ZFS: To Dedupe or not to Dedupe...

...that is the question.

Ever since the introduction of deduplication into ZFS, users have been divided into two camps: One side enthusiastically adopted deduplication as a way to save storage space, while the other remained skeptical, pointing out that dedupe has a cost, and that it may not be always the best option.

Let's look a little deeper into the benefits of ZFS deduplication as well as the cost, because ultimately it boils down to running a cost/benefit analysis of ZFS deduplication. It's that simple.

ZFS Deduplication: What Value Do You Get?

ZFS dedupe will discard any data block that is identical to an already written block, while keeping a reference so that it can always reproduce the same block when read. You can read more about ZFS deduplication and how it works here.

Before you decide to use deduplication, it's good to know what value you'll get out of it. Here are a few options to figure out how much space you'll save as a result of using ZFS deduplication:

- **Test it** with some real data. This is the most accurate and straightforward option: Set up a test pool, enable ZFS deduplication on it, then copy a representative amount of the data you are considering onto it. Then use `zpool list` and look at the `DEDUP` column for the deduplication ratio. The important thing here is to use a *representative* amount of data, so you get an accurate estimate of how much savings to expect.

- **Simulate it** by applying the `zdb -S` command to an existing pool with the data you want to deduplicate. This option is less accurate than using real data with a real deduped zpool, but it can provide you with a ballpark estimate based on your existing data.

- **Guess it**, based on the knowledge you have of your data. Not the best option, but sometimes, setting up a test pool is not feasible and simulating dedupe on existing data doesn't work because you simple don't have any data to analyze with. For example, if you plan to run a storage server for virtual machines: How many machines do you support? How often are they patched? How likely will people apply the same software/patches/data to your

> machines? How many GB of dedupe-able data is this likely to generate? Can you come up with a representative test case after all to make the guess less guessy?

In any case, you'll end up having an expected deduplication ratio for the data: For every GB of data you actually store, how many GBs of retrievable data will you get? This number can have any value: Some people see a value of 1.00 (no duplicates whatsoever), others see some moderate savings like 1.5 (store 2 GB, get one free), and some very lucky people can see as much as 20x, for example a virtualization storage server with a very repetitive usage profile.

Now take your total amount of storage and divide it by the dedup ratio, then subtract the result from your total amount of storage. That is your expected storage savings as a result of deduplication:

Total Storage - (Total Storage / Expected Dedupe ratio) = Expected Storage Savings

As a fictional example, let's assume that we're looking at a 10 TB storage pool to be used for storing virtual machine images in a virtual desktop scenario. In a quick test, we set up a 1 TB pool and copy some existing VM data to it, which yielded a dedup ratio of 2. This means that we only need about 5 TB of capacity to provide the 10 TB of data thanks to deduplication, hence we would save 5 TB of disk storage.

Let's assume that the average cost of 1 TB of disk (including controller, enterprise class drives, etc.) is at $1000 for the sake of simplicity: Then dedup would save us $5000 in this particular example.

So what do we need to *spend* in order to realize these cost savings?

ZFS Dedupe: The Cost

Saving space through deduplication doesn't come for free. There is a cost. In the case of ZFS, it's memory: ZFS keeps a dedup table in which it stores all the checksums of all the blocks that were written after deduplication was enabled. When writing new blocks, it uses this table to determine whether a block has been written yet, or not.

Over time, the table becomes larger and larger, and since every write operation has to use it, it should be kept in main memory to avoid unnecessary extra reads from disk. To be clear: ZFS can work perfectly well even if the table is not in memory. But the bigger the deduplication table grows, the slower write performance will become, as more and more writes trigger more and more extra reads for dedup table lookups.

How much memory does one need to keep the ZFS dedup table in memory, and hence your system happy?

According to the ZFS dedup FAQ, each entry in the dedup table costs about 320 Bytes of memory per block. To estimate the size of the dedup table, we need to know *how many blocks* ZFS will need to store our data. This question can be tricky: ZFS uses a variable block size between 512 bytes and 128K, depending on the size of the files it stores. So we can't really know in advance how many blocks ZFS will use for storing our data.

If we mainly store *large* files (think videos, photos, etc.), then the average block size will be closer to 128K, if we store *small* files (source code, emails, other data), we'll probably be closer to a few K. Here are some ways to find out for sure:

Option 1: Count Your Blocks With ZDB

The most accurate way to determine the number of blocks is to use the `zdb -b <poolname>` command, which will print out detailed block statistics. But beware: This command may take a long time to complete as it will scan all of the metadata blocks in your pool:

```
  constant@walkuere:~$ zdb -b oracle

  Traversing all blocks to verify nothing leaked ...

    No leaks (block sum matches space maps exactly)

    bp count:          306575
    bp logical:    19590419968      avg:  63900
    bp physical:   17818332160      avg:  58120
compression:   1.10
    bp allocated:  17891765760      avg:  58360
compression:   1.09
    bp deduped:             0    ref>1:      0
deduplication:   1.00
    SPA allocated: 17891765760     used: 52.48%
```

The number to look for here is `bp count` which is the number of block pointers, hence the number of blocks in the pool.

In this example, the dedup table would take up 306575 blocks * 320 bytes, which yields around 100 MB. This file system has a size of 32GB, so we can assume that the dedup table can only grow to about 200 MB, since it's about 50% full now.

Option 2: Estimate Your Average Block Size, Then Divide

If you don't have your data in a ZFS pool already, you'll have to guess what your average block size is, then divide your expected storage capacity by the average block size to arrive at an estimated number of ZFS blocks. For most cases of mixed data like user home directories, etc., we can assume an average block size of 64K, which is in the middle between the minimum of 512 bytes and 128K. This works reasonably well for my own example:

```
  constant@walkuere:~$ zpool list oracle
  NAME     SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
  oracle  31.8G  16.7G  15.1G  52%  1.00x  ONLINE  -
```

we see that the average block size of my work pool is 16.7G divided by 306575 which yields about 54K. Your mileage will vary, especially if you use different kinds of data such as VM images, databases, pictures, etc.

Example: If a zpool stores 5 TB of data at an average block size of 64K, then 5TB divided by 64K yields 78125000 blocks. Multiplied by 320 Bytes, we get a dedup table size of 25 GB!

The Total RAM Cost of Deduplication

But knowing the size of your deduplication table is not enough: ZFS needs to store more than just the dedup table in memory, such as other metadata and of course cached block data. There's a limit to how much of the ZFS ARC cache can be allocated for metadata (and the dedup table falls under this category), and it is capped at **1/4 the size of the ARC**.

In other words: Whatever your estimated dedup table size is, you'll need at least four times that many in RAM, if you want to keep all of your dedup table in RAM. Plus any extra RAM you want to devote to other metadata, such as block pointers and other data structures so ZFS doesn't have to figure out the path through the on-pool data structure for every block it wants to access.

RAM Rules of Thumb

If this is all too complicated for you, then let's try to find a few rules of thumb:

- For every TB of pool data, you should expect 5 GB of dedup table data, assuming an average block size of 64K.
- This means you should plan for at least 20GB of system RAM per TB of pool data, if you want to keep the dedup table in RAM, plus any extra memory for other metadata, plus an extra GB for the OS.

The Alternative: L2ARC

So far, we have assumed that you want to keep *all* of your dedup table in RAM at all times, for maximum ZFS performance. Given the potentially large amount of RAM that this can mean, it is worth exploring some alternatives.

Fortunately, ZFS allows the use of SSDs as a second level cache for its RAM-based ARC cache. Such SSDs are then called "L2ARC". If the RAM capacity of the system is not big enough to hold all of the data that ZFS would like to keep cached (including metadata and hence the dedup table), then it will spill over some of this data to the L2ARC device. This is a good alternative: When writing new data, it's still much faster to consult the SSD based L2ARC for determining if a block is a duplicate, than having to go to slow, rotating disks.

So, for deduplicated installations that are not performance-critical, using an SSD as an L2ARC instead of pumping up the RAM can be a good choice. And you can mix both approaches, too.

Adding up the Dedup Cost

Back to our example: Our 10 TB pool is expected to just utilize 5 TB which, at an average block size of 64K would need a dedup table that is approximately 25GB in size. Last time I checked, an enterprise-class SSD with 32GB was in the range of $400, while 25GB of Memory was in the range of $1000. So this is the range of what using deduplication will actually cost in terms of extra SSD and/or RAM needed.

Putting Together the Business Case

There you have it: For our fictitious example of a 10 TB storage pool for VDI with an expected dedup savings of 5 TB which translates into $5000 in disk space saved, we'd need to invest in $400 worth of SSD or better $4000 of RAM. That still leaves us with at least $1000 in net savings which means that in this case dedup is a (close) winner!

But if we assume the same amount of raw data (10 TB) but only a dedup savings factor of 1.1, then our equation would be different: We'd still save close to 1 GB of disk storage (ca. $1000) but we'd need to build up a dedup table that can manage 9 TB of data, which would be in the range of 45GB. That means about $600 in SSD capacity for storing the dedup table. For RAM, we'd need 4 times that amount (180GB), since only 1/4 of ZFS ARC RAM is available for metadata. That doesn't look very attractive to me.

So it really boils down to what you get (= amount of space saved) and what you need to spend in order to get that (= extra SSD and/or RAM for the DDT), depending on whether you want to sacrifice some performance (by going SSD only) or not (by adding enough RAM).

Now, you can do your own calculations on a case by case basis.

Finding the Break-Even Point

Given some standard cost for disk space, SSDs and RAM, you can calculate a break even point. That way, you can just ask: What's the expected dedup savings factor? Then you can instantly decide whether it's worth deduplicating or not.

At our fictitious values of $1000 per TB of disk space, $400 for a 32 GB SSD and $1000 for 24GB of memory, and assuming an average block size of 64K, we can derive two break-even dedup ratios depending on the performance requirements:

- For applications where performance isn't critical, you can get away with no extra RAM for the DDT, but with some extra space for storing the DDT at least on an SSD in L2ARC. Each TB of pool capacity will cost 5GB of dedup table, no matter how much dedup will save. 5GB of dedup table will cost $62.5 when stored in a standard SSD ($400 for 32GB). Hence, for each TB of pool capacity, at least 62.5 GB need to be saved through dedup for the SSD to pay for itself (1000 GB cost $1000, 62.5 GB saved will save 62.5$, the price of having that dedup table stored in SSD). That translates into a minimum dedup factor of 1.0625 to warrant the extra SSD capacity needed.

- For applications that are more performance-sensitive, you'll need the same amount of memory for the DDT per TB (5GB), but this time you want to store it fully in RAM. ZFS limits metadata use in RAM to 1/4 of total ARC size, so we need to make sure our system has at least 20GB of extra RAM per TB of stored data. That means each TB of deduped pool data will cost us approx. $834 in x86 memory for storing its dedup table, so the minimum dedup savings factor needs to be 1.834 here.

Rules of Thumb

These are all fictitious numbers, YMMV, but I think some good rules of thumb are:

- If performance isn't critical, and if you expect to save more than 20% of storage capacity through deduplication, then go for it but add at least 5GB of L2ARC SSD capacity per TB of pool data to store the dedup table in.

- If performance is important, wait until you expect at least a 2:1 reduction in storage space through deduplication, and add 30GB of RAM to your system for every TB of disk capacity to make sure that the dedup table is always in memory so optimal write performance is ensured..

Why the extra 10GB of RAM in the latter rule of thumb? You don't want to fill up ZFS' metadata RAM cache entirely with the dedup table. The other metadata you want to have quick access to are ZFS' pointer data strcutures so it knows where each data block is stored on disk etc. That can be estimated at 1% of total pool capacity which is 10GB per TB of pool data.

Conclusion

The decision to use ZFS deduplication or not is almost always a simple cost/benefit analysis. When using deduplication, one needs to plan for at least some extra L2ARC SSD requirements, or better some extra RAM for storing the dedup table in a manner that doesn't negatively impact write performance.

Especially RAM can become a decisive factor in deciding for or against deduplication, so usually a dedup savings factor of 2 is a necessary threshold for deduplication to become a real cost saver.

Your Take

So here are some fictitious numbers. Did you do your own dedup analysis? What were your results? What do you base your decision to use deduplication on? Place a comment below and share your own dedup cost/benefit analysis cases.

By Constantin Gonzalez , 2011-07-27, updated: 2017-10-03 in Solaris.

Tweet | Share | G+ +1 | € Flattr

The Solaris Eco-System is Expanding

Solaris 11 Available for Early Adopters

Comments

38 Comments Constant Thinking 1 Login

Recommend 4 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Aaron Toponce • 5 years ago

I know this article is old, and prices have changed. However, I want to just nitpick on the finances in your article. You say:

"Let's assume that the average cost of 1 TB of disk (including controller, enterprise class drives, etc.) is at $1000 for the sake of simplicity: Then dedup would save us $5000 in this particular example."

Assuming "enterprise class drives" means 15k SAS drives, then that works. But, ZFS is designed with commodity storage in mind. Further, it's likely that "enterprise" datacenters with "enterprise storage" aren't designing storage with ZFS in mind. Intsead, they're using EMC or NetApp SANs (I've worked in the "enterprise" industry, and know what these Big Budgets purchase).

I would be willing to bet that most shops that do actually deploy large ZFS storage pools, are either using Sun/Oracle hardware, with a sales representative, or they're a SMB with Supermicro commodity x86 hardware, and they're deploying SATA disk. There are many

see more

3 • Reply • Share ›

Constantin Mod Aaron Toponce • 5 years ago

Hi Aaron,

thanks for your thoughtful comment.

I agree: The economics of storage are changing all the time and it’s always useful to assess whether feature X is really worth the effort. And yes, as CPU power becomes cheaper and more abundant vs. the scarcity of IOPS, compression is almost always a win.

The need to dedup is really a function of the cost of storing data vs. the amount of new data that is coming in that you don’t want to ignore/sacrifice/delete. The changes in technology and cost are just one side of the equation – the availability and the value of acquiring new data are the other side.

For businesses with a stagnant or not dramatically increasing volume of data, dedupe is becoming less interesting as the cost of storage goes down.

But there are also businesses with a very large amount of data to store. And then it makes sense to consider dedupe, beyond just mere compression.

Cheers,
Constantin

• Reply • Share ›

Ludovic Urbain • 6 years ago

Nice article - I think it's worth mentioning that only RAM will help your dedupe speed if you're already running an all-flash array, and also that dedupe factors (and collisions) are directly related to pool size (I don't remember the probability details but it's better than linear by nature).

1 • Reply • Share ›

Constantin Mod Ludovic Urbain • 6 years ago

i Ludovic,

thank you for your comment!

Yes, only RAM helps if your pool is Flash-only, but then you're very lucky to not depend on rotating rust at all :).

The dedupe factor is dependent of the amount of redundancy in the pool (the number of equal blocks vs. the number of unique blocks), which for statistical reasons improves as the pool size grows. But it really depends on the nature of the data and how likely the use case is going to generate duplicate blocks, so there's no real rule of thumb other than to measure the dedupe ratio with an as represantative subset of data as possible.

Cheers,
Constantin

• Reply • Share ›

Ludovic Urbain ➦ Constantin • 6 years ago

I don't think you can measure a dedupe ratio with a subset - that's the very nature of dedupe ;)

Also one needs to be careful about "use cases" as many seem to take the general word that databases don't get good dedupe/compression, which is blatantly false (just all those 15000 users with the "1234" pin code are a good example ;).

1 • Reply • Share ›

George ➦ Ludovic Urbain • 6 years ago

Ahem, I should hope that dedup would not be useful in this case, because you're using individual user password salts aren't you?

2 • Reply • Share ›

Ludovic Urbain ➦ George • 6 years ago

And what exactly makes you think "individual user password salts" are *such* a good idea ?

If you're referring to the captain hindsights discussing the database leaks, you should by now be aware that the problem lies not with the salts but with the lack of security around backups.

Furthermore, do you *really* think 512-byte dedupe (standard) is going to dedupe BLOCKS that contain two passwords (often 256 byte hashes), no matter the hashing method ?

Lastly, compression, which imho is _also_ or without hashes and at any level of cryptography.

see more

^ | ˅ • Reply • Share ›

Gobs ➦ Ludovic Urbain • 6 years ago

That's not a responsible way to handle your user's security. You should protect your data and backups, sure, but you should also make sure that an attacker can't get a usable password list out of a database dump. This is to protect the users who reused their passwords elsewhere.

tl;dr: defense in depth

1 ^ | ˅ • Reply • Share ›

Ludovic Urbain ➦ Gobs • 6 years ago

That's utter bullshit.
defense in depth ?

So on one side, you're going to generate semi-random individual salts to protect from possible db leaks, which you will store in the same database, thereby nullifying most of your *tactical* advantage.

And on the other side, your system has so many security flaws database backups get leaked ?

I think very much that individual salts are completely overkill for people who use SSL (lol) as only security, and have security holes big enough to leak db backups.

^ | ˅ • Reply • Share ›

Constantin Mod ➦ Ludovic Urbain
• 6 years ago

Guys, stay cool. This is a ZFS article, not a security one. Go discuss this on a security forum or better yet, in private.

Thanks :).

Constantin

1 ^ | ⌄ • Reply • Share ›

George ➦ Ludovic Urbain • 6 years ago

This is an incredibly irresponsible and naive view of user credentials management. You seem to be arguing that if we consider the protection worthwhile in the first place, we are admitting we're insecure enough to get dumped.

Yes, we'll also do everything in our power to prevent that. But in the cases where it happens (and you must KNOW it's not possible to GUARANTEE you won't get your db leaked), at least you can tell your users - don't worry, your passwords are secure and there is (virtually) no risk they will decipher them and pair them up with your e-mail address, to, say your PayPal account.

Sorry Constantin but this guy might be managing a db with one of MY passwords on!

^ | ⌄ • Reply • Share ›

Redstar • 7 years ago

Hello,
this article helped me a lot understanding the RAM requirements for high-speed zfs use, especially the part about block size measurement.

Still I think I agree with Olli when it comes to the 1/4 of ARC limitation: If I imagine a system without dedup, then all matadata has to be stored in the given RAM. So when I add RAM in order to accomodate the dedup table, I should be able to increase the amount of metadata in the ARC without a severe penalty: There still is the same amount for other ARC data and the new dedup table sits in the 'new RAM'.

So I think following your advice and adding 4x the expected dedup table size in RAM means that you invest more than you need to which means the break-even point for dedup use might be at lower deduplication multipliers already.

So I would be pleased if you could explain once more, why I want addional RAM for ARC in case I want to use dedup. Nevertheless the article (including your replies to the comments) is a great source of information about dedup.

1 • Reply • Share ›

Constantin Mod Redstar • 7 years ago

Hi Redstar,

thanks for your comment!

Yes, the ARC uses some of its RAM to store the dedup table and some to store metadata. For optimum performance, you want both to fit into RAM. If RAM is expensive (which it usually is not, compared to the cost of not getting to critical data quickly enough), then you can trade off RAM for performance by using less RAM and letting the ARC prioritize which pieces of the dedup table are most likely needed and which metadata it will need next.

So RAM sizing per my requirements assumes you want maximum performance and then you get enough to store both dedup table and metadata. But you can trade off as much as you want, including the use of L2ARC on SSDs.

Once more: If you use dedup, then ZFS needs more data

see more

• Reply • Share ›

77109 • 7 years ago

Hi,
i love your explication. So for each block we have 230 bytes of dedup table but do you know the amount of the rest of metada? To know the real amount of metada by block on zfs and have to calculate the amont of ram.
Thanks

1 • Reply • Share ›

Constantin Mod 77109 • 7 years ago

Hi 77109,

thanks for your comment!

The other metadata that ZFS needs to go through to access a specific block is the sequence of metadata blocks from the uberblock that leads to it. Every metadata-block contains up to 256 pointers to the next level of metadata blocks. If we take the lowest level of metadata blocks before the actual blocks, we're talking about a 256th or less of total blocks devoted to metadata at that level. Each metadata block is stored 3 times for redundancy, which makes around 1.2% of all blocks. We can ignore all upper levels of metadata blocks for simplicity since they'll only influence the number in the order of 1% of the actual percentage.

Assuming that your ZFS file system is not full and assuming that the metadata block sizes are similar to data block sizes, we can assume that total metadata for a ZFS pool is in the

see more

• Reply • Share ›

Olli • 7 years ago

But what if you change your ARC to use more than 1/4 for Metadata Caching? Just set zfs:zfs_arc_meta_limit in /etc/system and you can use much much more RAM for Metadata-L2ARC (in fact we use almost the whole available RAM for this)

1 • Reply • Share ›

Constantin Mod Olli • 7 years ago

Yes, you can tune your way out of spending that much RAM and grow the Metadata piece of ARC - at the expense of caching data blocks. If all of your data read is 100% random, you can do that, but in most cases, there's going to be a good number of data blocks you want to cache in that other 3/4 of your ARC.

• Reply • Share ›

Howadah • 7 years ago

I didn't realize that dedupe gobbled so much ram. Hard-drives are getting cheaper every day too....

Foresee any problems with turning dedupe off after its been on for a while?

1 • Reply • Share ›

Constantin Mod Howadah • 7 years ago

Hi Hayes,

thanks for your comment!

Well, it doesn't necessarily eat RAM: It just grows a significantly large data structure that will compete for ARC space in you limited RAM depending on the frequency it is being used. That said, if the dedup table isn't in RAM due to it being limited and if an app needs to write something, that write can be quite slow. And in situations of higher sustained write load (as in "virtualization" or "file server"), the dedup table can become a bulky cache citizen.

If you turn dedupe off, then the DDT won't be used for writing new data any more, so the ARC will be freed to store other data and metadata.

Cheers,
Constantin

• Reply • Share ›

fblittle ➦ Constantin • 7 years ago

What is the cost of Dedupe in terms of write speed if you use a spinning disk? For large file writes, as in video, what percentage will it slow the writes? Rule of thumb.

Also with reference to the above question:
Foresee any problems with turning dedupe off after its been on for a while?

If I have turned Dedupe on in a pool and it has been running a few months with a ratio of 1.2 or so, If I turn Dedupe off, how will it repopulate my pool with the duplicate data? Will my data be highly fragmented when done?

• Reply • Share ›

Constantin Mod ➦ fblittle • 7 years ago

It's hard to tell, because the cost of dedup is highly individual. Here's what happens over time:
- As the dedup table grows, it starts to compete with other metadata for ARC space.
- As a result, read speeds can suffer as ZFS needs to handle extra metadata reads to access blocks on disks.
- Similarly, writes will trigger extra reads for dedup entries and metadata if RAM becomes scarce

It's not the end of the world, but it is noticeable and I've heard of numerous cases where performance was bad after some time and the reason was simply low RAM and dedup. After turning it off, those cases immediately experienced better performance.

see more

• Reply • Share ›

BoruG ➦ Constantin • 5 years ago

Wonder if this is the case, why these engineers comeup with some way to do the dedupe in the background. zfs try to do this real time and that is not practical. They need to perform dedup in background. Hope someone can comeup with a way to do it.

1 • Reply • Share ›

Ben Pottinger • 5 years ago

I'm a big fan of ZFS but I do have to ask. Why is dedup so much more "expensive" on ZFS when its virtually "free" in windows 2012? It seems like (from the limited amount i've read on the subject) that for just about any scenario where it would be useful you should turn it on in windows 2012, while in ZFS you have to be very very careful when and where you use it.

• Reply • Share ›

Constantin Mod ➦ Ben Pottinger • 5 years ago

Hi Ben,

I don't know how Microsoft handles deduplication, so I can't comment on that. The design of deduplication in ZFS is pretty simple and straightforward because it leverages the checksums that are already there, with little overhead. And it is instant: No after the fact scanning of duplicate blocks. The price is not that high: More RAM and/or an cache device will speed it up and also benefit many other ZFS transactions, but it's not strictly required, though strongly advised. This article is more about how to do your cost/benefit analysis, then decide whether dedupe is a good idea for your particular use-case.

identifying duplicates, managing the deduplicated blocks etc.

Cheers,
Constantin

• Reply • Share ›

Ben Pottinger Constantin • 5 years ago

Ok that leads me into another question: Can dedup be enabled on a per "filesystem" basis? In other words could I make a separate "filesystem" in my pool (since all the "filesystems" will all share the same total space available on the drive) and enable dedupe *only* on that filesystem? If that would work then would it "pay off" to use it on a filesystem just for backups from windows machines? In other words if I ran a full "drive image" (ISO) style backup 3 times a week would I end up saving space or would those ISO files need to be identical or could ZFS dedup them?

Windows 2012 "seems" to offer extremely high dedup rates with almost no memory "costs". Its doing something vastly different then ZFS apparently. I've seen people enable dedup on 4+ TB of data and the machine itself not have more then 8-16GB of ram while here with ZFS we are talking 20GB *per TB* of storage.

Honestly I'll probably just use a dedup enabled backup program like eXdupe or something. :)

• Reply • Share ›

Constantin Mod Ben Pottinger • 5 years ago

Yes, dedupe on ZFS can be enabled on a per filesystem basis.

You can have dedupe on ZFS with 8-16GB of RAM on a 4+ TB pool without any problems, we're just optimizing performance here.

I suggest you just try it out with your use-case and see for yourself.

Dedupe on ZFS is based on blocks, so multiple ISOs of the same or incremental data should dedupe just fine.

Cheers,
Constantin

^ | ˅ • Reply • Share ›

Yawhann Chong • 6 years ago

Great article! While this isn't exactly ZFS related, one really good selling point for a business case is reduced network utilization and job length for off-site dedupe backup. :)

^ | ˅ • Reply • Share ›

Constantin Mod → Yawhann Chong • 6 years ago

Hi Yawhann,

thanks for your comment! Yes, deduplication saves IO everywhere. This is why compressed filesystems are also a plus in most cases: the IOs they save are more significant than the CPU overhead for compression.

Cheers,
Constantin

^ | ˅ • Reply • Share ›

Boris Protopopov • 6 years ago

Hello, Constantin,
interesting discussion; the post refers to the ZFS FAQs for the way to assess the max amount of memory taken by the DDT entries. I think there is a bit of overestimation there:

looking at the code in illumos, for instance, one can see that there are two ways DDT consumes memory:

1) as blocks buffered in ARC
2) as ddt_entry_t in the avl_trees in ddt_t structs

the latter is transient as it is destroyed at every txg processing boundary (spa_sync() call), and the former is limited to 1/4 of ARC size by default, as mentioned in this post. If we were to let the ARC grow unlimited, the size taken in ARC would be:

'on-disk DDT entry size' times 'total number of entries'

The on-disk size is 64 bytes (echo "::sizeof ddt_phys_t" | sudo mdk -

see more

^ | ˅ • Reply • Share ›

Constantin Mod ↪ Boris Protopopov • 6 years ago

Hi Boris,

thanks for sharing, this sounds like a great idea!

Thanks,
Constantin

Reply • Share ›

zhivotnoe • 6 years ago

Great! Thank you for analysis.

Reply • Share ›

patkoscsaba • 7 years ago

I am curious, wasn't the '1/4' limit of zfs_arc_meta_limit removed with the solving of CR# 6957289 - "ARC metadata limit can have serious adverse effect on dedup performance". ?
Reading that bug and some forums, it seems like, the limit was modified to default to the arc_c_max. (https://forums.oracle.com/f...)

Reply • Share ›

Constantin Mod ↪ patkoscsaba • 7 years ago

Hi Patkos,

yes, if you're running a version of Solaris where this Bug has been integrated, then you don't need to worry about the 1/4 ARC limitation of metadata.

Cheers,
Constantin

Reply • Share ›

James Trent • 7 years ago

Great article, do you think it is necessary to use industry level data deduplication software for home use?

Reply • Share ›

Constantin Mod ↪ James Trent • 7 years ago

ZFS offers industry level data deduplication at a fraction of the cost, so it is accessible for home use. So the question boils down to: How can I save more, through buying more disks for my data or for expanding RAM so I can dedup efficiently?

BTW, please refrain from SEO-Linking non-relevant stuff in comments, I removed the link from your post. Nice try.

• Reply • Share ›

Jeff • 7 years ago

If you turn dedupe off for the performance and later turn it back on, will it try (or can you make it) scavenge duped blocks. I guess I am wondering if there would be a reasonable way to schedule high performance during peak usage, and still ultimately get the space savings by doing cleanup during idle time.

• Reply • Share ›

Constantin Mod Jeff • 7 years ago

Hi Jeff,

no, there is no such thing as dedup-after-the-fact in ZFS dedup. The only thing you could do would be to trigger a re-write of the blogs by forcing a re-write of the data.If you expect significant savings from dedupe, then it makes sense to expand your RAM and add an L2ARC SSD for handling the extra dedupe table requirements. That is the most efficient option.

Hope this helps,
Constantin

• Reply • Share ›

Steve Gonczi Constantin • 3 years ago

Hello everybody,

In addition to the drawbacks already mentioned, dedup imposes a performance penalty on the entire zpool where it is turned on.

There are 3 on-disk tables: the singletons, the duplicates (entries with ref count >1) and the dittos. Even of we achieve little or no actual dedup, we still have this speculative singleton table. This has to be consulted in the write path, to see if we can dedup. This imposes a constant extra random read load when writing data, for little or no benefit.

As we delete deduped files, the table does not go away, because it is a zap table. It shrinks a somewhat because of compression and trailing zero elimination, but not significantly. To observe the effect of this: Take a look at the dedup stats via zdb or the zpool

see more

Reply • Share ›

ALSO ON **CONSTANT THINKING**

Iron Sky Update: A Behind the Scenes Video Interview

4 comments • 7 years ago

Max — Dude, they come back in 2018, not 2038 :)

The Rise of Engineered Systems

9 comments • 7 years ago

UX-admin — "The storage doesn't like to play well with the server, which turns out to be a …

The Business Value of Engineered Systems

7 comments • 7 years ago

Constantin — Hi Andrius,try to connect to the U5 over the serial console. Can you see the …

Three Enterprise Architecture Principles for Building Clouds

2 comments • 7 years ago

rahulkp — informative content..!!

Subscribe Add Disqus to your siteAdd DisqusAdd

Disqus' Privacy PolicyPrivacy PolicyPrivacy