Host configuration

Enterprise applications

NetApp
May 08, 2024

Table of Contents

Host configuration

MySQL containerzation

Containerization of MySQL databases is becoming more prevalent.

Low-level container management is almost always performed through Docker. Container management platforms such as OpenShift and Kubernetes make management of large container environments even simpler. The benefits of containerization include lower costs, because there is no need to license a hypervisor. Also, containers allow multiple databases to run isolated from one another while sharing the same underlying kernel and operating system. Containers can be provisioned in microseconds.

NetApp offers Astra Trident to provide advanced management capabilities of storage. For example, Astra Trident allows a container created in Kubernetes to automatically provision its storage on the appropriate tier, apply export policies, set snapshot policies, and even clone one container to another. For additional information, see the Astra Trident documentation.

MySQL and NFSv3 slot tables

NFSv3 performance on Linux depends on a parameter called `tcp_max_slot_table_entries`.

TCP slot tables are the NFSv3 equivalent of host bus adapter (HBA) queue depth. These tables control the number of NFS operations that can be outstanding at any one time. The default value is usually 16, which is far too low for optimum performance. The opposite problem occurs on newer Linux kernels, which can automatically increase the TCP slot table limit to a level that saturates the NFS server with requests.

For optimum performance and to prevent performance problems, adjust the kernel parameters that control the TCP slot tables.

Run the `sysctl -a | grep tcp.*.slot_table` command, and observe the following parameters:

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

All Linux systems should include `sunrpc.tcp_slot_table_entries`, but only some include `sunrpc.tcp_max_slot_table_entries`. They should both be set to 128.

Caution
Failure to set these parameters may have significant effects on performance. In some cases, performance is limited because the linux OS is not issuing sufficient I/O. In other cases, I/O latencies increases as the linux OS attempts to issue more I/O than can be serviced.

I/O schedulers and MySQL

The Linux kernel allows low-level control over the way that I/O to block devices is

scheduled.

The defaults on various distributions of Linux vary considerably. MySQL recommends that you use `NOOP` or a `deadline` I/O scheduler with native asynchronous I/O (AIO) on Linux. In general, NetApp customers and internal testing show better results with NoOps.

MySQL's InnoDB storage engine uses the asynchronous I/O subsystem (native AIO) on Linux to perform read-ahead and write requests for data file pages. This behavior is controlled by the `innodb_use_native_aio` configuration option, which is enabled by default. With native AIO, the type of I/O scheduler has greater influence on I/O performance. Conduct benchmarks to determine which I/O scheduler provides the best results for your workload and environment.

See the relevant Linux and MySQL documentation for instructions on configuring the I/O scheduler.

MySQL file descriptors

To run, the MySQL server needs file descriptors, and the default values are not sufficient.

It uses them to open new connections, store tables in the cache, create temporary tables to resolve complicated queries, and access persistent ones. If mysqld is not able to open new files when needed, it can stop functioning correctly. A common symptom of this issue is error 24, "Too many open files." The number of file descriptors mysqld can open simultaneously is defined by the `open_files_limit` option set in the configuration file (`/etc/my.cnf`). But `open_files_limit` also depends on the limits of the operating system. This dependency makes setting the variable more complicated.

MySQL cannot set its `open_files_limit` option higher than what is specified under `ulimit 'open files'`. Therefore, you need to explicitly set these limits at the operating system level to allow MySQL to open files as needed. There are two ways to check the file limit in Linux:

- The `ulimit` command quickly gives you a detailed description of the parameters being allowed or locked. The changes made by running this command are not permanent and will erase after a system reboot.
- Changes to the `/etc/security/limit.conf` file are permanent and are not affected by a system reboot.

Make sure to change both the hard and soft limits for user mysql. The following excerpts are from the configuration:

```
mysql hard nofile 65535
mysql soft nofile 65353
```

In parallel, update the same configuration in `my.cnf` to fully use the open file limits.

Copyright information

Trademark information