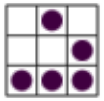

Aaron Toponce

{ 2013 12 18 }

ZFS Administration, Appendix D- The True Cost Of Deduplication

Table of Contents

This post gets filed under the "budget and planning" part of systems administration. When planning out your ZFS storage pool, you will need to make decision about space efficiency, and the cost required to build out that architecture. We've heard over and over that ZFS block deduplication is expensive, and I've even mentioned it on this blog, but how expensive is it really? What are we looking at out of pocket? That's what this post is about. We'll look at it from two perspectives- enterprise hardware and commodity hardware. We should be able to make some decent conclusions after looking into it.

We're only going to address storage, not total cost which would include interconnects, board, CPU, etc. Those costs can be so variable, it can make this post rather complicated. So, let's stick with the basics. We're going to define enterprise hardware as 15k SAS drives and SLC SSDs, and we'll define commodity hardware as 7200 SATA drives and MLC SSDs. In both cases, we'll stick with high quality ECC DDR3 RAM modules. We'll use a base ZFS pool of 10TB.

So, without further ado, let's begin.

Determining Disk Needs

Before we go off purchasing hardware, we'll need to know what we're looking at for deduplication, and if it's a good fit for our data needs. This can be a hard puzzle to solve, without actually storing all the data in the pool, and seeing when you end up. However, here are a few ideas for coming to that solution (the "three S tests"):

1. **Sample Test**: Get a good representative sample of your data. You don't need a lot. Maybe 1/5 of the full data. Just something that represents what will actually be stored. This will be the most accurate test, provided you get a good sample- the more, the better. Store that sample on the deduplicated pool, and see where you end up with your dedupratio.
2. **Simulation Test**: This will be less accurate than the sample test above, but it can still give a good idea of what you'll be looking at. Run the "zfs -S" command, and see where the cards fall. This will take some time, and may stress your pool, so run this command off hours, if you must do it on a production pool. It won't actually deduplicate your data, just simulate it. Here is actual an actual simulation histogram from my personal ZFS production servers:

```
# zdb -S
Simulated DDT histogram:

bucket              allocated                       referenced
______   ______________________________   ______________________________
refcnt   blocks   LSIZE   PSIZE   DSIZE   blocks   LSIZE   PSIZE   DSIZE
------   ------   -----   -----   -----   ------   -----   -----   -----
     1    5.23M    629G    484G    486G    5.23M    629G    484G    486G
     2     860K   97.4G   86.3G   86.6G    1.85M    215G    190G    190G
     4    47.6K   4.18G   3.03G   3.05G     227K   19.7G   14.2G   14.3G
     8    11.8K    931M    496M    504M     109K   8.49G   4.25G   4.33G
    16    3.89K    306M   64.3M   68.3M    81.8K   6.64G   1.29G   1.37G
```

```
    32    5.85K     499M     116M     122M     238K    17.9G    4.64G    4.86G
    64    1.28K    43.7M    20.0M    21.0M     115K    3.74G    1.69G    1.79G
   128    2.60K    50.2M    20.0M    22.0M     501K    9.22G    3.62G    3.99G
   256      526    6.61M    3.18M    3.62M     163K    1.94G     946M    1.06G
   512      265    3.25M    2.02M    2.19M     203K    2.67G    1.72G    1.86G
    1K      134    1.41M     628K     720K     185K    2.13G     912M    1.02G
    2K       75    1.16M     188K     244K     222K    3.37G     550M     716M
    4K       51     127K    85.5K     125K     254K     657M     450M     650M
    8K        2       1K       1K    2.46K    26.7K    13.3M    13.3M    32.8M
   16K        1      512      512    1.94K    31.3K    15.6M    15.6M    60.7M
 Total    6.15M     732G     574G     576G    9.38M     920G     708G     712G

dedup = 1.24, compress = 1.30, copies = 1.01, dedup * compress / copies = 1.60
```

3. **Supposed Test**: Basically, just guess. It's by far the least accurate of our testing, but you might understand your data better than you think. For example, is this 10TB server going to be a Debian or RPM package repository? If so, the data is likely highly duplicated, and you could probably get close to 3:1 savings, or better. Maybe this server will store a lot of virtual machine images, in which case the base operating system will be greatly duplicated. Again, your ratios could be very high as a result. But, you know what you are planning on storing, and what to expect.

Now you'll have a deduplication ratio number. In my case, it's 1.24:1. This number will help us "oversubscribe" our storage. In order to determine how much disk to purchase, our equation should be:

```
Savings = Need - (Need / Ratio)
```

With my ratio of 1.24:1, which is running about a dozen virtual machines, rather than purchasing the full 10TB of disk, we really only need to purchase 8TB of disk. This is a realistic expectation. So, I can save purchasing 2TB worth of storage for this setup. The question then becomes whether or not those savings are worth it.

Determining RAM Needs

Ok, now that we know how much disk to purchase, we now need to determine how much RAM to purchase. Already, we know that the deduplication table (DDT) will occupy no more than 25% of installed RAM, by default. This is adjustable with the kernel module, but we'll stick with default for this post. So, we just need to determine how large that 25% is, so we can understand exactly how much RAM will be needed to safely store the ARC without spilling over to spinning platter disk. In order to get a handle on this metric, we have two options:

1. **Counting Blocks**: With the "zdb -b" command, you can count the number of currently used blocks in your pool. As with the "zdb -S" command, this will stress your pool, but it will give you the most accurate picture of what to expect with a deduplication table. Below is an actual counting of block on my production servers:

```
# zdb -b pool

Traversing all blocks to verify nothing leaked ...

        No leaks (block sum matches space maps exactly)

        bp count:       11975124
```

```
bp logical:    1023913523200      avg:  85503
bp physical:   765382441472       avg:  63914     compression:   1.34
bp allocated:  780946764288       avg:  65214     compression:   1.31
bp deduped:             0    ref>1:      0   deduplication:   1.00
SPA allocated: 780946764288     used: 39.19%
```

In this case, I have 11975124 used blocks, and my 2 TB pool is 39.19% full, or 784GB. Thus, each block is about 70KB in size. You might see something different. [According to Oracle](), each deduplicated block will occupy about 320 bytes in RAM. Thus, 2TB divided by 70KB blocks gives a total storage space of about 30,700,000 total blocks. 30,700,000 blocks multiplied by 320 bytes, is 9,824,000,000 bytes, or 9.8GB of RAM for the DDT. Because the DDT is no more than 25% of ARC, and the ARC is typically 25% of RAM, I need at least 156.8GB, or basically 160GB of installed RAM to prevent the DDT from spilling to spinning platter disk.

2. **Rule of Thumb**: This is our "rule of thumb" rule that you've read in this series, and elsewhere on the Internet. The rule is to assign 5GB of RAM for every 1TB of disk. This ratio comes from the fact that a deduplicated block seems to occupy about 320 bytes of storage in RAM, and your blocks could occupy anwhere between 512 bytes to 128KB, usually averaging about 64KB in size. So, the ratio sits around 1:208, which is where we come up with the "5GB RAM per 1TB disk" metric. So with a 10TB pool, we can expect to need 50GB of RAM for the DDT, or 200GB of RAM for the ARC.

In both cases, these RAM installations might just be physically or cost prohibitive. In my servers, the motherboards do not allow for more than 32GB of physically installed RAM modules. So 40GB isn't doable. As such, is deduplication out of the question? Not necessarily. If you have a

fast SSD, something capable of 100k IOPS, or roughly the equivalent of your RAM install, then you can let the DDT spill out of RAM onto the L2ARC, and performance will not be impacted. A 256GB SSD is much more practical than 200GB of physical RAM modules, both in terms of physical limitations and cost prohibition.

Enterprise Hardware

Without SSD

15k SAS drives don't come cheap. Currently, the Seagate Cheetah drives go for about $1 per 3GB, or about $330 per 1TB. So, for the 8TB we would be spending on disk, we would be spending about $2600 for disk. We already determined that we need about 200GB of space for the ARC. If we need to fit everything in RAM, and our motherboard will support the install size, then ECC registered RAM goes for about $320 per 16GB (how convenient). I'll need at least 14 sticks of 16GB RAM modules. This would put my RAM cost at about $4480. Thus my total bill for storage only would be $7080. I'm only saving $670 by not purchasing 2 disks to save on deduplication.

With SSD

Rather than purchasing 14 16GB memory modules, we could easily purchase an enterprise 256GB fast SLC SSD for about $500. A 256GB SSD is attractive, because as an L2ARC, it will be storing more than just the DDT, but other cached pages from disk. The SSD could also be partitioned to store the ZIL, acting as a SLOG. So, we'll only need maybe 16GB installed RAM (2x8GB modules for dual channel), which would put our RAM cost at $320, our SSD cost at $500 and our drive cost at $2600, or $3420 for the total setup. This is half of the initial price

using only ECC RAM to fit the DDT. That's significant, IMO. Again, I only saved $670 by not purchasing 2 disks.

Commodity Hardware

Without SSD

7200 SATA drives come cheap these days. ZFS was designed with commodity disk in mind, knowing it's full of failures and silent data corruption. I can purchase a single 2TB disk for $80 right now, brand new. Four of those put my total cost at $320. However, the ECC RAM doesn't change, and if I needed 14 of the 16GB sticks as with my enterprise setup, then I can count on my total cost for this commodity setup at $4800. But, a RAM install of that size does not make sense for a "commodity" setup, so let's reduce the RAM footprint, and add an SSD.

With SSD

A fast commodity SSD puts us at the MLC SSDs. The 256GB Samsung 840 Pro is going for $180 right now, and can sustain 100k IOPS, which could possibly rival your DDR3 RAM. So, again sticking with 4 2TB drives at $320, 16GB of RAM at $320 and our Samsung SSD at $180 our total cost for this setup is $820, only saving $80 by not purchasing an additional 2TB SATA drive. This is by far the most cost effective solution.

Additional Hidden Costs & SSD Performance Considerations

When we made these plans on purchasing RAM, we were only considering the cost of storing the ARC and the DDT. We were not considering that your operating system will still need room outside of the ARC to operate. Most ZFS administrators I know won't give more than 25% of RAM to the ARC, on memory intensive setups, and no more than 50% on less memory intensive setups. So, for our 200GB ARC requirement, it may be as much as 400GB of RAM, or even 800GB. I have yet to administer a server with that sort of RAM install. So, SSDs all of the sudden become MUCH more attractive.

If you decide to go the route of an SSD for an L2ARC, you need to make sure that it performs on par with your installed RAM, otherwise you'll see a performance hit when doing lookups in your DDT. It's expected for DDR3 RAM to have a rate of 100k to 150k sustained sequential read/write IOPS. Getting an SSD to perform similarly means getting the high end SATA connected SSDs, or low end PCIe connected, such as the OCZ RevoDrive.

However, suppose you don't purchase an SSD that performs equally with your DDR3 modules. Suppose your DDR3 sustains 100k IOPS, but your SSD only does 20k IOPS. That's 5x as slow as DDR3 (spinning 7200 RPM disk only sustains about 100 IOPS). With as frequently as ZFS will be doing DDT lookups, this is a SIGNIFICANT performance hit. So, it's critical that your L2ARC can match the same bandwidth as your RAM.

Further, there's a hidden cost with SSDs, and that's reliability. Typical enterprise SLC SSDs can endure about 10k write cycles, with wear leveling, before the chips begin to wear down. However, for commodity, more "consumer grade" SSDs, they will only sustain about 3k-5k write cycles. Don't fool yourself though. For our 256GB SSD, this means you can write 256GB 10,000 times, or 2.56PB worth of data on a SLC SSD, or 256GB 3,000 times, or 768TB on an MLC SSD. That's a lot of writing, assuming again, that the SSDs have wear leveling algorithms on

board. But, the SSD may fail early, which means the DDT spilling to disk, and completely killing performance of the pool. By putting a portion of the DDT on the SSD, the L2ARC becomes much more write intensive, as ZFS expands the DDT table for new deduplicated blocks. Without deduplication, the L2ARC is less write intensive, and should be very read intensive. So, by not using deduplication, you can lengthen the life of the SSD.

Conclusion

So, now when you approach the CFO with your hardware quote, with a dedup ratio of 1.24:1, it's going to be a hard sell. With commodity hardware using an SSD, you're getting close to your savings in disk (10.25:1), as compared to enterprise hardware where you're spending much, much more to get close to those space savings (5.10:1). But, with commodity hardware, you're spending 1/4 of the enterprise equivalent. In my opinion, it's still too costly.

However, if you can get your ratio close to 2:1, or better, then it may be a good fit. You really need to know your data, and you really need to be able to demonstrate that you will get solid ratios. A storage server of virtual machines might be a good fit, or where you have redundant data that is nearly exactly the same. For general purpose storage, especially with a ratio of 1.24:1, it doesn't seem to be worth it. However, you're not out of luck, if you wish to save disk.

For good space savings on your disk, that you get nearly for free, I strongly encourage compression. Compression doesn't tax the CPU, even for heavy workloads, provides similar space savings (in my example above, I am getting a 1.3:1 compression ratio versus 1.24:1 dedup ratio), doesn't require an expensive DDT, and actually provides enhanced performance. The extra performance comes from the fact that highly compressable data does not need to physically write

as much data to slow spinning platter, and also does not need to read as much physical disk. Your spinning disk is the slowest bottleneck in your infrastructure, so anything you can do to optimize the reads and writes could provide large gains. Compression wins here.

Hopefully, this post helps you analyze your deduplication plans, and identify the necessary costs.

Posted by Aaron Toponce on Wednesday, December 18, 2013, at 2:09 am. Filed under Debian, Linux, Ubuntu, ZFS. Follow any responses to this post with its comments RSS feed. You can post a comment or trackback from your blog. For IM, Email or Microblogs, here is the Shortlink.

{ 9 } Comments

1. Gavin | December 18, 2013 at 2:16 am | Permalink

   "zfs -S" in your simulation test should actually be "zdb -S". dedup simulation is a feature of the ZFS debugger rather than the standard ZFS command set.

2. Aaron Toponce | December 18, 2013 at 7:43 am | Permalink