Progressive Virtual Fulls™

Combining Global Endpoint Deduplication with Virtual Full Backups

How to get the most out of two independently valuable features for potentially significant further performance gains.
Conceptual Overview

We assume the concepts of the Incremental Forever approach and Data Deduplication to be known in general. For the purposes of this introduction we assume that it is planned to do incremental-level backups extensively, and we also assume that Virtual Full backups will be done often.

The underlying idea here is to ensure that, at all times, a sequence of backups exists which consists of one full-level backup and a number of incremental ones. Virtual full backups are done comparatively often, so that in an example case there would always exist a backup history of 30 days of data, and no older backups would have to be kept.

The backup data would then consist of one full backup plus 30 incremental ones. Each day after an incremental backup has been done, a new virtual full backup would be created and the previously-existing one could be immediately removed (or, probably more valuable, would be moved to cheaper storage occasionally). Thus, the history of backed-up data would essentially allow reasonably fast restores to each of the previous 30 days, and not take up any more disk space than required. Essentially, the required backups would slide forward one day, once per day, as shown in figure 1.

Figure 1: Backup Sequence Slides Forward One Day, Each Day
Basic Implementation in Bacula

Obviously, the key element of the outlined backup scheme is to create a new Virtual Full backup each day, aggregating the previous (virtual) full backup and the oldest incremental backup into a new virtual full one. This is easily done with Bacula, though it requires a bit of scripting to do. This is due to the fact that, by default, Bacula would aggregate all individual backups of a sequence, whereas here we want to limit the resulting Virtual Full to a much shorter, apparently “older”, sequence. From the point of view of a Bacula user, this is done by explicitly providing the Job ids of the backups to consolidate, and these need to be, for this purpose, determined by querying the Catalog database. Once the Job ids are known, however, the actual consolidation can be started using a very straightforward `bconsole` command.

The actual consolidation is done by reading all relevant blocks of data related to the jobs being consolidated – just like it would be at a restore to a point in time – and writing them out as a new backup instead of passing them to a File Daemon as would be done during an actual restoration.

This consolidation or virtual backup job would usually be done outside of the backup window, and it would not involve the actual client at all, so no negative impact on network throughput or client machine load would be caused. However, as all the relevant data has to be read and written, the amount of data that is moved can be quite considerable, and an accordingly long time would be required to finish the consolidation.
Adding Deduplication to the Picture

However, using Bacula Enterprise Edition’s Global Endpoint Deduplication feature, the amount of data that actually is moved can be significantly reduced. This is due to the way that Bacula Enterprise Edition stores deduplicated data inside the deduplication engine only, not on the storage volumes. Backup data that refers to deduplicated data blocks is, on the Volume, stored as just a pointer to a specific block managed by the Storage Daemon’s Deduplication Engine. These pointers are much smaller in size than the actual data is, and thus, the amount of data stored on the Volumes is much smaller.

Now, when Bacula consolidates Jobs to a deduplicating storage device, and the source data is already deduplicated, and the source and target storage devices share the same Deduplication Engine (i.e., they are hosted by the same Storage Daemon), only the pointers to the deduplicated data need to be read and written, nothing else.

Thus, combining Incremental backups, Virtual backups, and Bacula Enterprise Edition’s native Deduplicated Storage, it becomes possible to create Virtual Backups frequently without having to move the full amount of backup data again and again.
Here, each dark red box represents a block of data as part of a full backup, and the lighter red ones represent incremental backup data blocks. The green arrows indicate the process of reading and writing during a virtual backup.

Figure 3: Virtual Backup – Moving Pointers

Here, the blocks represent the much smaller pointers to data stored inside the deduplication engine.
Implementation Advice

Following the above outline, to implement Progressive Virtual Full backups most efficiently, the following items are required:

- A Storage Daemon providing Global Endpoint Deduplication, i.e. running with the corresponding plugin.
- At least two Storage Devices of Device Type = Dedup; Bacula Systems recommends a Bacula-native virtual Autochanger for this purpose.
- A script to initiate the required consolidation Job, or, more general, a way to select the Jobs to consolidate. Bacula Systems can provide a template solution and help adapt to the exact requirements.

In addition, some limitations and considerations need to be kept in mind:

- Global Endpoint Deduplication will need additional memory and disk capacity, and will cause much higher rates of random disk I/O than standard backups.
- Virtual Backups are not compatible with all plugin-generated backup data; Bacula Systems’ support is available to assist in determining if, for any specific environment, this backup scheme is appropriate.
- For long runs of Incrementals, it is advisable to use Accurate = Yes, which will increase the load on the client during backups.
- With Global Endpoint Deduplication mode both sides, the network and client load during backups will be quite different compared to “regular” backups.

An animation example of Progressive Virtual Full can be seen at www.baculasystems.com/ml/pvf3.svg.
For More Information

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

Headquarters
Bacula Systems SA
Rue Galilée 5
CH-1400 Yverdon-les-Bains
Switzerland
Phone: +41 21 641 6080
Fax: +41 21 641 6081

USA
Western Region
3612 Eccles Ave
Ogden, UT 84403

Eastern Region & Canada
269 Carmita Ave
Rutherford, NJ 07070
Toll Free: +1 800 256 0192