Deduplication Optimized Volumes

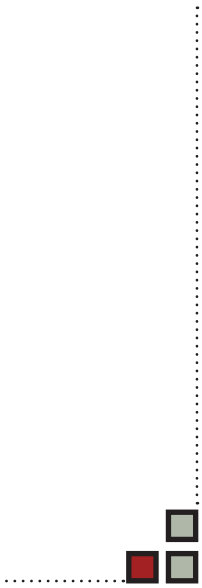
Bacula Communtity Version

This document is intended to provide insight into the considerations and processes required to implement deduplication with Deduplication Optimized Volume backups and Bacula Community Version.

Bacula Systems White Paper

Version 1.8, December 15, 2018

Contents



Executive summary

IT organizations are constantly being challenged to deliver high quality solutions with reduced total cost of ownership. One of those challenges is the growing amount of data to be backed up, together with limited time to run backup jobs (backup window). Bacula offers several ways to tackle these challenges, one of them being *Deduplication Optimized Volumes*, which write Bacula Volumes in a deduplication friendly format. This feature is available with Bacula Community Binaries that will be provided for Bacula version 9.0.0. The Deduplication Optimized Volume layout and use has been patented by Bacula Systems SA. The patent is still pending, consequently during this waiting period Bacula Systems has been advised not to provide source code, but does offer the community the binary plugin. Note: this plugin works only with Bacula Community compiled binaries (actually compiled by Bacula Systems for the community). Bacula Systems offers you a patent waver for use of the plugin provided in binary form exclusively with Bacula.

This document is intended to provide insight into the considerations and processes required to successfully implement this backup technique.

1 Deduplication

Deduplication is a complicated subject. Generally speaking, it permits detecting that data being backed (usually blocks) have already been stored and rather than making an additional backup copy of the same data, the deduplication software just keeps a pointer referencing the previously stored data (block). Detecting that a block has already been stored is done by computing a hash code (also know as signature or fingerprint) of each block, and comparing the hash code with those of blocks already stored.

The picture becomes much more complicated when one considers where the deduplication is done: either on the server and/or on the client machines. In addition, most deduplication is done on a block by block basis, which some deduplication systems permitting variable length blocks and/or blocks that start at arbitrary boundaries rather than on specific alignments (sliding blocks).

Bacula's first step in doing deduplication is to offer an alternate (additional) Volume format that is aligned on specific block boundaries. This permits an underlying file system that does deduplication to efficiently deduplicate this new Bacula Deduplication Optimized Volume format (or often called "Aligned" Volume format. Another way of describing this is that we have filtered out all the metadata and record headers and put them in the Metadata Volume (same as existing Volume format) and put only file data that can be easily deduplicated into the Aligned Volume.

Since there are a number of deduplicating file systems available on Linux systems (ZFS, lessfs, ddumbfs, SDFS (OpenDedup), LiveDFS, ScaleDFS, NetApp (across NFS), Epitome (OpenBSD), Quantum (in their appliance), ..., this Bacula implementation allows the users to choose what deduplication engine they want to use.

Although a number of these dedupe engines use large blocks (128K), the block sizes on most can range from 4K to 1M. Bacula with Deduplication Optimized Volumes has implement several new Storage Device directives that permit tuning what Bacula does to match what is optimal for the underlying deduplication engine.

1.1 Advantages of Deduplication

There are two main advantages of deduplication:

- Deduplication can significantly reduce the disk space needed to store your data. In good cases, it may reduce disk space by half, and in the best cases, it may reduce disk space by a factor or 10 or 20.
- Storing data (backups) can be much faster since a whole block of data can be replaced by a pointer to an existing block that is already stored on disk.

1.2 Cautions About Using Deduplication

Here are a few of the things that you should be aware of before using deduplication techniques.

- Deduplication takes a lot of CPU and memory resources. To do efficient and fast deduplication, you will need lots of CPU power (for computing hash codes), and additional amounts of RAM memory (for fast lookups of hash codes).
- The extra CPU power and memory needed can be expensive, but SSD can largely solve the memory requirements at a reasonable cost.
- Due high CPU and RAM requirements as well as the extra complexity of deduplication, performance tuning is more complicated.
- If your disk develops a bad block instead of damaging one file (that may be stored many times), it may damage all (hundreds or thousands) files that contain the same data. That is you have a single point of failure that can cause more damage than would happen on a non-deduplicated system. These problems can be reduced by using deduplicating systems that checksum everything such as ZFS, or by using hardware or RAID technology.
- Possible loss of data can be mitigated by using the duplicate copy feature of a number of the deduping filesystems (copies in ZFS). In fact some systems such as ZFS permit you to limit the number of references to a single block of data. Once that limit is reached, it will store the block another time. Can minimize any loss due to bad disk blocks.
- Deduplication collisions can cause data corruption. That is if the deduplicating system uses a weak hash code (such as MD5, SHA1, Fletcher, ...), it is possible that several blocks with different data will hash to the same value. If the system does not do a byte for byte comparison, then the second block

stored will be corrupted when it is read back, because it will get the first block that was stored rather than the second block that had the same hash code.

- The problem of hash code collisions can be mitigated by using a stronger (and usually more CPU intensive) hash code (SHA256 or SHA512) or by doing a byte by byte comparison with the block that is already stored. ZFS has options for enabling both of these techniques.

1.3 How Bacula Deduplication Optimized Volumes Work

- First, please be aware that you need the **aligned-sd.so** or the **bacula-sd-aligned-driver-x.y.z.so** Storage daemon plugin for Aligned Volumes to work. Please do not forget to define the **Plugin Directory** in the Storage daemon configuration file **bacula-sd.conf**.
- Aligned Volumes are enabled by specifying the Aligned keyword on a Device-Type directive in each Device resource in the bacula-sd.conf where you want to use aligned volumes:
- Aligned Volumes must have a unique Media Type that is different from Volumes that are used on non-Aligned devices.

```
DeviceType = Aligned
```

- When Aligned Volumes are enabled, the Device will create two files for each Volume. For example, if the Volume name is Test001. The files will be named Test001 and Test001.add. Test001 will contain all the metadata (filename, date created, permissions, ...) in the same format as non-aligned Volumes, and Test001.add will contain all the aligned file data (i.e. it will be block aligned).
- If you are writing the Volume with a single Job, each file backed up will have its data stored contiguously (i.e. no block interleaving. We recommend this way to write Volumes.
- If you set a maximum volume size for an aligned volume device, Bacula will only allow one backup Job at a time. In any case, it is probably more efficient to only backup to aligned volumes with one Job at a time. Rather than rely on Bacula to allow only one backup Job at a time, we strongly recommend that you following the advice in the next item below.
- We strongly recommend that you only write a Volume with a single Job at a time. This will be most efficient use of the Volume. To ensure that only one Job accesses the Volume at a given time, use the following directive in each Aligned device:

```
Maximum Concurrent Jobs = 1
```

  Note, later versions of Bacula Version (9.0.0) enforce this recommendation even if you do not specifically set the Maximum Concurrent Jobs.

1.4 New Storage daemon Device Directives

- Plugin Directory = `<directory-name>`
  This directive is required to enable the aligned volume feature. You must also have the Aligned Volume plugin loaded in the defined Plugin Directory before the Storage daemon is started.

- Device Type = Aligned
  This directive is required to make the Device write aligned volumes. Once this is turned on, each Bacula Volume will be split into two Volumes, one for the metadata, which does not deduplicate well, and one for the file data, which can be deduplicated.

- File Alignment = `<bytes>`
  Once the aligned volume format has been chosen, this is the key directive that will make the file data deduplication friendly. Each new file that is written into the .add file will be aligned to a multiple of this size. The value for this directive should be equivalent to the basic deduplication block size used by your Operating System. For NetApp, the deduplication block size is fixed at 4K. For ZFS the deduplication block size is set via the **recordsize** variable, and is by default 128K.

- Padding Size = `<bytes>`
  When Bacula is ready to write a block, and if it is not full, the block will be padded to the next multiple of the Padding Size by filling with zeros. If you want all blocks to be the same size as the Maximum Block Size, then set the Padding Size to the same value as the Maximum Block Size. The default for this is 0 bytes. In general, you should set it to the minimum physical block size used by your Operating System. For NetApp, the minimum physical block size is 4K, and for ZFS it appears that the minimum block size is 512 bytes. Setting the Padding Size larger than the smallest physical block size will cause Bacula to use more disk space than is necessary. Setting it too small or to zero will, in general, do no harm.

- Aligned Device = `<Directory-path>`
  This directive is similar to the Archive Device. It is optional, but if specified the Aligned Volume (xxx.add) will be placed in the Aligned Device directory specified, which is presumably different from the Archive Device where the Metadata Volume is placed. In the absence of the Aligned Device directive, both volumes will be placed in the Archive Device directory. Having two directives allows placing the Metadata volume in a directory (or partition) that is not deduplicated while the Aligned data volume can be deduplicated.

- Minimum Aligned Size = `<bytes>`
  This Directive is set to 4096 by default, and any file that is this size or smaller will be placed in the Metadata volume rather than the Aligned Volume. This allows you to put smaller files that are unlikely to deduplicate well into the

Metadata volume rather than overload the Aligned volume with data that does not deduplicate well.

Below we supply a few examples. The directives show should be in the **bacula-sd.conf** file within a **Device** resource:

For an EMC Data Domain DD800, the following works well:

```
Device Type = Aligned
Media Type = Aligned
Maximum Block Size = 64K
Minimum Block Size = 0
File Alignment = 4K
Padding Size = 4K
Minimum Aligned Size = 4K
Maximum Concurrent Jobs = 1
```

For ZFS assuming the default recordsize of 128K:

```
Device Type = Aligned
Media Type = Aligned
Maximum Block Size = 128K
Minimum Block Size = 0
File Alignment = 128K
Padding Size = 512
Minimum Aligned Size = 4096
Maximum Concurrent Jobs = 1
```

For NetApp:

```
Device Type = Aligned
Media Type = Aligned
Padding Size = 4K
File Alignment = 4K
Maximum Block Size = 64K
Minimum Block Size = 0
Minimum Aligned Size = 4K
Maximum Concurrent Jobs = 1
```

For ddumbfs:

```
Device Type = Aligned
Media Type = Aligned
Maximum Block Size = 128K
Minimum Block Size = 0
File Alignment = 128K
Padding Size = 512
Minimum Aligned Size = 4096
Maximum Concurrent Jobs = 1
```

For lessfs:

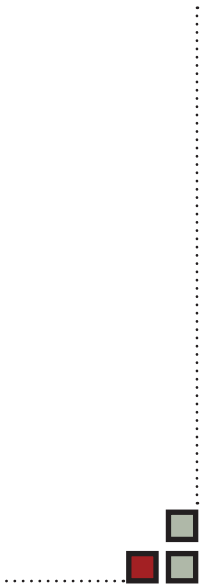
```
Device Type = Aligned
Media Type = Aligned
Maximum Block Size = 128K
Minimum Block Size = 0
File Alignment = 128K
Padding Size = 512
Minimum Aligned Size = 4096
Maximum Concurrent Jobs = 1
```

1.5 Things to Know About Deduplication

- Generally system tools such as **ls** or **du** do not know about sparse files (files with holes such as Bacula's Aligned Volumes), nor do they know about deduplicating and compressing filesystems such as ZFS. As a result, the size numbers that such system tools produce can be misleading.
- In order to really know the disk space used by a given file or a directory, you generally will need to use tools such as **zpool** and **zfs** to report disk utilization, and file sizes.

1.6 Things to Know About Bacula

- You must take particular attention to define a unique Media Type for devices that are Aligned as well as for each Virtual Autochanger that uses a different Archive Device directory. If you use the same Media Type for an Aligned device type that you use for a normal disk Volume, you run the risk that you will have data corruption on disk Volumes that are used on Aligned and non-Aligned devices.
- Blocks are may be written in any size and the files can be aligned at any value. However, they must generally be a multiple of 1024 bytes.
- When the values that you have specified for **Padding Size** is smaller than **File Alignment**, Bacula will generate Volumes that are sparse (i.e. that will have holes or unused areas of the Volume). This is normal, and it permits aligning the beginning of a file at a suitable deduplication boundary without wasting space.
- If you are using the Virtual Disk Changer feature of Bacula, it should work fine.
- We strongly recommend not using the Bacula disk-changer script, because it was written only for testing. Instead of using disk-changer, we recommend using the Virtual Disk Changer feature of Bacula, for which there is a specific white paper.
- We strongly recommend that you update all File daemons that are used to write data into an Aligned Volume. It is not required, but old File daemons do not have the newer FD to SD protocol, so consequently the **Minimum Aligned Size** is not respected for any older File daemons. Other than the fact that all file data will be written to the Aligned volume regardless of the file size and the value set for **Minimum Aligned Size** old File daemons will work correctly.



1.7 Things to Know About ZFS

- First and most important, you **must** set the Bacula **File Alignment** value to the same sized as the ZFS **recordsize** (128K by default). If you do not, deduplication will not work correctly.
- Second, you **must** set the Bacula **Maximum Block Size** to the same size or greater than the **File Alignment** value. If you do not, deduplication on ZFS may not work.
- Our testing with a ZFS kernel module installed on a Ubuntu 12.04 system indicates that for our simple dataset (the Bacula Source + binaries), the following Bacula parameters are optimal:

```
Device Type = Aligned
Maximum Block Size = 128K
Minimum Block Size = 0
File Alignment = 128K
Padding Size = 512
Minimum Aligned Size = 4K
```

- Our testing (as noted above) indicates that the following ZFS parameters are optimal, where **tank** is the pool:

```
sudo zfs set atime=off tank
sudo zfs set compress=on tank
sudo zfs set dedup=on tank
sudo zfs set recordsize=128k tank (default)
```

- ZFS has a surprisingly large number of settable options. The key ones for doing deduplication are: dedup, compression, and recordsize.
- ZFS has a number of tools such as **zdb**, but unfortunately they do not seem to have complete documentation. This means that in some of the examples given below, we have had to guess at the meaning of certain values that are displayed by the various commands.
- As mentioned above, a number of the ordinary Unix tools such as **ls** and **du** either do not correctly report sizes for sparse files (ls) or do not report allocated disk space taking into account deduplication and compression (ls and du).
- To get an accurate view of how much disk and memory is actually being used with deduplication and/or compression turned on, one must resort to the output from several ZFS commands. The most important being **zpool list tank**. See below for more details.

Output from a Bacula Job report might look something like the following:

```
...
FD Files Written:       2,126
SD Files Written:       2,126
FD Bytes Written:       141,738,979 (141.7 MB)
SD Bytes Written:       142,047,709 (142.0 MB)
Rate:                   141869.5 KB/s
Last Volume Bytes:      367,264,779 (367.2 MB)
```

The interesting thing is that the size of the data backed up is 142.0 MB as indicated by the SD Bytes Written, but the Volume size used by this job is 367.2 MB. Note: this number represents the sum of the last address of the metadata Volume and the Aligned Data Volume.

Is the data more than doubling in size?

The answer is **No** the actual sized used with dedupe and compression enabled on ZFS is about 78.7 MB (1/2 the original size) as will be shown below. The largest disk address used by Bacula is indeed on the order of 300 MB, but much of the Volume has holes and no disk space is allocated to those holes.

When one sets the File Alignment to 128K as shown above, each of the 2,126 files that were written will be aligned to a 128K byte boundary, while the original data was aligned to a 4K byte boundary since it came from a Linux ext4 filesystem. Since the Padding Size is 512, Bacula will write short blocks zero filed to the next 512 byte boundary for every file that is less than 128K then it will seek to the next 128K boundary to begin writing the next file. This means that there will be holes in the file, and the apparent file size (Last Volume Bytes) will indicate the address of the last byte in the file rather than the real number of bytes allocated (used) on disk.

The simplest way to know how much space was really used is to look at the output from a **zpool list** command:

```
zpool list tank
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH
tank  19.9G  78.7M  19.8G     0%  1.02x  ONLINE
```

and we see under the ALLOC column, the space used is 78.7 MB. This is approximately half of the original space of 142 MB that the original data used. The difference is due to compression in this case.

After doing a total of 10 identical Full backups, we get the following output:

```
zpool list tank
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH
tank  19.9G  84.2M  19.8G     0%  10.00x  ONLINE
```

Notice that 10 times the data (i.e. 10 X 142 MB) increased the total ZFS disk space allocated to only 84.2 MB. This is due to the fact that it was perfectly deduplicated and only a small amount of Bacula metadata and ZFS pointers were written in addition to the original data. Bottom line, ZFS stores 1.42GB of data using 84.2MB of disk space! At the same time, the Bacula Aligned Volume appears to be using 3.4GB of data. See below for more details.

```
zdb -DD tank
DDT-sha256-duplicate: 2724 entries, size 306 on disk, 165 in core
DDT-sha256-unique: 45 entries, size 7884 on disk, 9830 in core

DDT histogram (aggregated over all DDTs):

bucket              allocated                     referenced
------  ------------------------------   --------  --------------------
refcnt  blocks  LSIZE  PSIZE  DSIZE   blocks  LSIZE  PSIZE  DSIZE
------  ------  -----  -----  -----   ------  -----  -----  -----
     1      45  5.38M  2.23M  2.23M       45  5.38M  2.23M  2.23M
     8   2.61K   334M  75.4M  75.4M    26.1K  3.26G   754M   754M
    16      42  5.25M   466K   466K      860   108M  9.18M  9.18M
    32      10  1.25M   530K   530K      400    50M  20.7M  20.7M
 Total   2.70K   346M  78.6M  78.6M    27.4K  3.42G   786M   786M

dedup=10.00, compress=4.46, copies=1.00, dedup*compress/copies=44
```

Another command that can be useful to understand what is going on is **zdb -DD tank**, shown in the next figure:

The important numbers from above all all in the Total row. The first is the 78.6M under the allocated DSIZE column (second column). It corresponds to the real disk space used by the data. The second number is 27.4K, the total number of referenced blocks (sixth column). This number is important when computing the memory requirements to keep all the the Dedupe table (DDT) in memory. Each DDT item takes approximately 170 to 300 bytes, so when the number of referenced blocks times 300 becomes a large percentage of your main RAM, all disk operations will slow down because the ZFS must page the lookup tables. The final number of importance is the 3.42G under the referenced LSIZE column (seventh column). This number is equivalent to the largest file address that Bacula reports in the Job report.

1.8 More on ZFS

Good sources of additional information on using and tuning ZFS can be found at:

```
http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide

http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

http://www.brendangregg.com/Slides/zfsperftools2012.pdf
```

1.9 Creating a ddumbfs

You can use this command to create the ddumbfs filesystem:

```
mkddumbfs -s 200G -B 128k /d0/ddumbfs/
```

```
and used this in fstab

oparent=/d0/ddumbfs/        /tank        fuse.ddumbfs    noauto 0 0
```

1.10 Restrictions and Limitations

- Aligned Volumes do not permit running multiple concurrent Jobs to the same Volume. So you **must** add the following directive to all Aligned volume Device resources in the Storage daemon's configuration file:

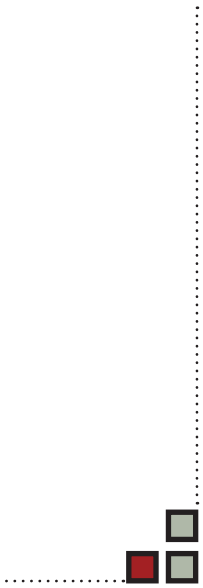
  ```
  Maximum Concurrent Jobs = 1
  ```

  Despite the fact that a only a single Job can write to an Aligned Volume at a given time, you may run multiple simultaneous Aligned backup Jobs by having each Job write to a different Device. The easiest way to set this up is to use a Bacula Virtual Autochanger (see the White Paper on this subject).

  Note that multiple Jobs may be written to the same Aligned volume, the only constraint is that they may not use the volume simultaneously. For example, if you have a nightly backup of a specific client machine, that client may write to the same Volume on different nights.

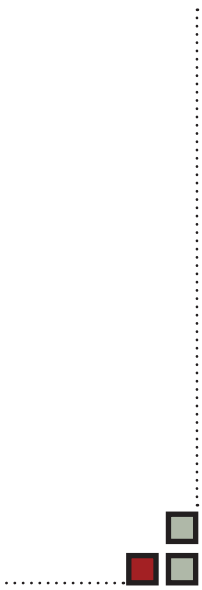
- You must take care to define unique Media Types to Aligned Volumes that is different from Media Types for non-Aligned Volumes.

- If you are using VTL software that is not part of a Bacula release, it is unlikely it will work. If the VTL simulates a tape drive, it definitely will not work. Bacula's Aligned Volumes work only with Bacula disk Volumes. If you need a VTL, consider using Bacula's Virtual Autochanger feature.



For More Information

For more information on Bacula Enterprise Edition, or any part of the broad Bacula Systems services portfolio, visit www.baculasystems.com.



Rev : 2 V. 1.8
Author(s): KES