend adopting a similar strategy to the one above when you next have the pleasure of configuring a SAN storage device. It's not much fun doing it, and it can take a very long time to get all the right people agreeing on the layout, but the end result pays for itself within a matter of days.

About the Author. *James Morle is the founder of Scale Abilities Ltd, a specialist consulting company offering both high-end consulting, and unique software products. With over 10 years experience in architecting and building some of the world's largest Oracle systems, James is a well respected member of the Oracle community. Scale Abilities was founded in 2000 to concentrate on providing the highest quality consulting services, in addition to developing revolutionary new consulting-based software products.*

http://www.scale-abilities.co.uk