Bacula Enterprise Performance Fine Tuning

Bacula Systems Documentation

Contents

Contents

Optimizing the performance of Bacula Enterprise is crucial for achieving optimal data backup performance. To achieve this, it is recommended to fine tune the operating system on which the Director, Storage Daemon, and Catalog are running.

Note: This document highlights the settings for optimal performance of the Bacula Catalog database and the Bacula Director. If the Bacula Catalog database is stored on a remote PostgreSQL server/cluster, follow the tips in this document to fine-tune the PostgreSQL server host.

When dealing with large amounts of data, especially in the range of several terabytes or more, it is important to also consider the configuration of the Storage Daemon. The server-configuration-1 section provides valuable insights and should be taken into account.

Furthermore, it is advisable to consult the performance advice provided in the dedicated plugin articles, depending on the plugins implemented. Specifically, the Virtualization, the Microsoft 365 Plugin and the Global Endpoint Deduplication products should be carefully considered in that regards.

1 Performance Tuning of Director and Catalog Host

This document presents guidelines for tuning the performance of Bacula Director and Catalog Host.

1.1 Introduction

Why Choose PostgreSQL for Bacula Enterprise?

- Restore and Accurate backup run faster with fewer resources.
- Jobs can despool attributes faster and in parallel.
- Bacula Systems support team knows PostgreSQL very well.
- PostgreSQL has no data corruption after a crash.
- BVFS and BWeb restore components run faster on PostgreSQL.
- Bacula Enterprise Customers can be eligible to access dedicated PostgreSQL experts.

Which PostgreSQL Version to Use?

All the supported major PostgreSQL versions are supported for Bacula Enterprise Catalog. Please install the latest available minor release for whatever major version you choose. Given the improvements newer PostgreSQL versions bring, the latest stable PostgreSQL version for which binary packages are available makes a good choice. We do not recommend compiling the server locally. Please use the latest PostgreSQL verson available with the operating system repository as it is the one tested by Bacula Systems QA team. If in doubt, please get in touch with **Bacula Systems** support.

1.2 Director and Catalog Configuration

Disks Considerations

Catalog Size

For 1 billion objects, the Catalog will use around 300 GB to 400 GB of space. The number of objects depends on your retention period requirements. Bacula will prune file records, thus reducing the number of database objects and the database space used, after the retention period you specify. For common operations, the database needs some free space (between 10 and 20 GB) in the temporary disk area. For best performance, we advise you to stay under 80% of used space on your filesystems.

RAID Configuration

Bacula Systems advises using a RAID configuration that provides write cache capability, which will usually require a hardware RAID HBA protected with a battery. Typically RAID 10 will provide the best performance and redundancy. Today, using SSDs to build the database storage system is a reasonable approach to achieve the best performance.

Whenever using a hardware RAID controller, it is essential to have a battery-backed cache when write caching is configured, which is advised for best performance. **Bacula Systems** recommends using a dedicated HBA known for good performance for the database disk system.

Disks Layout

As with all database systems, you can optimize disk throughput using different physical devices for your database. Generally, you can get performance enhancements by using different physical disks for Write Ahead Log (`pg_wal`) files and data files. Writes to the WAL are sequential, and it is generally only read during recovery or for replication. Data files (heap and index) I/O are often random and it makes sense to optimize data storage for fast writes.

Filesystems

A modern filesystem is recommended for all PostgreSQL data, XFS, EXT4 and ZFS are most used. Old filesystems like ext2/ext3 should be avoided.

Using the `deadline` or `noop` disk scheduler can also improve throughput significantly. Please note that some recent systems offer `mq-deadline` now:

```
# echo deadline > /sys/block/sda/queue/scheduler
```

To use it for all disks on boot, add the `elevator=deadline` option to the kernel command line of the boot loader (for GRUB, this is usually `/boot/grub/grub.conf`).

PostgreSQL executes system fsync calls to flush data to disk. With a properly set up battery-backed RAID controller providing non-volatile cache write barriers can be turned off. Mounting the volume with the `nobarrier` option disables barriers for the volume.

Testing Hardware

For optimal performance, you will need to iterate configuration changes and tests on your server a few times before using it in production. Tests with tools such as `bonnie++` or `iozone` can be very useful.

Memory Considerations

Depending on your configuration, you can need considerable amounts of memory for your Catalog server. PostgreSQL logs give you information on the memory used by Bacula. It's rather easy to know when you need to add more memory.

The amount of memory needed depends a lot on your production needs. If you are planning to handle multi-million file jobs within a very small backup window, the Catalog server will need a fast storage system and lots of memory.

On servers with large memory, you need to adjust the kernel write cache to ensure smooth flushing to the disk.

```
# cat /etc/sysctl.conf
vm.dirty_ratio = 2
vm.dirty_background_ratio = 1

vm.swappiness = 1
vm.zone_reclaim_mode = 0

# sysctl -p
```

If the server has a large amount of memory, we advise disabling the kernel overcommit.

```
# tail /etc/sysctl.conf

vm.overcommit_memory = 2

# sysctl -p
```

Server Monitoring

We recommend that you install `sysstat` tools to have CPU, disk, and network usage information available if fine-tuning becomes necessary.

Additional monitoring of CPU and memory usage, disk space availability, and disk system availability (like I/O wait times or queue lengths) can also be desirable. A network monitoring system like Nagios or Zabbix can help with monitoring, logging, and alerting.

CPU Considerations

The number of CPUs depends on the number of jobs that you want to run concurrently, you can expect to run 5 to 10 jobs in parallel per CPU core smoothly.

Server Configuration

To run a large number of concurrent jobs, you need to configure the `nofile` and `nproc` `ulimit` setting for the `root`, the `bacula` and `postgres` unix account.

```
# cat /etc/security/limits.conf
root soft nofile 10000
root hard nofile 10000

root soft nproc 10000
root hard nproc 10000

bacula soft nofile 10000
bacula hard nofile 10000

bacula soft nproc 10000
bacula hard nproc 10000
```

Having 200 jobs in parallel consumes an average of 1024 open files and 500 threads. Multiply the number of jobs in parallel with 50 for the limit of open files. On some Linux distributions, the services will not read the `/etc/security/limits.conf` configuration file to set limits, for example, with `systemd`, the settings are in the set in the service file. **Bacula Systems** recommends configuring the limit of open files in both Director and Storage Daemon(s) hosts, considering the number of parallel jobs that will be run.

1.3 PostgreSQL Configuration

On Debian / Ubuntu, configuration files are located in `/etc/postgresql/` (check the local system documentation for information relevant to your installed version of OS and database), on RHEL, they are by default located in `/var/lib/pgsql/data`.

Server Configuration

Tuning the PostgreSQL configuration is very important to have decent performance with Bacula. The default configuration values are small and not suitable for production. In several large **Bacula** environments, some settings were proven which result in a good performance of the Catalog database with **Bacula's** specific workload.

As usual, fine-tuning here can not be generalized. However, the way PostgreSQL works is to rely on the operating system to keep data files quickly available (i. e. in cache), instead of loading all data into allocated memory. This allows the operating system to balance caching of data, buffering "dirty" data before committing to disk, and satisfying any other memory requests. Trying to balance this statically in the database configuration is most likely less flexible and will, in general, decrease the overall system performance. Thus, the most important aspect is to leave enough memory available to the OS, but give the database enough memory to do more complex work. Restricting the `shared_buffers` size is one of the essential elements of this configuration. For large **Bacula** instances, we can assume that the complete Catalog can never be held in memory, so it's reasonable to leave more flexibility. For Catalog sizes that can be expected to completely fit into memory, leaving enough headroom for the OS and other processes, increasing the `shared_buffers` may be reasonable.

For a server with more than 4 GB of memory, please use the following configuration in `postgresql.conf`

```
shared_buffers = 1GB                # 128MB by default is low
                                    # 25% of RAM is generaly advised

work_mem = 64MB                     # 4MB by default is low

maintenance_work_mem = 256MB        # 1GB for systems with more than 16GB of RAM
                                    # 2GB for systems with more than 32GB of RAM

effective_cache_size = 3GB          # 75% of available RAM

min_wal_size = 1GB
max_wal_size = 4GB

checkpoint_timeout = 20min
checkpoint_completion_target = 0.9
checkpoint_warning = 90s

log_checkpoints = on
log_temp_files = 80000

autovacuum_vacuum_cost_delay = 5ms # should be 5ms at most on raid array
cursor_tuple_fraction = 1.0        # used for cursors to retrieve all rows

random_page_cost = 1.1             # if SSDs are used
                                   # the default 4.0 is good for HDDs
```

It is important also to adjust the max_connections parameter when running several simultaneous concurrent jobs. The default value for this parameter is max_connections=100. This value should be 2 times larger than Director's Maximum Concurrent Jobs plus connections for each Console. Please note if you have this value already adjusted due to other applications using the same PostgreSQL environment as Bacula, you should increase the current value by the number of Maximum Concurrent Jobs you have configured for your Director.

Considering you have already max_connections=1000 and you have installed a Bacula Director with "Maximum Concurrent Jobs=300" and "Maximum Console Connections=20", you should adjust max_connections to a value higher than 1620. For example:

```
max_connections = 1620
```

Security and Access Configuration

The `pg_hba.conf` file should allow your `bacula` user to login tn your database. (Adapt the `host` line for your remote access).

```
local   bacula bacula                            md5
host    bacula bacula       192.168.0.10/32      md5
```

To apply this change, you must reload the PostgreSQL postmaster process. Since PostgreSQL version 10, a more secure authentication mechanism is available, `SCRAM-SHA-256`. However, if md5 is specified as a method in pg_hba.conf but the user's password on the server is encrypted for SCRAM, then SCRAM-based authentication will automatically be chosen instead. To control password-encryption set `password_encryption` configuration parameter before setting the password for the user. To set the bacula user password, use `\password` command in PSQL.

```
$ psql
postgres=# \password bacula
Enter new password:
Enter it again:
```

The next step will be to move to the psql-catalog-administration.