Disk Backup Design

Using Bacula

This document is intended to provide insight into the considerations and processes required to design and implement a Disk Backup strategy for a large site with Bacula.

Bacula Systems White Paper

Version 2.5, November 25, 2014

www.baculasystems.com

Contents

General

This White Paper will outline the best practices for setting up a large disk based backup using Bacula version 7.0 or later.

This document has been adapted from a Bacula Enterprise white paper and is provided to you by Bacula Systems for your personal use in the hopes that it will improve your experiences with the Bacula community version.

1.1 Assumptions

The following are the assumptions that we have made:

- You have a basic understanding of Bacula, Volumes, recycling, and backup strategies.
- You are using Bacula 7.0 or later. If using Bacula 5.2 or older, the Changer Command should be set to an empty string instead of `/dev/null`.
- The retention period for recovering a file on a daily basis is 30 days. That is for 30 days, you can go back to any prior version of a file on a day by day basis.
- The retention period for recovering a file on a monthly basis is 60 days.
- You are doing only disk based backup.
- You have more than one Storage daemon that has access to large data stores.
- You have one to two thousand client machines to backup each day.
- You are using PostgreSQL as your Bacula catalog database. This is highly recommended by Bacula Systems.
- You are using multiple Pools to separate the Volumes that various clients write to, and to help optimize recycling of old data that is no longer needed. More on Pools below.

Volume Management

When using disk based backup, there are a number of basic Bacula issues we must discuss before getting into the details of designing a backup strategy.

2.1 Optimal Volume Size

When writing a disk Volume, you might ask: What size should I use for a disk Volume? In general, our answer would be it depends on a lot of factors, but somewhere between 10 Gigabytes and 100 Gigabytes for today's systems seems reasonable, because 10-100 GB is a compromise between having a lot of Volumes and having Volumes that are very large. The more Volumes you have, the more complicated is the task of record keeping. The larger the Volume size, the more data that may be not recycled because it contains recent records. Also larger Volumes take more time to copy if you ever need to move them.

2.2 Recycling Volumes

Let's assume, as noted above, that you want to be able to restore a given file for at least a month on a daily basis. Generally you do so by running an Incremental backup once a day (usually at night) with occasional Full backups. If you write all your Incremental Jobs to the same Volume, and you have a Volume retention period of 30 days, and you started with your first Incremental Job written to Vol001 on the first day, then on the 31st day, you begin writing to Vol002, you will not be able to recover the space used by the Job written to Vol001 on the first day until 61 days after it was written. This is because the 30 day retention period on Vol001 begins counting down only after the last time it was written.

If instead of writing 30 days of Incremental data to a single Volume, you had written a new Volume every day, when your retention has expired, you will be able to recycle a bit each day rather than waiting an additional 30 days and recycling a big volume. In order to have 30 days of backup, you will need to keep your Full backups for at least 30 + 30 days so that you can always recover your system back 30 days. For Incremental Volumes you will need to keep them for 30 + 29 days (the one day of difference being the second Full which can substitute for an Incremental. This means that your retention period as specified in Bacula must be at least twice the period you want to be able to restore to on a day by day basis.

Thus the smaller the period of time that you write into a given Volume the faster the Volume can be recycled, but the more Volumes you will need. As an example, if you backup 1,000 clients a night, and all the data for a single day goes into one Volume, then at the end of 30 days, you will have 30 Volumes. If you write each client into a separate Volume, at the end of the 30 days, you will have 30,000 Volumes. It should be obvious that for Bacula to manage 30,000 Volumes or more is more compute intensive than if it is managing only 30 Volumes. However, the size of the Volumes for a given amount of data will be very different.

Often times, you will want to make some compromise to balance Volume size versus the number of Volumes. For example, you might limit the number of Jobs to 100 for a given Volume. Doing so with the above example, will mean that you will need 10 Volumes per night to store 1,000 Jobs or 300 for 30 days. Depending on your data size this may be a good compromise.

2.3 Limiting Volume Size

When Bacula writes to a disk, Bacula treats that disk much like a tape. For example, unless you have either defined some limitation to writing into a disk Volume, Bacula will continue writing until the Volume totally fills the space on the disk partition that contains that Volume, which is not very desirable, as it effectively prevents **Bacula** from reusing volumes when the data on them is older than the configured retention time.

There are a number of different ways to limit the size of a given disk Volume: by the number of Jobs, by the time the Volume is used, by the Volume size, ... In this design document we will generally either restrict a Volume size by the number of Jobs written to the Volume or by the Volume Size.

Most of the different ways to limit the Volume size are defined in the Pool resource of the bacula-dir.conf file. There are many options, but the most commonly used are:

- **Volume Use Duration**
- **Maximum Volume Jobs**
- **Maximum Volume Bytes**

Setting `Volume Use Duration` to 12 or 23 hours, and `Maximum Volume Bytes` to between 10 and 50 GB is probably a reasonable choice for most sites, as in the example below:

```
Pool {
  Name = Default
  Pool Type = Backup
  Recycle = yes
  AutoPrune = yes                 # Prune expired volumes
  Volume Retention = 365d         # one year
  Label Format = Vol-             # Allow to auto label
  Action On Purge = Truncate      # Allow to volume truncation
  Volume Use Duration = 14h       # Force volume switch
  Maximum Volume Bytes = 50GB     # Limit volume size
  Maximum Volumes = 100           # allow 5TB for this Pool
  Maximum Volume Jobs = 100       # Force a Volume switch after 100 Jobs
}
```

If you change the values in the Pool definition the new values will apply to any new Volumes that you create. If you want your old existing Volumes to use the new values, you must explicitly do so with the:

```
update volume FromAllPools
```

2.4 Pruning Volumes

Since automatic catalog pruning can be time consuming, for maximum performance, you may want to avoid automatic pruning of old jobs at the end of each backup Job. To do that, set the directive `AutoPrune = No` in all Client resources, and Bacula will not automatically try to prune jobs. However, unless you want your catalog to grow very large, you will need to schedule a RunScript to explicitly handle the pruning. described in the following section.

2.4.1 New Prune "Expired" Volume Command

It is now possible to prune all volumes (from a pool, or globally) that are "expired". This option can be scheduled after or before the backup of the Catalog and can be combined with the Truncate On Purge option.

This new Expired option to the Prune commands can be scheduled in a RunScript to explicitly handle the pruning. The following two variations are accepted:

```
* prune expired volumes
* prune expired volumes pool=FullPool
```

To schedule this option automatically, it can be added as a Runscript to the BackupCatalog job definition.

```
Job {
  Name = CatalogBackup
  ...
  RunScript {
    Console = "prune expired volume yes"
    RunsWhen = Before
  }
}
```

2.4.2 Manual Pruning

In older Bacula versions prior to the Expired option on the Prune command, manual pruning was done via a Perl script. This method is now deprecated, but shown here for historical reasons:

```
Job {
   Name = "PRUNING"
   Type = Admin
   JobDefs = JobDefault

   RunBeforeJob = "manual_prune.pl --doprune --expired"
}
```

Note, the file manual_prune.pl is available on the bacula.org web site with this white paper.

2.5 Attribute Spooling

By default, when using disk backup, Attribute Spooling is disabled to save space on the Storage Daemon side. The disadvantage is that it makes individual jobs run much slower. For maximum performance, it is better to spool the attribute data so

that it can be inserted in one big batch at the end of each job. To do so, use the following ...

```
Job {
  ...
  Spool Attributes = Yes
}
```

The file attributes are spooled into the **working** directory, and require about 150 bytes per file, which means a backup consisting of a million files (moderately large backup) would need 150 MB of space in the working directory. This must be multiplied by the number of simultaneous Jobs you are running.

Due to the importance of attribute spooling, we have made it default on for Bacula version 6.0.3 and later.

2.6 Creating New Volumes

In disk based backup configurations, you may want to let Bacula create new volumes when needed. For that, you can add the `Label Format` option to all your Pool resources in the **bacula-dir.conf** file, and configure your Storage daemon (SD) devices in the file **bacula-sd.conf** to label new volumes automatically (`Label Media = Yes`). When using these directives, you may also want to limit the number of Volumes that can be created with the `Maximum Volumes` Pool option to avoid creating too many Volumes if something goes wrong with pruning (not mandatory if you schedule a pruning job).

2.7 Volume Fragmentation

By default, Bacula is able to share a device between multiple concurrent jobs by allowing them to write simultaneously. Simultaneous writes cause the Volume to have interleaved Job blocks as shown in the picture on the left of Figure 2.1. The drawback is that the restore process could be a bit slower, particularly on tape Volumes, because the Storage Daemon will have to read more chunks of data and must seek between each block. For disk storage, the seek time is very rapid so interleaved Job blocks should not cause a noticeable performance degradation.

However, for tape we believe that running from 5 to 10 concurrent Jobs per tape Volume, with data spooling turned on (see: Figure 2.1), is a good compromise, unless you have a restore time objective that is very short.

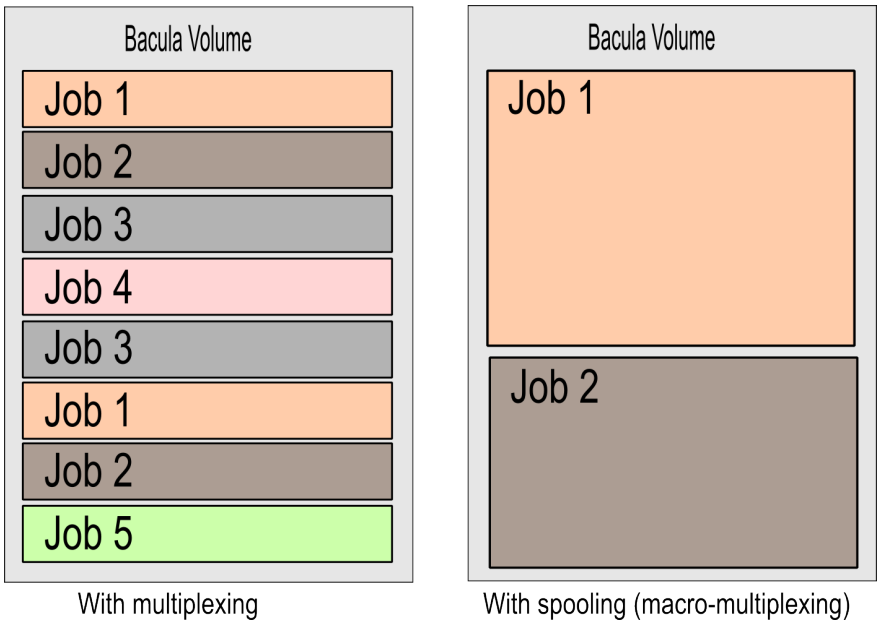


Figure 2.1: Impact of Volume Fragmentation

Writing Concurrently to Multiple Devices

We have seen that disk backup can use multiplexing with multiple concurrent jobs without problems. Now we will discuss how to configure Bacula to spread the work load over multiple devices automatically.
While it is possible to define multiple Devices in the Storage daemon, and multiple Storage resources in the Director that use those devices, it can be very complicated to decide what Job to put on what storage device. In the extreme case, where you want every Job to have its own set of Volumes to cleanly separate the data, and you run 2000 Jobs a night, say 50 at a time, you will need 50 separate Storage definitions in the Director and 50 separate Device definitions in the Storage daemon, further more, each of your Jobs will have to reference one of the 50 devices that it can write to. The next section explains how to do this.

3.1 Grouping Storage Devices

A simple way to define multiple devices is to use Bacula's Virtual Autochanger definitions. To do so, you define a single Storage devic in the Director that connects to a single Autochanger in the Storage daemon, which has 50 devices connected to it. This means that all your Jobs use a single Storage definition. Then by the use of the **Maximum Concurrent Jobs** directive in each Device definition and by using Pools, you can ensure that one or a given number of Jobs write to each of the Devices you have defined. In general, you can group Jobs by using Pools. This is shown in the example below.

In the **bacula-dir.conf** file, you can define an Autochanger as follows:

```
# Definition of file Virtual Autochanger device
Storage {
  Name = File1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = FileChgr1
  Media Type = File1
  Maximum Concurrent Jobs = 10       # run up to 10 jobs a the same time
  Autochanger = yes                  # important
}
```

In the **bacula-sd.conf** file, the equivalent definitions are:

```
#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr1
  Device = FileChgr1-Dev1, FileChgr1-Dev2
  Changer Command = /dev/null
```

```
  Changer Device = /dev/null
}

Device {
  Name = FileChgr1-Dev1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = FileChgr1-Dev2
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}
```

Notice that the Changer Command is empty and the Changer Device points to /dev/null. This is because the autochanger is virtual, and Bacula knows from each of the Device definitions where the Volumes are located (in the Archive Device directive).
Each Device should point to the same directory, have the same MediaType. If two Devices point to different directories, they should have different MediaType.
Setting **Maximum Concurrent Jobs** will limit the number of jobs that will be able to use each Device at the same time (5 in the above example). When the first Device is running the maximum concurrent jobs defined, Bacula will automatically have the next Job use the next available Device.

3.2 Writing to Multiple Disks

In addition to writing multiple Jobs simultaneously to the same Volume, and writing to multiple Volumes a the same time as show above, Bacula can, of course, simultaneously write to multiple Volumes on different Disks or partitions. There are two ways to write to separate disks or partitions. Probably the simplest is to simply create a second Autochanger definition in the Director and the Storage daemon that points to a different disk partition. When creating a second Autochanger, it is very important to use a different Media Type. This will permit Bacula during restore to find a suitable device. If you do not use a separate Media Type, it is possible that one day Bacula will ask to mount a Volume in one Autochanger on the second Autochanger, which is probably something you never want to do.

A second way to write to a separate disk or partition is to simply define one or more of the Devices in the Autochanger definition of the Storage daemon to point to a different mount point on another disk or partition. As with the case of defining a

second Autochanger, you will want to define a different Media Type so that restores will work correctly.

If we take the example configuration file given above and expand it to include writing to a second Autochanger, we might have something like the following:

In the **bacula-dir.conf** file, you can define two Autochangers as follows:

```
# Definition of file Virtual Autochanger device
Storage {
  Name = File1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = FileChgr1
  Media Type = File1
  Maximum Concurrent Jobs = 10       # run up to 10 jobs a the same time
  Autochanger = yes
}
```

```
# Definition of file Virtual Autochanger device
Storage {
  Name = File2
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = FileChgr2
  Media Type = File2
  Maximum Concurrent Jobs = 10       # run up to 10 jobs a the same time
  Autochanger = yes
}
```

In the **bacula-sd.conf** file, the equivalent definitions are:

```
#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr1
  Device = FileChgr1-Dev1, FileChgr1-Dev2
  Changer Command = /dev/null
  Changer Device = /dev/null
}

Device {
  Name = FileChgr1-Dev1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = FileChgr1-Dev2
  Media Type = File1
  Archive Device = /tmp
```

```
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}
```

```
#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr2
  Device = FileChgr2-Dev1, FileChgr2-Dev2
  Changer Command = /dev/null
  Changer Device = /dev/null
}

Device {
  Name = FileChgr2-Dev1
  Media Type = File2
  Archive Device = /var/tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = FileChgr2-Dev2
  Media Type = File2
  Archive Device = /var/tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}
```

For each physical device on which you write Volumes (/tmp and /var/tmp in the above example), you will need sufficient disk space to hold all the data in your Volumes. If Bacula fills a disk partition completely, it will stop and ask you to mount a new Volume and you will be stuck unless you can recycle or delete some of your existing Volumes. This is because once Bacula chooses a drive to write on at the beginning of the Job, it cannot switch during that Job to writing on another drive. Thus currently, there is no way for you to create Volumes on another disk partition and have Bacula automatically switch when the disk fills.

Just the same, there is a simple way to make Bacula read or write to a different partition. You may do it by manually creating Volumes and using symbolic linking. For example:

```
$ ln -s /var/tmp/Vol001 /tmp/Vol001
$ ln -s /var/tmp/Vol002 /tmp/Vol002
$ ln -s /var/tmp/Vol003 /tmp/Vol003
```

Thus the Volumes you created on `/var/tmp` appear to Bacula as if they are on `/tmp`. So after linking as shown above, you can label the Volumes as Volume `Vol001`, `Vol002`, ... and Bacula will use them as if they were on /tmp but actually write the data to `/var/tmp`. The only minor inconvenience with this method is that you must explicitly name the disks and cannot use automatic labeling unless you take care so that the labels exactly match the links you have created.

Due to the manual nature of the above, we do not particularly recommend symbolic linking. However, a bit of scripting could certainly make it much simpler.

3.3 Using Pools

Another way to use multiple devices at the same time is to define different pools and distribute your jobs among them. Those Pools can have the same attributes (such as retention), their main purpose will be to let the Storage Daemon write concurrently to multiple devices. Depending on how many jobs you run at the same time (Max Concurrent Jobs), this technique could be a bit complex to setup. This is shown in Figure 3.1.

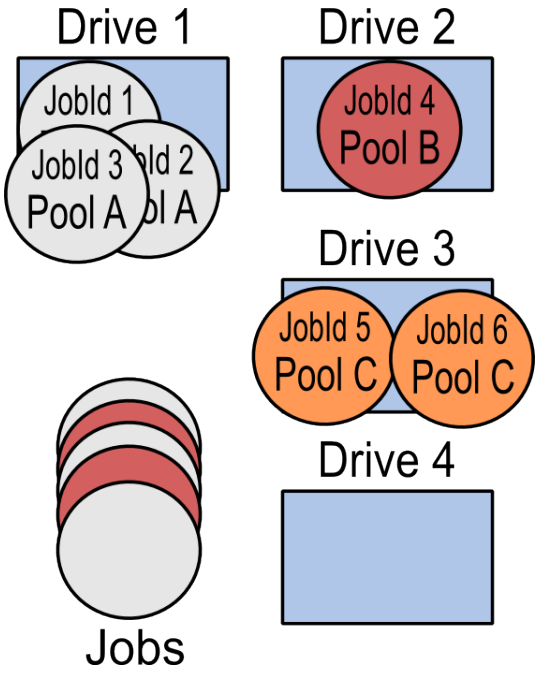


Figure 3.1: Job Distribution using Pools

By having a different Pool for each Job, you can also force Bacula to write all the data for each Job on separate Volumes. This technique can also be applied to groups within an organization. You may want the data for your Personnel department, R&D department, Sales, and Management all on separate Volumes. This is relatively easy to do by configuring each Client machine to use the appropriate Pool.

3.4 A Dedicated Device per Client

If you want every Client to write to a separate Volume, probably the easiest way is to define a separate Pool for each Client. When the Job is run to backup that Client, Bacula will automatically ensure that no other Jobs use the same Device because the Pool will be different. This will effectively force each Device to run a single Job at a time. Consequently, in your **bacula-sd.conf** file, you will want to define one Device for each Job that will run simultaneously. However, since we are using a Virtual Autochanger, each Job in the **bacula-dir.conf** file can reference the same Storeage resource. Bacula will automatically manage which Job uses which Drive.

3.5 Running Restores and Backups Concurrently

In order to be able to restore a file at any time, you may want to define a special restore Device in the bacula-sd.conf file that will be used only for restoring. This is to ensure that even during a heavy backup load, there will be at least one Device that is always available for restores. This special device can either be a device that is not included in your Autochanger definition, or it can be a device in the Autochanger that has been marked not to be automatically selected for backups. In both cases, the device can only be accessed by naming it directly. An example of a device in an Autochanger that is reserved is as follows:

In the **bacula-dir.conf** file, you can define an Autochanger as follows:

```
# Definition of file Virtual Autochanger device
Storage {
  Name = File1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = FileChgr1
  Media Type = File1
  Maximum Concurrent Jobs = 10       # run up to 10 jobs a the same time
  Autochanger = yes                  # Important
}

Storage {
  Name = RestoreStorage1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = RestoreStorage1
  Media Type = File1
  Autochanger = yes                  # important
}
```

Above, we have added another Storage device definition in the Director that is named RestoreStorage1, and this device will be always ready in the Storage daemon to accept restore Job. During a restore, if all drives are busy, you can manually enter the Restore storage to be used for the restore.

In the **bacula-sd.conf** file, the equivalent definitions are:

```
#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr1
  Device = FileChgr1-Dev1, FileChgr1-Dev2, Restore
  Changer Command = /dev/null
  Changer Device = /dev/null
}

Device {
  Name = FileChgr1-Dev1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = FileChgr1-Dev2
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = RestoreStorage1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  AutoSelect = no
  Autochanger = yes
}
```

Notice that we have set **AutoSelect = no** in the Device definition. This prevents Bacula from selecting that particular device for backup. When AutoSelect=no is present, the only way to use that Device is by explicitly referencing it (rather than the Autochanger) in the Director.
During the restore process, Bacula will choose the original Storage resource that was used to backup your files to restore your files, and if all devices are busy, you will be able to manually select the RestoreStorage and thus complete the restore.
If you are using Virtual autochanger or group of devices, you can add this Device to the Autochanger device group.

3.6 Truncate Volume on Purge

By default, when Bacula purges a volume, it just changes the volume status in the catalog. It doesn't physically touch your Volume or its data. Thus you can still recover data after the retention period has expired, but the Volume continues to occupy disk space. This is generally what one wants providing one has sufficient disks space. For tape based backup, this default behavior is also very useful because you may want to avoid extra mount operations to just truncate volumes.
However, by truncating the Volume when it is purged, you can recover the disk space allocated. To do this, you will need to add the directive:

```
Action On Purge = Truncate
```

to all your Pool resources, and you will need to schedule a console RunScript to execute the purge during a period when Bacula is not backing up files. (Note: if you add this directive, you may need to run an `update` command to apply Pool's configuration changes to the Volumes already created in the catalog).
Below, we show an example of executing the truncate in the Job that runs the nightly catalog backup.

```
Job {
 Name = CatalogBackup
 ...
 RunScript {
   RunsWhen=After
   RunsOnClient=No
   Console = "purge volume action=all allpools storage=File"
 }
}
```

3.7 Job Levels

A major part of designing a good backup scheme is deciding what Job levels (Full, Differential, and Incremental) you will use and when. Job levels are normally determined in the Bacula Schedule resources, but at the same time one must understand the interactions with Retention periods. A set of Job levels which is very common is a Full backup once a month, a Differential once a week, and Incremental backups all other days.

The main reason for using Differential backups is to reduce the number of individual JobIds needed to do a restore. Keeping the number of JobIds to a minimum is especially important for tape storage, where each JobId may be stored on a different tape, and where seeking to the end of the tape to find one Job may be particularly time consuming.

When backing up to disk, seeks are very rapid and changing Volumes is almost instantaneous, so the need for Differential backups is much reduced. In fact, with the exception of some unusual pattern of file updates, using Differential backups for disk storage generally increase the amount of data backed up, so for disk storage, we generally recommend that you use Full and Incremental backups.

Your requirements for keeping specific numbers of backup copies of a given file will, in general, determine how often you want to do Full backups.

3.8 Accurate

Normally, Bacula does not track deleted files nor does it detect files added to a system but with an old date. If you have a requirement to track deleted and/or files added to the system but with old dates, then you should consider using **Accurate = yes** in your Job definitions. However, it does require more memory on the Client machines.

If do not absolutely require Accurate backups, you might still turn it on once a week if you do only one Full per month as this will keep your backups a bit more in sync with the real state of the filesystem. To do this, we suggest that you add it as an option in your Schedule. Please see the Schedule examples at the end of this white paper.

3.9 Max Full Interval

It is quite easy for a Full backup to fail for some reason (hardware, comm line, ...), so to ensure that you have at least one Full backup a month, we recommend that you set:

```
Max Full Interval = 31 days
```

in your Job resource. This will cause an Incremental Job to be upgraded to a Full if Bacula does not find a Full that is less than 31 days old.

Another interesting variation of this is to not have any Full backup in your schedule but to simply set:

```
Max Full Interval = 30 days
```

This will cause the Director to automatically upgrade your Incremental Jobs to a Full at the right interval.

3.10 Schedules

If you are doing a monthly Full backup and then daily Incrementals, and you have 1000 Jobs, to balance your load, you will probably want to run roughly an equal number of Jobs with Full backups each night of the month, so you need to have 30 different Schedules so that you don't overload your backup system by running all your full Jobs the same night.

3.11 Email your Catalog BSR Files

We recommend that you modify the script:

```
/opt/bacula/scripts/delete_catalog_backup
```

to email a copy of the BSR file that was written for the Job that does your Catalog backup to yourself or to some safe email address where these BSR files can be stored on another machine. If your Bacula server crashes due to a disk failure or some other catastrophic event, with the BSR files, you will be able to easily recover your catalog database and thus your server configuration.

4

Putting it All Together

Now that we have covered the main points of how one devises a good backup strategy for Bacula, let's put it all together using the assumptions at the beginning of this white paper, namely:

- you have a big shop; 1000-2000 backups/night;
- you have a number of large storage devices or have a number of Storage daemons attached to large storage devices;
- you want to be able to restore a file on a daily basis for 30 days and on a monthly basis for 60 days;
- and finally, each client machine must have its own Volumes.

The steps to take to get to where you have a good system are:

- In the Director define one Storage resource for each Storage daemon you will have. It should have a sufficient number of Devices to run approximately 50-60 simultaneous Jobs, plus one or two Devices to be reserved for restores. If you put all your data for a given Client on one Volume, you will need 50-60 Device definitions plus say two reserved Devices. If you allow 10 simultaneous Jobs per Device, you will only need 5 or 6 Devices.
- You will probably want 30 different Schedules so that your Full backups are even distributed across the different days of the month.
- You will want two Pools per client: one for Full backups, and one for Incremental backups, because the two Pools will have different retention periods to keep storage space to a minimum.

4.1 bacula-dir.conf

Your Director configuration file **bacula-dir.conf** has the following main elements:

```
Director {                    # define myself
  Name = bacula-dir
  DIRport = 9101              # where we listen for UA connections
  QueryFile = "/opt/bacula/scripts/query.sql"
  WorkingDirectory = "/opt/bacula/working"
  PidDirectory = "/opt/bacula/working"
  Password = "rq6zcEBTvsKEaQ9XUjPbi"
  Messages = Daemon
  Maximum Concurrent Jobs = 50
}

JobDefs {
  Name = "DefaultJob"
  Type = Backup
  Level = Incremental
```

```
  Client = bacula-fd
  FileSet = "All"
  Schedule = "First-Sun-Full"
  Storage = File1
  Messages = Standard
  Pool = Client1-Full
  SpoolAttributes = yes
  Priority = 10
  Write Bootstrap = "/opt/bacula/bsr/%c.bsr"
  Max Full Interval = 31 days # a full once a month
  Spool Attributes = yes
}

Job {
  Name = "CatalogBackup"
  Type = Backup
  JobDefs = DefaultJob
  Priority = 11
  Schedule = EveryNight
  RunBeforeJob = "/opt/bacula/scripts/make_catalog_backup bacula bacula"
  RunScript {
    RunsWhen = After
    RunsOnClient = No
    Command = "/opt/bacula/scripts/delete_catalog_backup"
    Console = "purge volume action=all allpools storage=File"
  }
}

Job {
  Name = "Pruning"
  Type = Admin
  JobDefs = DefaultJob
  Priority = 12
  Schedule = EveryNight
  RunBeforeJob = "/opt/bacula/scripts/manual_prune.pl --doprune --expired"
}

Job {
  Name = Client1
  JobDefs = "DefaultJob"
  Pool = Client1-Full
  Full Backup Pool = Client1-Full
  Incremental Backup Pool = Client1-Inc
  Client = Client1
  Schedule = First-Sun-Full
}

...

Client {
  Name = Client1
  Address = Client1
  ...
}

...

Pool {
  Name = Client1-Full
  Pool Type = Backup
  Recycle = yes
  AutoPrune = no                # Done by Pruning Job
```

```
  Volume Retention = 70d          # 60 days + a bit extra
  Label Format = Client1-Full     # Allow to auto label
  Action On Purge = Truncate      # Allow to volume truncation
  Maximum Volume Bytes = 50GB     # Limit volume size
  Maximum Volumes = 100           # allow 5TB for this Pool
  Maximum Volume Jobs = 1         # Force a Volume switch after 100 Jobs
}

Pool {
  Name = Client1-Inc
  Pool Type = Backup
  Recycle = yes
  AutoPrune = no                  # Done by Pruning Job
  Volume Retention = 70d          # 2 X 30 days + a bit extra
  Label Format = Client1-Inc      # Allow to auto label
  Action On Purge = Truncate      # Allow to volume trucation
  Maximum Volume Bytes = 50GB     # Limit volume size
  Maximum Volumes = 20
  Maximum Volume Jobs = 6         # Force a Volume switch after a week
}

...

Schedule {
  Name = "EveryNight"
  Run = Level=Full sun-sat at 5:04
}

Schedule {
  Name = First-Sun-Full
  Run = Full 1st sun at 22:00
  Run = Incremental mon-sat at 22:00
  Run = Incremental Accurate=yes 2nd-5th sun at 22:00
}

Schedule {
  Name = First-Mon-Full
  Run = Full 1st mon at 22:00
  Run = Incremental tue-sun at 22:00
  Run = Incremental Accurate=yes 2nd-5th mon at 22:00
}

...

# Definition of file Virtual Autochanger device
Storager {
  Name = File1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = FileChgr1
  Media Type = File1
  Maximum Concurrent Jobs = 10       # run up to 10 jobs a the same time
  Autochanger = yes
}

Storage {
  Name = RestoreStorage1
# Do not use "localhost" here
  Address = bacula-sd                # N.B. Use a fully qualified name here
  SDPort = 9103
```

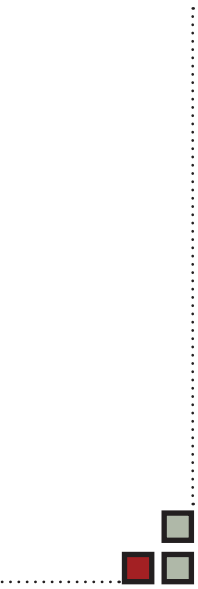
```
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
  Device = RestoreStorage1
  Media Type = File1
  Autochanger = yes
}
```

4.2 bacula-sd.conf

Your **bacula-sd.conf** has the following main elements:

```
Storage {                 # definition of myself
  Name = bacula-sd
  SDPort = 9103           # Director's port
  WorkingDirectory = "/opt/bacula/working"
  Pid Directory = "/opt/bacula/working"
  Maximum Concurrent Jobs = 60
}

Director {
  Name = bacula-dir
  Password = "a+YXY3kfS9ObgZ52ebJT3W"
}

Messages {
  Name = Standard
  director = bacula-dir = all
}

#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr1
  Device = FileChgr1-Dev1, FileChgr1-Dev2, RestoreStorage1
  Changer Command = /dev/null
  Changer Device = /dev/null
}

Device {
  Name = FileChgr1-Dev1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

Device {
  Name = FileChgr1-Dev2
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
```

```
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

...

Device {
  Name = RestoreStorage1
  Media Type = File1
  Archive Device = /tmp
  LabelMedia = yes          # lets Bacula label unlabeled media
  Random Access = Yes
  AutomaticMount = yes      # when device opened, read it
  RemovableMedia = no
  AlwaysOpen = no
  AutoSelect = no
  Autochanger = yes
}

...
```



5

Conclusions

To summarize, Bacula Systems advises you to:

- send the BSR files created by backing up your catalog to an email address that is on a separate machine.
- Properly configure your Pools with limits on the usage duration and the maximum size of your volumes
- configure the truncate operation for purged volumes
- turn off AutoPrune, and prune your volumes regularly
- label your volumes automatically with AutoLabel
- configure multiple devices in the Storage daemon and group them in a Virtual Autochanger resource in the Director's configuration file
- configure extra Device dedicated for Restore in the Virtual Autochanger
- use a different MediaType for each physical location
- limit the Maximum Concurrent Jobs for each device
- test your disk system to find the best global Maximum Concurrent Jobs for the Storage daemon
- turn on attribute spooling on all your jobs (Spool Attributes=Yes)
- use xfs with barrier=0 over traditional filesystems such as ext3
- use new filesystems such as ext4, zfs or btrfs, xfs, specially if your storage array is larger than 8TB
- if possible mirror your volumes to another place over the network, or to copy important volumes to tape
- either turn on Accurate in your Job resources or as a compromise turn it on once a week in a Schedule resource
- use MaxFullInterval = 31 to ensure you always have a current Full backup

If you would like the above **bacula-dir.conf** and **bacula-sd.conf** files, they should accompany this document with the names: **disk-example-bacula-dir.conf** and **disk-example-bacula-sd.conf**, otherwise please let us know, and before using them, please test them to ensure that they work correctly on your system as it is possible that there are some typos or other problems.

Revision History

Version	Date	Owner	Changes
2.3	18 Sep 2014	Kern	Adapt to community version
2.4	27 Sep 2014	Kern	Tweak
2.5	25 Nov 2014	Kern	Tweak