RAID Theory: An Overview

The Cuddletech Veritas Volume Manager Series

Ben Rockwood, Cuddletech <benr@cuddletech.com>

Revision History

Revision v1.0 April 08th 2002 br

Initial conversion to DocBook

This is the first paper in The Cuddletech Veritas Volume Manager Series. Concepts of RAID (Redundant Array of Inexpensive/Independant Disks) are discused in detail.

Table of Contents

RAID: That it is; What it does

RAID is something all of us have heard about but very few of us understand, at least fully. So lets get off on the right foot. RAID stands for Redundant Array of Inexpensive (or Independent) Disks. There are a dozen or so theories as to why RAID was conceptualized, but the most accepted reason is that once upon a time, not long ago, disks were small and expensive. In order to provide a large amount of data you had to have a bunch of disks all mounted in a single file tree, which was a real mess. So, to solve this problem RAID was born. With RAID you could take a bunch of disks at create a big virtual disk out of them which made administration much easier and more logical. Over time RAID grew to include new solutions for old problems, like disk performance, redundancy, and scalability. And for any skeptics out there, tell me where I can get a 10 terabyte disk drive.... that should make us all agree that RAID has a place in the universe.

Just to try and clear things up a bit more, lets see why we don't simple just need RAID, but actually WANT it. Let's say we're building a production NFS server that will be used to store all of our software. We'll need this system to extremely stable, because if it goes down no one can get or submit code. With RAID we could build a single virtual disk (volume) that would meet our need for 200G of disk. But we also what to make sure that if disks die that we don't go down. So we use a mirror (another set of disks identical to the first set of disks). If a disk dies we're okey, because the mirror will take over; we essentially have 2 identical sets of the same data which are constantly kept up to date. See? Using these 2 simple RAID concepts we've achieved both availability (thats our mirror saving us from disk crashes) and increased capacity (we've got a whole bunch of disks working together, which is cheaper than buying a single 200G disks... if you can find one!).

Okey, enough of the bad examples. Lets look at the different forms of RAID in use today.

RAID: The Details

RAID Type: Concatenation

Concatenations are also know as "Simple" RAIDs. A Concatenation is a collection of disks that are "welded" together. Data in a concatenation is layed across the disks in a linear fashion from on disk to the next. So if we've got 3 9G (gig) disks that are made into a Simple RAID, we'll end up with a single 27G virtual disk (volume). When you write data to the disk you'll write to the first disk, and you'll keep writing your data to the first disk until it's full, then you'll start writing to the second disk, and so on. All this is done by the Volume Manager, which is "keeper of the RAID". Concatenation is the cornerstone of RAID.

Now, do you see the problem with this type of RAID? Because we're writing data linearly across the disks, if we only have 7G of data on our RAID we're only using the first disk! The 2 other disks are just sitting there bored and useless. This sucks. We got the big disk we wanted, but it's not any better than a normal disk drive you can buy off the shelves in terms of performance. There has got to be a better way..........

RAID Type: Striping (RAID-0)

Striping is similar to Concatenation because it will turn a bunch of little disks into a big single virtual disk (volume), but the difference here is that when we write data we write it across ALL the disks. So, when we need to read or write data we're moving really really fast, in fact faster than any one disk could move. There are 2 things to know about RAID-0, they are: stripe width, and columns. They sound scary, but they're totally sweet, let me show you. So, if we're going to read and write across multiple disks in our RAID we need an organized way to go about it. First, we'll have to agree on how much data should be written to a disk before moving to the next; we call that our "stripe width". Then we'll need far kooler term for each disk, a term that allows us to visualize our new RAID better..... "column" sounds kool! Alright, so each disk is a "column" and the amount of data we put on each "column" before moving to the next is our "stripe width".

Let's solidify this. If we're building a RAID-0 with 4 columns, and a stripe width of 128k, what do I have? It might look something like this:

Look good? So, when we start writing to our new RAID, we'll write the first 128k to the first column, then the next 128k to the second column, then the next 128k to the third column, then the next 128k to the fourth column, THEN the next 128k to the first column, and keep going till all the data is written. See? If we were writing a 1M file we'd wrap that one file around all 4 disks almost 3 times! Can you see now where our speed up comes from? SCSI drives can write data at about (depending on what type of drive and what type of SCSI) 20M/s. On our Striped RAID we'd be writing at 80M/s! Kool huh!?

But, now we've got ANOTHER problem. In a Simple RAID if we had, say, 3 9G disks, we'd have 27G of data. Now, if I only wrote 9G of data to that RAID and the third disk died, so what, there is no data on it. (See where I'm going with this?) We'd only be using one of our three disks in a simple. BUT, in a Striped RAID, we could write only 10M of data to the RAID, but if even ONE disk failed, the whole thing would be trash because we wrote it on ALL of the disks. So, how do we solve this one?

RAID Type: Mirroring (RAID-1)

Mirroring isn't actually a "RAID" like the other forms, but it's a critical component to RAID, so it was honored by being given it's own number. The concept is to create a separate RAID (Simple or RAID0) that is used to duplicate an existing RAID. So, it's literally a mirror image of your RAID. This is done so that if a disk crashes in your RAID the mirror will take over. If one RAID crashes, then the other RAID takes its place. Simple, right?

There's not much to it. However, there is a new problem! This is expensive... really expensive. Let's say you wanted a 27G RAID. So you bought 3 9G drives. In order to mirror it you'll need to buy 3 more 9G drives. If you ever get depressed you'll start thinking: "You know, I just shelled out $400 for 3 more drives, and I don't even get more usable space!". Well, in this industry we all get depressed a lot so, they thought of another kool idea for a RAID......

RAID Type: Stripping plus Mirroring (RAID-0+1)

When we talk about mirroring (RAID-1) we're not explicitly specifying whether we're mirroring a Simple RAID or a Striped (RAID-0) RAID. RAID-0+1 is a term used to explicitly say that we're mirroring a Striped RAID. The only thing you need to know about it is this...

A mirror is nothing more that another RAID identical to the RAID we're trying to protect. So when we build a mirror we'll need the mirror to be the same type of RAID as the original RAID. If the RAID we want to mirror is a Simple RAID, our mirror then will be a Simple RAID. If we want to mirror a Striped RAID, then we'll want another Striped RAID to mirror the first. Right? So, if you say to me, we're building a RAID-0+1, I know that we're going to mirror a Striped RAID, and the mirror itself is going to be striped as well.

You'll see this term used more often than "RAID-1" simply because a mirror, in and of itself, isn't useful. Again, it's not really a "RAID" in the sense that we mean to use the word.

RAID Type: RAID-5 (Striping with Parity)

RAID-5 is the ideal solution for maximizing disk space and disk redundancy. It's like Striping (RAID-0) in the fact that we have columns and stripe widths, but when we write data two interesting things happen: the data is written to multiple disks at the same time, and parity is written with the data.

Okey, let's break it down a bit. Let's say we build a RAID-5 out of 4 9G drives. So we'll have 4 columns, and lets say our stripe width is 128k again. The first 128k is written on disks one, two AND three. At the same time it's written a little magic number is written on each disk with the data. That magic number is called the parity. Then, the second 128k of data is written to (watch carefully) disks two, three and four. Again, a parity number is written with that data. The third 128k of data is written to disks three, four and one. (See, we wrapped around). And data keeps being written like that.

Here's the beauty of it. Each piece of our data is on three different disks in the RAID at the same time! Let's look back at our 4 disk raid. We're working normally, writing along, and then SNAP! Disk 3 fails! Are we worried? Not particularly. Because our data is being written to 3 disks per write instead of just one, the RAID is smart enough to just get the data off the other 2 disks it wrote to! Then, once we replace the bad disk with a new one, the RAID "floods" all the data back onto the disk from the data on the other 2 adjacent disks! But, you ask, how does the RAID know it's giving you the correct data? Because of our parity. When the data was written to disk(s) that parity was written with it. We (actually the computer does this automatically) just look at the data on disks 2 and 4, then compare (XOR) the parity written with the data and if the parity checks out, we know the data is good. Kool huh?

Now, as you might expect, this isn't perfect either. Why? Okey, number 1, remember that parity that saves our butt and makes sure our data is good? Well, as you might expect the systems CPU has to calculate that, which isn't hard but we're still wasting CPU cycles for the RAID, which means if the system is really loaded we may need to (eek!) wait. This is the "performance hit" you'll hear people talk about. Also, we're writing to 3 disks at a time for the SAME data, which means we're using up I/O bandwidth and not getting a real boost out of it.

RAID Comparison: RAID0+1 vs RAID5

There are battles fought in the storage arena, much like the old UNIX vs NT battles. We tend to fight over RAID0+1 vs RAID5. The fact is that RAID5 is advantageous because we use less disks in the endeavor to provide large amounts of disk space, while still having protection. All that means is that RAID5 is inexpensive compared to RAID0+1 where we'll need double the amount of disk we expect to use, because we'll only need a third more disks rather than twice as many. But, then RAID5 is also slower than RAID0+1 because of that damned parity. If you really want speed, you'll need to bite the bullet and use RAID0+1 because even though you need more disks, you don't need to calculate anything, you just dump the data to the disks. In my estimates (this isn't scientific, just what I've noticed by experience) RAID0+1 is about 20%-30% faster than RAID5.

Now, in the real world, you rarely have much choice, and the way to go is clear. If you're given 10 9G disks and are told to create a 60G RAID, and you can't buy more disks, you'll need to either go RAID5, or be unprotected. However, if you've got thoughs same disks and they only want 36G RAID you can go RAID0+1, with the only drawback that they won't have much room to grow. It's all up to you as an admin, but always take growth into account. Look at what you've got, downtime availability to grow when needed, budget, performance needs, etc, etc, etc. Welcome to the world of capacity planning!

RAID History: The Lost Brothers

Wondering what ever happened to RAID-2, RAID-3, and RAID-4? You can look in history books for the details, but they were to be hybrids of mirroring and striping. Ways to include a parity with the data, for protection, but still staying away from mirroring each disks in a normal "one-to-one" mirror. One RAID type would have problems, so they would build another. RAID5, if you hadn't guessed, was the agreed upon solution. RAID-2 and RAID-3 died and burned and scattered into the sea of obsolescence. However, RAID-4 found a home with our friends at NetApp (www.netapp.com).

A RAID-4 volume is made up of one or more data disks which are stripped, and a dedicated parity disk which maintains the write checksums of the data written on each stripe. Checksums are just numbers, so they are very small and quick to write. The problem is that generally when you write data to the volume you write a stripe, then write parity, then write the stripe, then write parity, so on and so forth, but you have to wait for the parity to complete writting before writting the next stripe which is a bottleneck. Add to that from the avaliblity side of things, if you loose your parity disk due to failure your running with your pants down. Sure you can rebuild the parity disk after hot-swapping or replacing the disk, but that requires re-computing parity from each stripe which is an obviously time consuming proposition. NetApp however worked around these problems by using the Filers onboard memory for caching writes, and additionally an NVRAM as a databackup in case of a power failure. It gets the writes ready in the cache, then when it's ready it writes out the data stripes first, and then the parity because it's already calculated parity and the write pattern in memory. In this way we take away the bottleneck of parity disk. This is possible due to the intellegence of WALF, the NetApp OnTap File System.

NetApp has added a new feature in OnTap 6.5, actually, called RAID-DP, Double Parity. It's the same RAID-4 system but it employs mirror parity disks. This way you can loose 2 disks in a volume (assuming one is a parity disk) and keep running.

(It's unfortunate, but NetApp Filer's are the worlds fastest NFS servers. But one day Sun is going to kick their .... never mind.)

Conclusion: Closing Notes

Hopefully this all helps a bit. Even if you didn't know anything about RAID, you should now understand the different types of RAID and be able to make decisions about what you can do with one and in which cases each are useful. This is only the very, very beginning. Now it's time to talk Volume Managers, the mechanism by which all this comes together and works for us. In our next course, we'll learn about the Veritas Volume Manager in particular. Be aware that Veritas isn't the only Volume Manager out there, there is also Sun Solstice DiskSuite, Sun RAID Manager, and others that I don't even know the names of.

This isn't enough information, though. When you choose a volume manager, the documentation will almost always talk about RAID types in detail. This course you've just read is simply a different style of explanation which I've found is better than most others. Here are some links to other places that explain RAID concepts:

- SystemLogic wrote a great tutorial on RAID entitled *RAID: An In-Depth Guide To RAID Technology*. Find it here: http://www.systemlogic.net/articles/01/1/raid/.

- The RAID Advisory Board (RAB) a good source of information. For all I can tell the organization is dead (their phone number is wrong, they haven't done anything in years) but they still have some good stuff. You can find their page here: http://www.raid-advisory.com/.

  Most notably, they published an excellent book called *The RAID Book: A Storage System Technology Handbook* which should be on every storage admins shelf. Be aware that the last edition of this book is the 6th Edition, someone published a book and called it the 7th Edition RAID Book, which was not by the RAB. This book is hard to find, but generally Amazon can track down a copy.

- *A Case for Redundant Arrays of Inexpensive Disks (RAID)* [http://www-2.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf] [PDF] by David Patterson, et al. This is THE paper on RAID. The one that started it all. This paper should be read, if for no other reason, as a historical document.