Red Hat Directory Server 10

Performance Tuning Guide

Updated for Directory Server 10.3

Last Updated: 2018-10-24

Red Hat Directory Server 10 Performance Tuning Guide

Updated for Directory Server 10.3

Marc Muehlfeld
Red Hat Customer Content Services
mmuehlfeld@redhat.com

Petr Bokoč
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

Legal Notice

Abstract

This guide provides tips for improving server and database performance.

Table of Contents

CHAPTER 1. INTRODUCTION TO DIRECTORY SERVER PERFORMANCE TUNING

This article provides some procedures and options that administrators can use to optimize the performance of their Red Hat Directory Server deployments. Performance tuning a Directory Server instance is unique to each server because of differences for every server in its machine environment, directory size and data type, load and network use, even the types of operations that users and clients perform.

The purpose of this article is to highlight the features that Red Hat Directory Server provides for tracking and assessing server and database performance. There are also some procedures given to help tune server performance. For more in-depth planning information, however, check out the *Red Hat Directory Server Deployment Guide*, and for exhaustive command-line and UI-based administrative instructions, see the *Red Hat Directory Server Administration Guide*.

1.1. SETTING GOALS FOR DIRECTORY SERVER PERFORMANCE

Performance tuning is simply a way to identify potential (or real) bottlenecks in the normal operating environment of the server and then taking steps to mitigate those bottlenecks.

The general plan for performance tuning is:

1. Assess the environment. Look at everything around the Directory Server: its usage, the load, the network connection and reliability, most common operations, the physical machine its on, along with any services competing for its resources.

2. Measure the current Directory Server performance and establish baselines.

3. Identify the server areas which can be improved.

4. Make any changes to the Directory Server settings and, potentially, to the host machine.

5. Measure the Directory Server performance again to see how the changes affected the performance.

Directory Server provides some sort of monitoring in three areas:

- The server process (counters and logs)
- The databases (counters)
- Any database links (counters)

In the Directory Server, most performance measurements are going to be how well the Directory Server retrieves and delivers information to clients. With that in mind, these are the server areas that can be tuned for the best Directory Server performance (and these are the areas covered in this article):

- Search operations
- Indexing performance (which affects both search and write operations)
- Database transactions
- Database and entry cache settings
- Database links

Other changes can be made to the host machine's settings or hardware which can also affect Directory Server performance:

- Available memory (based on directory size)
- Other servers running on the same machine (which could compete for resources)
- Distributing user databases across other Directory Server instances on other machines
- Balancing server loads due to network performance

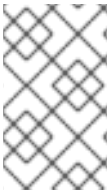
These changes relate much more to planning an effective Directory Server deployment than changes that can be made to an instance. Reviewing the *Deployment Guide* can provide more detail about how to plan an optimal enterprise deployment.

CHAPTER 2. TRACKING SERVER AND DATABASE PERFORMANCE

Red Hat Directory Server has two methods of recording and tracking performance data: performance counters and logs. Counters are used to determine how well the Directory Server performing, particularly in database performance; logs are used to diagnose any problem areas with server and LDAP operations and configuration.

Performance counters focus on the operations and information of the Directory Server for the server, all configured databases, and database links (chaining databases).

There are three types of logs: access (for client connections), errors (for errors, warnings, and details of events), and audit (changes to Directory Server configuration). The access and error logs run by default (and the errors log is required for the server to run). Audit logging, because of the overhead, must be enabled manually.



NOTE

The access log is buffered. This allows full access logging even with highly loaded servers, but there is a time lag between when the event occurs in the server and when the event is written to the log.

2.1. MONITORING SERVER ACTIVITY

The Directory Server's current activities can be monitored from either the Directory Server Console or the command line. It is also possible to monitor the activity of the caches for all of the database.

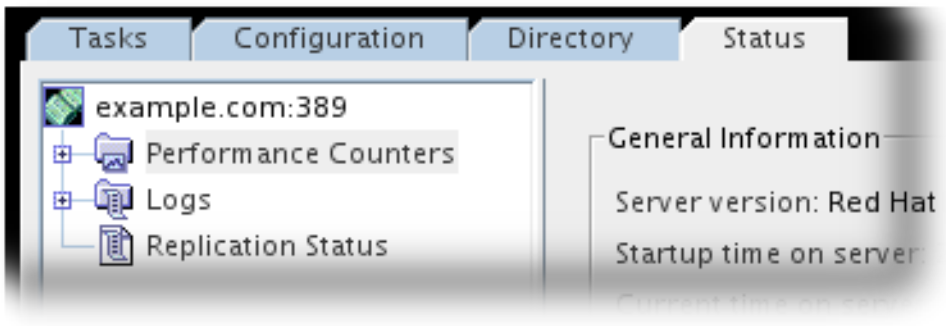
NOTE

Some of the counters for Directory Server database attributes monitored by server use 64-bit integers, even on 32-bit systems (total connections, operations initiated, operations completed, entries sent, and bytes sent). On high-volume systems, this keeps the counters from rolling too quickly and skewing monitoring data.

2.1.1. Monitoring the Server from the Directory Server Console

1. In the Directory Server Console, select the **`Status`** tab.

2. In the navigation tree, select **`Performance Counters`**.



The **`Status`** tab in the right pane displays current information about server activity. If the server is currently not running, this tab will not provide performance monitoring information.

The **General Information** table shows basic information about the server, which helps set a baseline about the statistics that have been gathered.

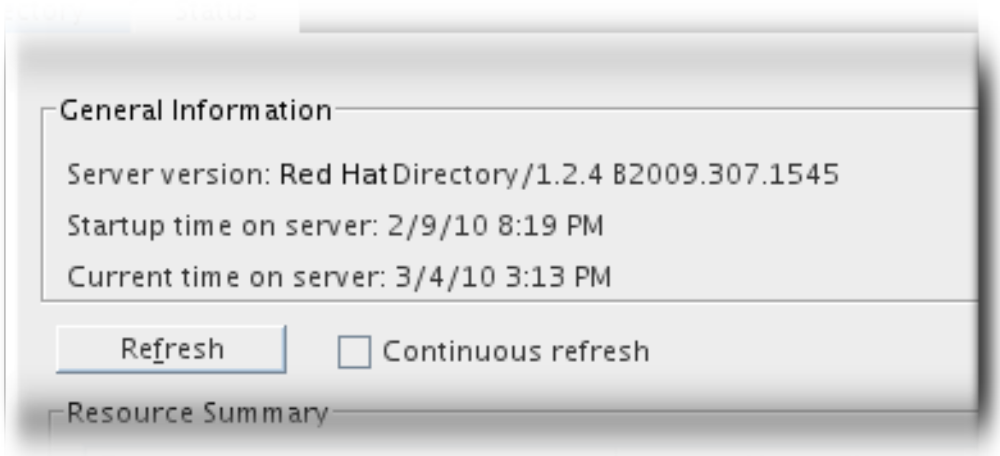


Table 2.1. General Information (Server)

Field	Description
Server Version	Identifies the current server version.
Startup Time on Server	The date and time the server was started.
Current Time on Server	The current date and time on the server.

The **Resource Summary** table shows the totals of all operations performed by that instance.

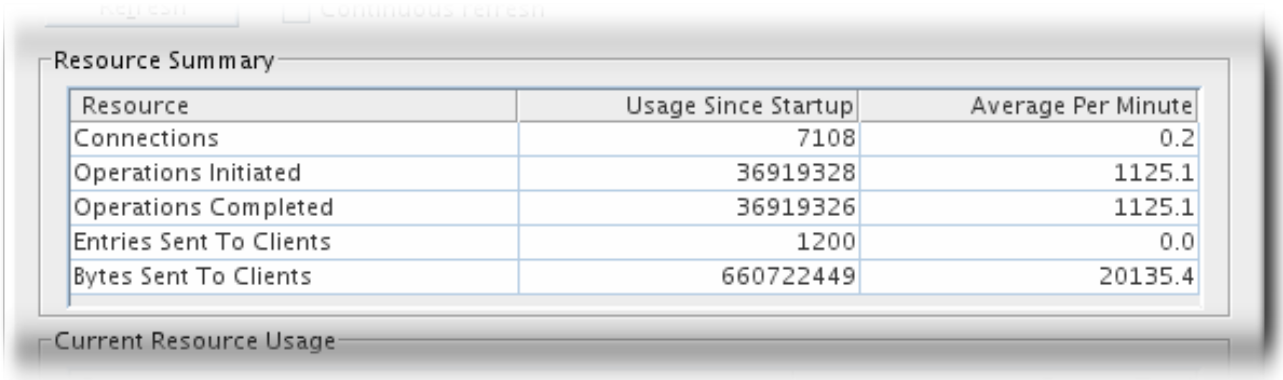


Table 2.2. Resource Summary

Resource	Usage Since Startup	Average Per Minute
Connections	The total number of connections to this server since server startup.	Average number of connections per minute since server startup.

Resource	Usage Since Startup	Average Per Minute
Operations Initiated	The total number of operations initiated since server startup. Operations include any client requests for server action,such as searches, adds, and modifies. Often, multiple operations are initiated for each connection.	Average number of operations per minute since server startup.
Operations Completed	The total number of operations completed by the server since server startup.	Average number of operations per minute since server startup.
Entries Sent to Clients	The total number of entries sent to clients since server startup. Entries are sent to clients as the result of search requests.	Average number of entries sent to clients per minute since server startup.
Bytes Sent to Clients	The total number of bytes sent to clients since server startup.	Average number of bytes sent to clients per minute since server startup.

The **`Current Resource Usage`** table shows the current demands on the server.

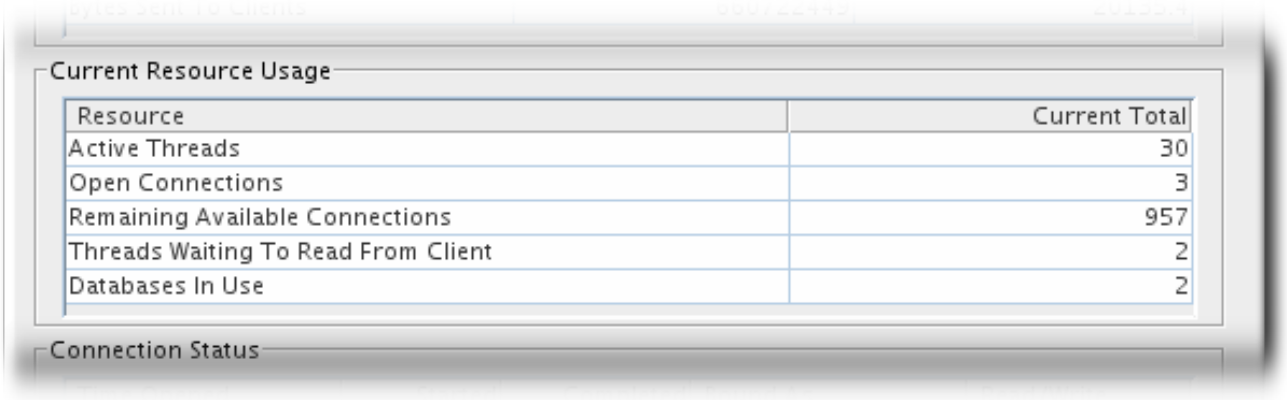


Table 2.3. Current Resource Usage

Resource	Current Total
Active Threads	The current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
Open Connections	The total number of open connections. Each connection can account for multiple operations, and therefore multiple threads.

Resource	Current Total
Remaining Available Connections	The total number of remaining connections that the server can concurrently open. This number is based on the number of currently open connections and the total number of concurrent connections that the server is allowed to open. In most cases, the latter value is determined by the operating system and is expressed as the number of file descriptors available to a task.
Threads Waiting to Write to Client	The total number of threads waiting to write to the client. Threads may not be immediately written when the server must pause while sending data to a client. Reasons for a pause include a slow network, a slow client, or an extremely large amount of information being sent to the client.
Threads Waiting to Read from Client	The total number of threads waiting to read from the client. Threads may not be immediately read if the server starts to receive a request from the client, and then the transmission of that request is halted for some reason. Generally, threads waiting to read are an indication of a slow network or client.
Databases in Use	The total number of databases being serviced by the server.

The **`Connection Status`** table simply lists the current active connections, with related connection information.

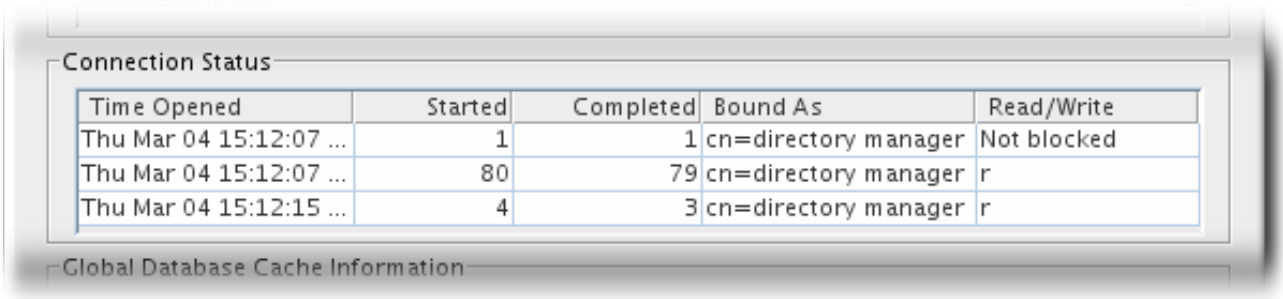
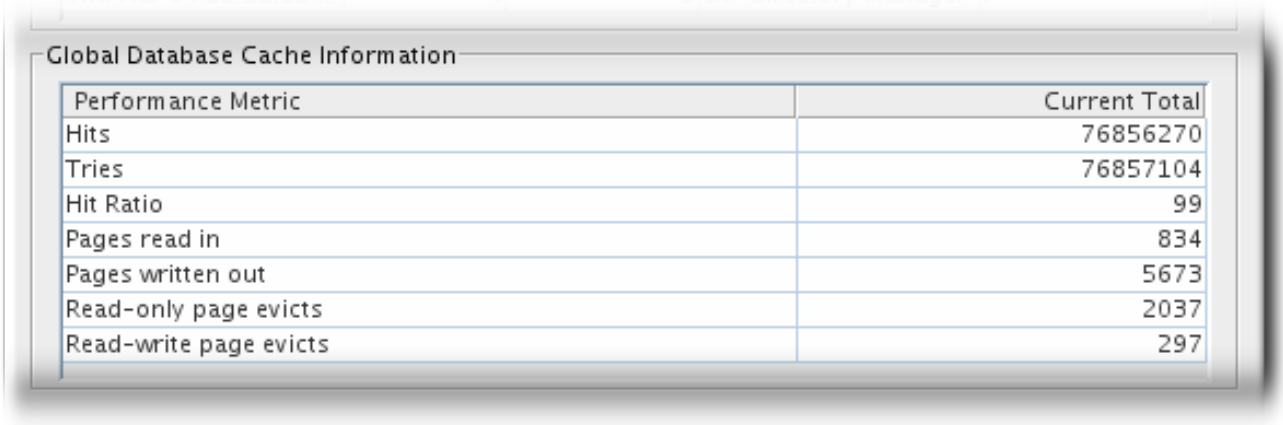


Table 2.4. Connection Status

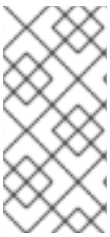
Table Header	Description
Time Opened	The time on the server when the connection was initially opened.
Started	The number of operations initiated by this connection.
Completed	The number of operations completed by the server for this connection.
Bound as	The distinguished name used by the client to bind to the server. If the client has not authenticated to the server, the server displays **`not bound`** in this field.

Table Header	Description
Read/Write	Indicates whether the server is currently blocked for read or write access to the client. There are two possible values: Not blocked means that the server is idle,actively sending data to the client, or actively reading data from the client. Blocked means that the server is trying to send data to the client or read data from the client but cannot. The probable cause is a slow network or client.

The **`Global Database Cache`** table lists the cache information for all databases within the Directory Server instance.


Global Database Cache Information

Performance Metric	Current Total
Hits	76856270
Tries	76857104
Hit Ratio	99
Pages read in	834
Pages written out	5673
Read-only page evicts	2037
Read-write page evicts	297



NOTE

Although the performance counter for the global database cache is listed with the other server performance counters in the Directory Server Console, the actual database cache entries are located and monitored in **`cn=monitor,cn=`*`database_instance`*`,cn=ldbm database,cn=plugins,cn=config`**, as are the other database activities.

Table 2.5. Global Database Cache Information

Table Header	Description
Hits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
Tries	The total number of database accesses since server startup.
Hit Ratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
Pages Read In	The number of pages read from disk into the cache.

Table Header	Description
Pages Written Out	The number of pages written from the cache back to disk.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **`Pages Written Out`** in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts, the better.

2.1.2. Monitoring the Directory Server from the Command Line

The Directory Server's current activities can be monitored using LDAP tools such as **`ldapsearch`**, with the following characteristics:

- Search with the attribute filter **`objectClass=*`**.
- Use the search base **`cn=monitor`**; the monitoring attributes for the server are found in the **`cn=monitor`** entry.
- Use the search scope **`base`**.

For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
-s base -b "cn=monitor" "(objectclass=*)"
```

The monitoring attributes for the Directory Server are found in the **`cn=monitor`** entry.

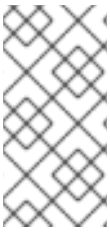
Table 2.6. Server Monitoring Attributes

Attribute	Description
version	Identifies the directory's current version number.
threads	The current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.

Attribute	Description
connection:fd:opentime:opsinitiated:opscompleted:binddn:[rw]	Provides the following summary information for each open connection (only available if you bind to the directory as Directory Manager): *fd* — The file descriptor used for this connection. *opentime* — The time this connection was opened. *opsinitiated* — The number of operations initiated by this connection. *opscompleted* — The number of operations completed. *binddn* — The distinguished name used by this connection to connect to the directory. *rw* — The field shown if the connection is blocked for read or write. By default, this information is available to Directory Manager. However, the ACI associated with this information can be edited to allow others to access the information.
currentconnections	Identifies the number of connections currently in service by the directory.
totalconnections	Identifies the number of connections handled by the directory since it started.
dtablesize	Shows the number of file descriptors available to the directory. Each connection requires one file descriptor: one for every open index, one for log file management, and one for **`ns-slapd`** itself. Essentially, this value shows how many additional condncurrent connections can be serviced by the directory. For more information on file descriptors, see the operating system documentation.
readwaiters	Identifies the number of threads waiting to read data from a client.
opsinitiated	Identifies the number of operations the server has initiated since it started.
opscompleted	Identifies the number of operations the server has completed since it started.
entriessent	Identifies the number of entries sent to clients since the server started.
bytessent	Identifies the number of bytes sent to clients since the server started.
currenttime	Identifies the time when this snapshot of the server was taken. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
starttime	Identifies the time when the server started. The time is displayed in Greenwich Mean Time (GMT) in UTC format.

Attribute	Description
nbackends	Identifies the number of back ends (databases) the server services.
backendmonitordn	Identifies the DN of each directory database.

2.2. MONITORING DATABASE ACTIVITY



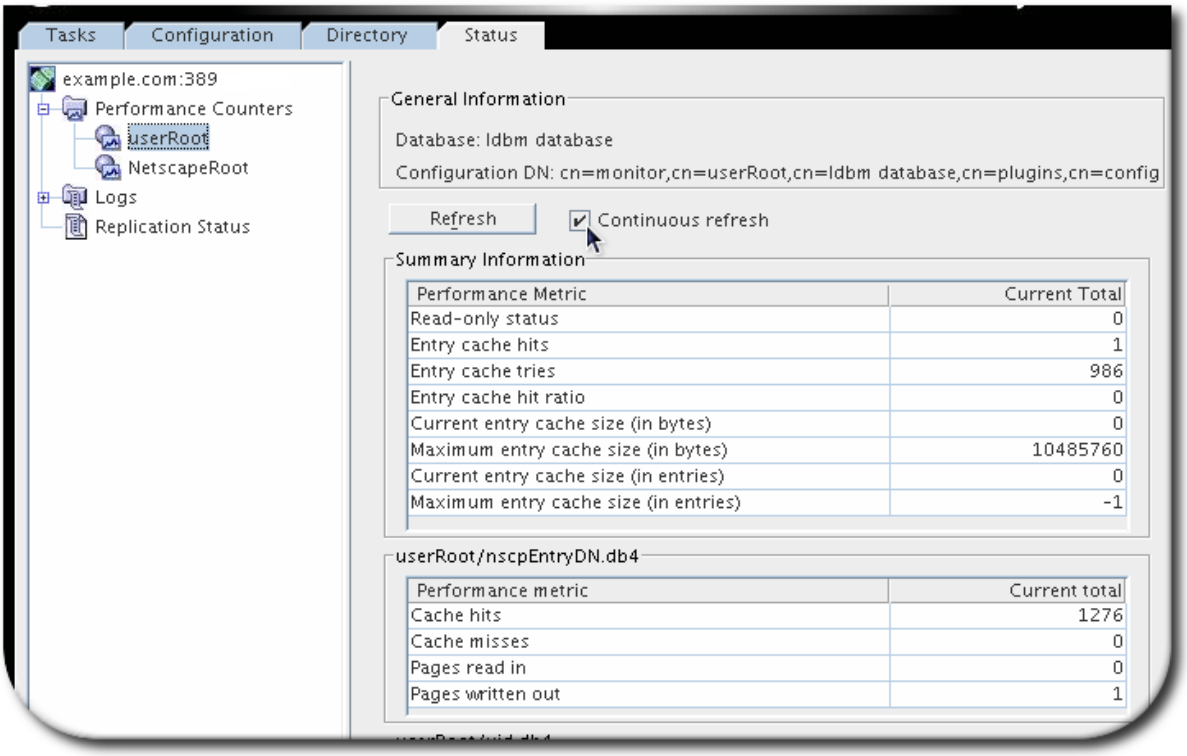
NOTE

Some of the counters for Directory Server database attributes monitored by server use 64-bit integers, even on 32-bit systems (entry cache hits, entry cache tries, the current cache size, and the maximum cache size). On high-volume systems, this keeps the counters from rolling too quickly and skewing monitoring data.

2.2.1. Monitoring Database Activity from the Directory Server Console

1. In the Directory Server Console, select the **`Status`** tab.
2. In the navigation tree, expand the **`Performance Counters`** folder, and select the database to monitor.

   The tab displays current information about database activity. If the server is currently not running, this tab will not provide performance monitoring information.



The **`Summary Information`** section shows the cumulative information for all of the databases being monitored and some cache-related configuration settings which are applied to all databases.

Table 2.7. Summary Information

Performance Metric	Current Total
Read-Only Status	Shows whether the database is currently in read-only mode. The database is in read-only mode when the ***nsslapd-readonly*** attribute is set to **on**.
Entry Cache Hits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
Entry Cache Tries	The total number of entry cache lookups since the directory was last started. That is, the total number of entries requested since server startup.
Entry Cache Hit Ratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever an operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the size of the entry cache by increasing the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
Current Entry Cache Size (in Bytes)	The total size of directory entries currently present in the entry cache.
Maximum Entry Cache Size (in Bytes)	The size of the entry cache maintained by the directory. This value is managed by the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. This is set in the **Memory available for cache** field in the database settings in the Directory Server Console.
Current Entry Cache Size (in Entries)	The number of directory entries currently present in the entry cache.
Maximum Entry Cache Size (in Entries)	*DEPRECATED.* The maximum number of directory entries that can be maintained in the entry cache. Do not attempt to manage the cache size by setting a maximum number of allowed entries. This can make it difficult for the host to allocate RAM effectively. Manage the cache size by setting the amount of RAM available to the cache, using the ***nsslapd-cachememsize*** attribute.

There are many different databases listed for the database monitoring page, by default, because databases are maintained for both entries and indexed attributes. All databases, though, have the same kind of cache information monitored in the counters.

Table 2.8. Database Cache Information

Performance Metric	Current Total
Hits	The number of times the database cache successfully supplied a requested page.
Tries	The number of times the database cache was asked for a page.
Hit Ratio	The ratio of database cache hits to database cache tries. The closer this value is to 100%, the better. Whenever a directory operation attempts to find a portion of the database that is not present in the database cache, the directory has to perform a disk access to obtain the appropriate database page. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory performance drops. To improve this ratio, increase the amount of data that the directory maintains in the database cache by increasing the value of the ***nsslapd-dbcachesize*** attribute. This is the **Maximum Cache Size** database setting in the Directory Server Console.
Pages Read In	The number of pages read from disk into the database cache.
Pages Written Out	The number of pages written from the cache back to disk. A database page is written to disk whenever a read-write page has been modified and then subsequently deleted from the cache. Pages are deleted from the database cache when the cache is full and a directory operation requires a database page that is not currently stored in cache.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **Pages Written Out** in that these are discarded read-write pages that have not been modified.

Table 2.9. Database File-Specific

Performance Metric	Current Total
Cache Hits	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
Cache Misses	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
Pages Read In	The number of pages brought to the cache from this file.
Pages Written Out	The number of pages for this file written from cache to disk.

2.2.2. Monitoring Database Activity from the Command Line

A database's current activities can be monitored using LDAP tools such as **ldapsearch**. The search targets the monitoring subtree of the LDBM database entry, **cn=monitor,cn=***database_name*,**cn=ldbm database,cn=plugins,cn=config**. This contains all of the monitoring attributes for the that specific database instance.

For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
-s base -b "cn=monitor,cn=database_name,cn=ldbm
database,cn=plugins,cn=config" "(objectclass=*)"
```

Table 2.10. Database Monitoring Attributes

Attribute	Description
database	Identifies the type of database currently being monitored.
readonly	Indicates whether the database is in read-only mode; **0** means that the server is not in read-only mode, **1** means that it is in read-only mode.
entrycachehits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
entrycachetries	The total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against the server since server startup.
entrycachehitratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever a search operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the size of the entry cache by increasing the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
currententrycachesize	The total size, in bytes, of directory entries currently present in the entry cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.

Attribute	Description
maxentrycachesize	The maximum size, in bytes, of directory entries that can be maintained in the entry cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
dbcachehits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
dbcachetries	The total number of database accesses since server startup.
dbcachehitratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
dbcachepagein	The number of pages read from disk into the cache.
dbcachepageout	The number of pages written from the cache back to disk.
dbcacheroevict	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
dbcacherwevict	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **Pages Written Out** in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
dbfilename-*number*	The name of the file. *number* provides a sequential integer identifier (starting at 0) for the file. All associated statistics for the file are given this same numerical identifier.
dbfilecachehit-*number*	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
dbfilecachemiss-*number*	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
dbfilepagein-*number*	The number of pages brought to the cache from this file.
dbfilepageout-*number*	The number of pages for this file written from cache to disk.

Attribute	Description
currentdncachesize	The total size, in bytes, of DNs currently present in the DN cache. To increase the size of the entries which can be present in the DN cache, increase the value of the ***nsslapd-dncachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database.
maxdncachesize	The maximum size, in bytes, of DNs that can be maintained in the DN cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-dncachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database.
currentdncachecount	The number of DNs currently present in the DN cache.

2.3. MONITORING DATABASE LINK ACTIVITY

The activity for database links (chained databases) can also be viewed, but only through the command line using **ldapsearch** to return the monitoring attributes that are required. The monitoring attributes are stored in the **cn=monitor,cn=***database_link_name*,**cn=chaining database,cn=plugins,cn=config**.

For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
-s sub -b "cn=monitor,cn=DBLink1,cn=chaining
database,cn=plugins,cn=config" "(objectclass=*)" nsAddCount
```

Table 2.11. Database Link Monitoring Attributes

Attribute Name	Description
nsAddCount	The number of add operations received.
nsDeleteCount	The number of delete operations received.
nsModifyCount	The number of modify operations received.
nsRenameCount	The number of rename operations received.
nsSearchBaseCount	The number of base-level searches received.
nsSearchOneLevelCount	The number of one-level searches received.
nsSearchSubtreeCount	The number of subtree searches received.
nsAbandonCount	The number of abandon operations received.

Attribute Name	Description
nsBindCount	The number of bind request received.
nsUnbindCount	The number of unbinds received.
nsCompareCount	The number of compare operations received.
nsOperationConnectionCount	The number of open connections for normal operations.
nsBindConnectionCount	The number of open connections for bind operations.

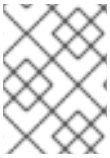
2.4. MONITORING THE LOCAL DISK FOR GRACEFUL SHUTDOWN

When the disk space available on a system becomes too small, the Directory Server process (**slapd**) crashes. Any abrupt shutdown runs the risk of corrupting the database or losing directory data.

It is possible to monitor the disk space available to the **slapd** process. A disk monitoring thread is enabled using the ***nsslapd-disk-monitoring*** configuration attribute. This creates a monitoring thread that wakes every ten (10) seconds to check for available disk space in certain areas.

If the disk space approaches a defined threshold, then the **slapd** begins a series of steps (by default) to reduce the amount of disk space it is consuming:

- Verbose logging is disabled.
- Access logging and error logging are disabled.
- Rotated (archived) logs are deleted.



NOTE

Error log messages are always recorded, even when other changes are made to the logging configuration.

If the available disk space continues to drop to half of the configured threshold, then the **slapd** begins a graceful shut down process (within a grace period); and if the available disk space ever drops to 4KB, then the **slapd** process shuts down immediately. If the disk space is freed up, then the shutdown process is aborted, and all of the previously disabled log settings are re-enabled.

By default, the monitoring thread checks the configuration, transaction log, and database directories. An additional attribute (***nsslapd-disk-monitoring-logging-critical***) can be set to include the logs directory when evaluating disk space.

Disk monitoring is disabled by default, but it can be enabled and configured by adding the appropriate configuration attributes to the **cn=config** entry. Table 2.12, “Disk Monitoring Configuration Attributes” lists all of the configuration options.

1. Using **ldapmodify**, add the disk monitoring attributes. At a minimum, turn on the ***nsslapd-disk-monitoring*** attribute to enable disk monitoring. The default threshold is 2MB; this can be configured (optionally) in the ***nsslapd-disk-monitoring-threshold*** attribute.

For example:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=config
changetype: modify
add: nsslapd-disk-monitoring
nsslapd-disk-monitoring: on
-
add: nsslapd-disk-monitoring-threshold
nsslapd-disk-monitoring-threshold: 3000000
-
add: nsslapd-disk-monitoring-grace-period
nsslapd-disk-monitoring-grace-period: 20
```

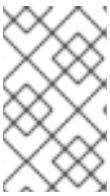
2. Restart the Directory Server to load the new configuration.

```
[root@server ~]# systemctl restart dirsrv.target
```

Table 2.12. Disk Monitoring Configuration Attributes

Configuration Attribute	Description
nsslapd-disk-monitoring	Enabled disk monitoring. This is the only required attribute, since the other configuration options have usable defaults.
nsslapd-disk-monitoring-grace-period	Sets a grace period to wait before shutting down the server after it hits half of the disk space limit. This gives an administrator time to address the situation. The default value is 60 (minutes).
nsslapd-disk-monitoring-logging-critical	Sets whether to shut down the server if the log directories pass the halfway point set in the disk space limit.This prevents the monitoring thread from disabling audit or access logging or from deleting rotated logfiles.
nsslapd-disk-monitoring-threshold	Sets the amount of disk space, in bytes, to use to evaluate whether the server has enough available disk space. Once the space reaches half of this threshold, then the server begins a shut down process. The default value is 2000000 (2MB).

2.5. VIEWING LOG FILES

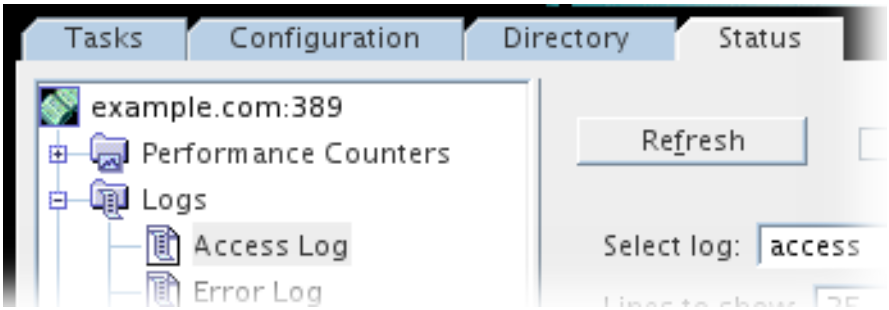


NOTE

The access and error logs are enabled by default and can be viewed immediately. before the audit log can be viewed, audit logging must be enabled for the directory, or the audit log will not be kept.

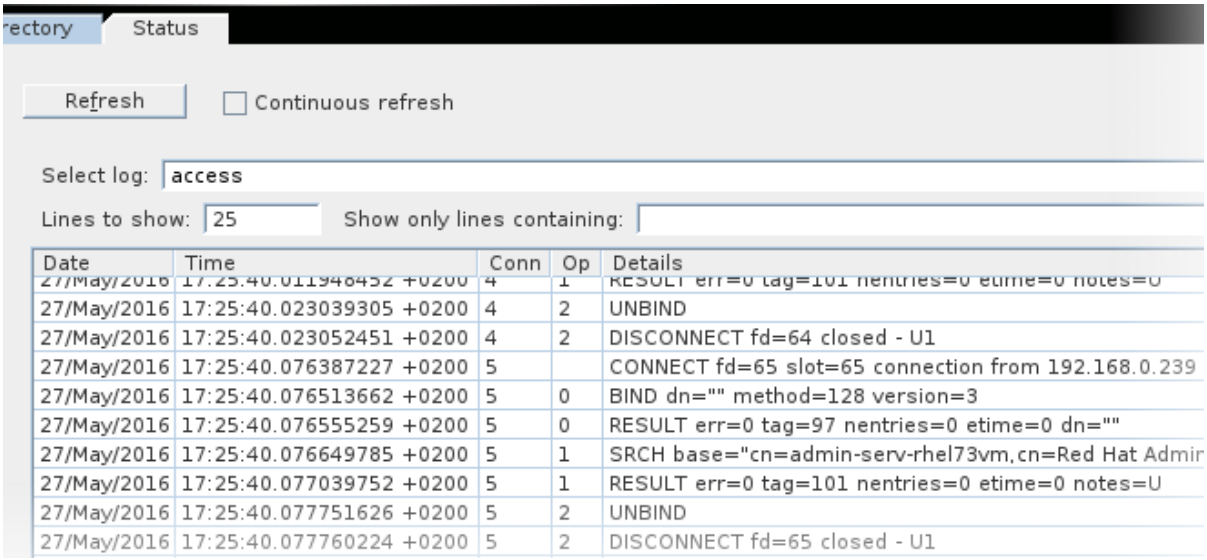
1. In the Directory Server Console, select the **Status** tab.

2. In the navigation tree, expand the **Log** folder. There are three folders available, for the access, error, and audit logs.

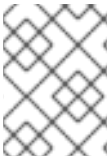


3. When you select the log type to view, a table displays a list of the last 25 entries in the selected log.

4. Optionally, change the settings of the log display and click **Refresh** to update the display.



- The **Select Log** pull-down menu allows you to select an archived (rotated) log rather than the currently active log.
- The **Lines to show** text box changes the number of log entries to display in the window.
- The **Show only lines containing** text box sets a filter, including regular expressions, to use to display only certain matching log entries.



NOTE

Selecting the **Continuous** check box refreshes the log display automatically every ten seconds.= Continuous log refresh does not work well with log files over 10 megabytes.

2.6. REPLACING LOG FILES WITH A NAMED PIPE

Many administrators want to do some special configuration or operation with logging data, like configuring an access log to record only certain events. This is not possible using the standard Directory Server log file configuration attributes, but it is possible by sending the log data to a named pipe, and then using another script or plug-in to process the data. Using a named pipe for the log simplifies these special tasks, like:

- Logging certain events, like failed bind attempts or connections from specific users or IP addresses

- Logging entries which match a specific regular expression pattern
- Keeping the log to a certain length (logging only the last number of lines)
- Sending a notification, such as an email, when an event occurs

The basic format of the script is is:

ds-logpipe.py *named_pipe* [--user *pipe_user*] [--maxlines *number*] [[--serverpidfile *file.pid*] | [--serverpid *PID*]] [--servertimeout *seconds*] [--plugin=*/path/to/plugin.py* | [*pluginfile.arg=value*]]

More detailed usage information is in the *Configuration, Command, and File Reference*.

However, while that has the advantage of being simple to implement and not requiring any Directory Server configuration changes, simply running the script has a big disadvantage: all of the log viewers in the Directory Server Console and any script or tool (such as **logconv.pl**) that expect to access a real file will fail.

If the Directory Server instance will permanently use the named pipe rather than a real file for logging, then it is possible to reconfigure the instance to create the named pipe and use it for logging (as it does by default for the log files). When the Directory Server instance is configured to use the named pipe then all of the log analysis tools — the Directory Server Console and any Directory Server scripts — work fine.

Three things need to be configured for the log configuration for the instance:

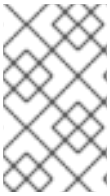
- The log file to use has to be changed to the pipe (***nsslapd-*log***)
- Buffering should be disabled because the script already buffers the log entries (***nsslapd-*log-logbuffering***)
- Log rotation should be disabled so that the server does not attempt to rotate the named pipe (***nsslapd-*log-maxlogsperdir***, ***nsslapd-*log-logexpirationtime***, and ***nsslapd-*log-logrotationtime***)

These configuration changes can be made in the Directory Server Console or using **ldapmodify**.

For example, this switches the access log to **access.pipe**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-accesslog
nsslapd-accesslog: /var/log/dirsrv/slapd-instance/access.pipe
-
replace: nsslapd-accesslog-logbuffering
nsslapd-accesslog-logbuffering: off
-
replace: nsslapd-accesslog-maxlogsperdir
nsslapd-accesslog-maxlogsperdir: 1
-
replace: nsslapd-accesslog-logexpirationtime
nsslapd-accesslog-logexpirationtime: -1
-
replace: nsslapd-accesslog-logrotationtime
nsslapd-accesslog-logexpirationtime: -1
```



NOTE

Making these changes using the `-f` option will cause the server to close the current log file and switch to the named pipe immediately. This can be very helpful for debugging a running server and sifting the log output for specific messages.

2.7. IMPROVING LOGGING PERFORMANCE

Larger server deployments can generate several dozen of megabytes of logs per hour. Depending on the resources available on the server host machine, reconfiguring or disabling access logging can improve system and Directory Server performance.

Before disabling access logging, first configure access log buffering. Buffering writes all log entries directly to the disk, so that the Directory Server performance does not degrade even under a heavy load.

The access log is buffered by default, but make sure the log is using buffering for best performance.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-accesslog-logbuffering
nsslapd-accesslog-logbuffering: on
```

If that does not improve performance, then disable access logging entirely.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-accesslog-enabled
nsslapd-accesslog-enabled: off
```

WARNING

Access logging is extremely helpful for debugging issues in the server and monitoring client connections and failed connection attempts. Don't disable access logging as the normal operating environment.

For alternatives, see Section 2.6, “Replacing Log Files with a Named Pipe”, since using named pipe log scripts can improve performance while still logging information on high performance production servers.

CHAPTER 3. TUNING THE NUMBER OF LOCKS

The lock mechanisms control how many copies of the Directory Server process can be running at once. For example, if there is an import job, then a lock is placed in the **imports/** directory to prevent any other **ns-slapd** (normal), **ldif2db** (another import), or **db2ldif** (export) operations from running. If the server is running as normal, there is a lock in the **server/** directory, which prevents import operations (but not export operations), while if there is an export operation, the lock in the **exports/** directory allows normal server operations but prevents import operations.

If there are error messages indicating that the lock table is out of available locks (for example, **libdb: Lock table is out of available locks**), double the value of the ***nsslapd-db-locks*** attribute in the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry.

For example, if the current value is **10000**, set it to **20000**. If the problem persists, double the number again. To monitor the current and maximum number of locks, do a search on **cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config**. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
-sub -b "cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config"
objectclass=* | grep -- -locks: )
```

For more information on using LDAP utilities, see the *Red Hat Directory Server Administration Guide*.

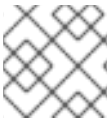
CHAPTER 4. IMPROVING SEARCH PERFORMANCE (AND BALANCING READ PERFORMANCE)

The most effective way to improve search operations against the directory is to configure thorough indexes for entries, combined with reasonable limits on search results.

4.1. USING INDEXES

An index (as it implies) is a tag that shows that a certain entry contains a certain attribute, without having to contain any other detail about the entry (which saves space and makes returning search results faster). Each index is organized around a Directory Server attribute and a certain way of matching that attribute:

- *Presence index (pres)* simply shows what entries contain an attribute.
- *Equality index (eq)* shows which attribute values match a specific search string.
- *Approximate index (approx)* is used for efficient *sounds-like* searches, which shows entries which have a value that phonetically matches a string.
- *Substring index (sub)* matches any substring of an attribute value to the given search string. (This index if very expensive for the server to maintain.)
- *International index* uses a matching rule to match strings in a directory which contains values in languages other than English.
- *Browsing index*, or *virtual list view (VLV) index*, sets an index to use to display entries in the Directory Server Console.



NOTE

Indexing is described in much more detail in the *Administration Guide*.

However, just creating indexes is not ipso facto going to increase server performance. Maintaining indexes puts a burden on the Directory Server for every modify, add, and delete operation by having to verify every attribute in the change against every index maintained by the server:

1. The Directory Server receives an add or modify operation.
2. The Directory Server examines the indexing attributes to determine whether an index is maintained for the attribute values.
3. If the created attribute values are indexed, then the Directory Server generates the new index entries.
4. Once the server completes the indexing, the actual attribute values are created according to the client request.

For example, the Directory Server adds the entry:

```
dn: cn=John Doe, ou=People,dc=example,dc=com
objectclass: top
objectClass: person
objectClass: orgperson
```

```
objectClass: inetorgperson
cn: John Doe
cn: John
sn: Doe
ou: Manufacturing
ou: people
telephoneNumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.
```

The Directory Server is maintaining the following indexes:

- Equality, approximate, and substring indexes for ***cn*** (common name) and ***sn*** (surname) attributes.
- Equality and substring indexes for the telephone number attribute.
- Substring indexes for the description attribute.

When adding that entry to the directory, the Directory Server must perform these steps:

1. Create the ***cn*** equality index entry for **`John`** and **`John Doe`**.
2. Create the appropriate ***cn*** approximate index entries for **`John`** and **`John Doe`**.
3. Create the appropriate ***cn*** substring index entries for **`John`** and **`John Doe`**.
4. Create the ***sn*** equality index entry for **`Doe`**.
5. Create the appropriate ***sn*** approximate index entry for **`Doe`**.
6. Create the appropriate ***sn*** substring index entries for **`Doe`**.
7. Create the telephone number equality index entry for **`408 555 8834`**.
8. Create the appropriate telephone number substring index entries for **`408 555 8834`**.
9. Create the appropriate description substring index entries for **`Manufacturing lead for the Z238 line of widgets`**. A large number of substring entries are generated for this string.

Before creating new indexes, make sure to balance the overhead of maintaining the indexes against the potential improvements in search performance. Especially important, match the *types* of indexes that you maintain to the type of information stored in the directory and the type of information users routinely search for.

- Approximate indexes are not efficient for attributes commonly containing numbers, such as telephone numbers.
- Substring indexes do not work for binary attributes.
- Equality indexes should be avoided if the value is big (such as attributes intended to contain photographs or passwords containing encrypted data).
- Maintaining indexes for attributes not commonly used in a search increases overhead without improving global searching performance.
- Attributes that are not indexed can still be specified in search requests, although the search performance may be degraded significantly, depending on the type of search.

- The more indexes you maintain, the more disk space you require.

NOTE

Creating indexes is much more effective for directories which have a high search operation load and low modify operation load.

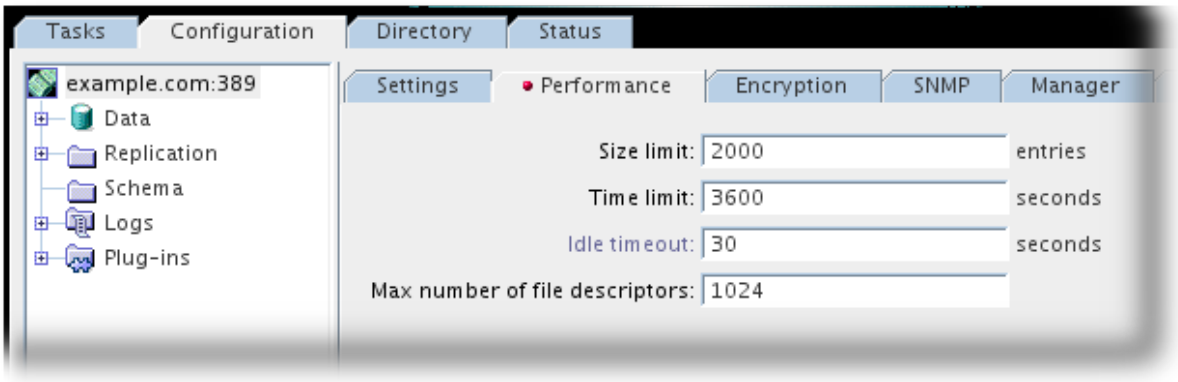
4.2. TUNING DIRECTORY SERVER RESOURCE SETTINGS

The server's performance can be managed and improved by limiting the amount of resources the server uses to process client search requests, which is done by defining four settings:

- The maximum number of entries the server returns to the client in response to a search operation (size limit attribute).
- The maximum amount of real time (in seconds) for the server to spend performing a search request (time limit attribute).
- The time (in seconds) during which the server maintains an idle connection before terminating it (idle timeout attribute).
- The maximum number of file descriptors available to the Directory Server (max number of file descriptors attribute).

To configure Directory Server to optimize performance:

1. In the Directory Server Console, select the **`Configuration`** tab, and then select the topmost entry in the navigation tree in the left pane.
2. Select the **`Performance`** tab in the right pane.



3. Set the maximum number of entries the server will return to the client in response to a search operation by entering a new value in the **`Size Limit`** text box.

   To keep from setting a limit, type **`-1`** in this text box.

4. Enter the maximum amount of real time (in seconds) for the server to spend performing a search request in the **`Time Limit`** text box.

   To keep from setting a limit, type **`-1`** in this text box.

5. Enter the time (in seconds) for the server to maintain an idle connection before terminating it in the **`Idle Timeout`** text box.

To keep from setting a limit, type zero (**0**) in this text box.

6. Set the maximum number of file descriptors available to the Directory Server in the **Max Number of File Descriptors** text box. For more information on this parameter, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

4.3. SETTING INDEX SCAN LIMITS

In large directories, the search results list can get huge. A directory with a million **inetorgperson** entries would have a million entries that were returned with a filter like **(objectclass=inetorgperson)**, and an index for the ***sn*** attribute would have at least a million entries in it.

Loading a long ID list from the database significantly reduces search performance. The configuration parameter, **nsslapd-idlistscanlimit**, sets a limit on the number of IDs that are read before a key is considered to match the entire primary index (meaning the search is treated as an unindexed search with a different set of resource limits).

For large indexes, it is actually more efficient to treat any search which matches the index as an unindexed search. The search operation only has to look in one place to process results (the entire directory) rather than searching through an index that is nearly the size of a directory, plus the directory itself.

The default value of the **nsslapd-idlistscanlimit** attribute is **4000**, which is gives good performance for a common range of database sizes and access patterns. It's usually not necessary to change this value. If the database index is slightly larger than the 4000 entries, but still significantly smaller than the overall directory, then raising the scan limit improves searches which would otherwise hit the default limit of 4000.

On the other hand, lowering the limit can significantly speed up searches that would otherwise hit the 4000 entry limit, but where it is not necessary to scan every entry.

4.4. FINE GRAINED ID LIST SIZE

In large databases, some queries can consume a large amount of CPU and RAM resources. To improve the performance, you can set a default ID scan limit that applies to all indexes in the database using the **nsslapd-idlistscanlimit** attribute. However in some cases it is useful to define a limit for certain indexes, or use no ID list. You can set individual settings for ID list scan limits for different types of search filters using the **nsIndexIDListScanLimit** attribute.

To set a limit, for example for the ***objectClass*** attribute, add the **nsIndexIDListScanLimit** parameter to the DN **cn=objectclass,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config**.

The **nsIndexIDListScanLimit** attribute is multi valued and takes the following list of parameters as a value:

```
nsIndexIDListScanLimit: limit=NNN [type=eq[,sub,...]]
[flags=AND[,XXX,...]] [values=val[,val,...]]
```

- **limit**: The maximum size of the ID list. Valid values are:
  - **-1**: Unlimited.

- **0**: Do not use the index.
- **1 to the maximum 32-bit integer (2147483647)**: Maximum number of IDs.

- **type**: Optional. The type of the index. **eq**, **sub**, **pres**, and so on. The value must be one of the actual ***nsIndexType*** specified for the index definition. For example, you cannot use **type=eq** if you do not have **nsIndexType=eq** defined.
- **flags**: Optional. Flags that alter the behavior of applying the scan limit. Valid values are:
  - **AND**: Apply the scan limit only to searches in which the attribute appears in an **AND** clause.
  - **OR**: Apply the scan limit only to searches in which the attribute appears in an **OR** clause.
- **values**: Optional. Comma separated list of values which must match the search filter in order for the limit to be applied. Since the matches are done one at a time, the values will match if any of the values matches.

  The values must be used with only one type at a time.

  The values must correspond to the index type, and must correspond to the syntax of the attribute to which the index is applied. For example, if you specified the integer based attribute ***uidNumber*** and it is indexed for **eq**, you cannot use **type=eq values=abc**.

  If the value contains spaces, commas, NULL, or other values which require to be escaped, the LDAP filter escape syntax should be used: backslash (\) followed by the 2 hex digit code for the character. In the following example, the commas in the DN value are escaped with **\2C**.

```
nsIndexIDListScanLimit: limit=0 type=eq
values=uid=user\2Cou=People\2Cdc=example\2Cdc=com
```

Example 4.1. Setting nsIndexIDListScanLimit

In a large database with 10 million entries that contain the object class **inetOrgPerson**, a search for **(&(objectClass=inetOrgPerson)(uid=user))** creates first an ID list containing all 10 million IDs matching **objectClass=inetOrgPerson**. When the database applies the second part of the filter, it searches the result list for objects matching **uid=user**. In this cases it is useful to define a limit for certain indexes, or use no ID list at all.

To set that no ID list is created for **objectClass=inetOrgPerson** in **AND** clauses, add the following **nsIndexIDListScanLimit**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -
x
dn: cn=objectclass,cn=index,cn=userRoot,cn=ldbm
database,cn=plugins,cn=config
changetype: modify
replace: nsIndexIDListScanLimit
nsIndexIDListScanLimit: limit=0 type=eq flags=AND values=inetOrgPerson

modifying entry "cn=objectclass,cn=index,cn=userRoot,cn=ldbm
database,cn=plugins,cn=config"
```

> No ID list is created for **objectClass=inetOrgPerson** when used in an **AND** clause. In all other situations the value of ***nsslapd-idlistscanlimit*** is applied.

4.5. TUNING THE DATABASE CACHE FOR SEARCHES

The database attributes that affect search performance mainly define the amount of memory available to the server. The maximum values that can be set for the database's cache size attributes depends on the amount of real memory on the machine. Roughly, the amount of available memory on the machine should always be greater than sum total of the default database cache size and sum of each entry cache size.

Use caution when changing these cache sizing attributes. The ability to improve server performance with these attributes depends on the size of the database, the amount of physical memory available on the machine, and whether directory searches are random (that is, if the directory clients are searching for random and widely scattered directory data).

If the database does not fit into memory and if searches are random, attempting to increase the values set on these attributes does not help directory performance. In fact, changing these attributes may harm overall performance.

The attributes of each database used to store directory data, including the server configuration data in the **NetscapeRoot** database, can be resized.

To improve the cache hit ratio on search operations, increase the amount of data that the Directory Server maintains in the database cache, as described in Section 6.5, “Setting the Database Cache Size”, by editing the values for the ***nsslapd-dbcachesize*** attribute.

4.6. TUNING THE DATABASE SETTINGS FOR SEARCHES

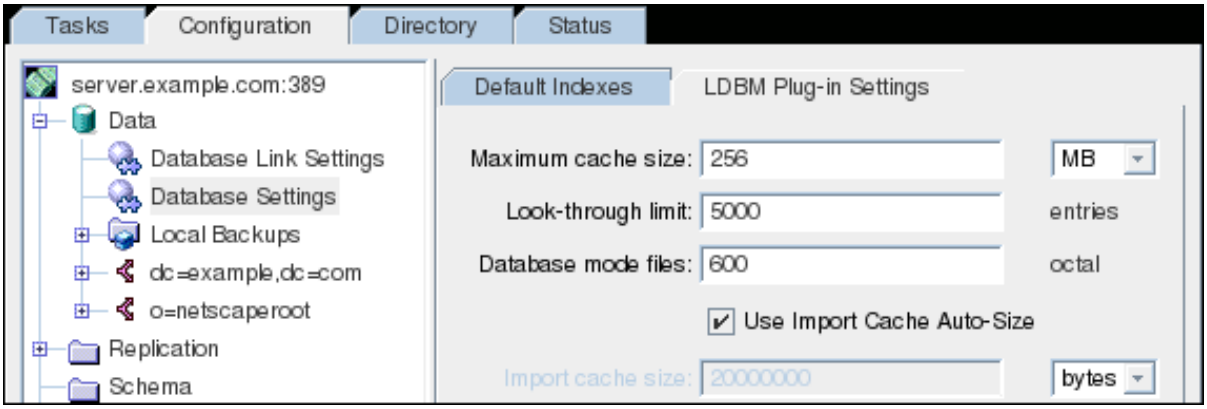
The Directory Server Console only shows the databases that contain the directory data and the **NetscapeRoot** database. However, the server uses another database to manage these. On this database, the following attributes can be changed to improve performance:

- The amount of memory to make available for all databases (maximum cache size), which is described in Section 6.3, “Setting the Entry Cache Size”.
- The maximum number of entries for the server to verify in response to a search request (look-through limit).
- The amount of memory to make available for import (import cache size).

To configure the default database attributes that apply to all other database instances:

1. In the Directory Server Console, select the **Configuration** tab; then, in the navigation tree, expand the **Data** icon, and highlight the **Database Settings** node.
2. Select the **LDBM Plug-in Settings** tab in the right pane.

   This tab contains the database attributes for all databases stored on this server.



3. In the **`Maximum Cache Size`** field, enter a value corresponding to the amount of memory to make available for all databases. This value is for the total of the *entire backend*, meaning all databases cumulatively rather than the amount per single database instance.

4. In the **`Look-Through Limit`** field, enter the maximum number of entries for the server to check in response to a search request.

5. There are two ways to set the amount of memory in bytes to make available for import. The default is to have auto cache sizing, meaning 50% of the free memory is allocated for the import cache. It is also possible to set the import cache size manually by deselecting the **`Use Cache Auto-Size`** check box and then setting the value in the **`Import Cache Size`** field. For creating a very large database from LDIF, set this attribute as large as possible, depending on the memory available on the machine. The larger this parameter, the faster the database is created.

WARNING

Setting this value too high can cause import failures because of a lack of memory.

4.7. MANAGING SPECIAL ENTRIES

The **`cn=config`** entry in the simple, flat **`dse.ldif`** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under **`cn=config`**, performance will probably suffer.

Although Red Hat recommends that simple user entries not be stored under **`cn=config`** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **`cn=config`** since this centralizes configuration information.

CHAPTER 5. TUNING TRANSACTION LOGGING

Every Directory Server contains a transaction log which writes operations for all the databases it manages. Whenever a directory database operation such as a modify is performed, the server creates a single database transaction for all of the database operations invoked as a result of that LDAP operation. This includes both updating the entry data in the entry index file and updating all of the attribute indexes. If *all* of the operations succeed, the server commits the transaction, writes the operations to the transaction log, and verifies that the entire transaction is written to disk. If *any* of these operations fail, the server rolls back the transaction, and all of the operations are discarded. This all-or-nothing approach in the server guarantees that an update operation is *atomic*. Either the entire operation succeeds permanently and irrevocably, or it fails.

Periodically, the Directory Server (through internal housekeeping threads) flushes the contents of the transaction logs to the actual database index files and checks if the transaction logs require trimming.

If the server experiences a failure, such as a power outage, and shuts down abnormally, the information about recent directory changes is still saved by the transaction log. When the server restarts, the directory automatically detects the error condition and uses the database transaction log to recover the database.

Although database transaction logging and database recovery are automatic processes that require no intervention, it can be advisable to tune some of the database transaction logging attributes to optimize performance.

WARNING

The transaction logging attributes are provided only for system modifications and diagnostics. These settings should be changed only with the guidance of Red Hat Technical Support. Setting these attributes and other configuration attributes inconsistently may cause the directory to be unstable.

5.1. MOVING THE DATABASE DIRECTORY TO A SEPARATE DISK OR PARTITION

To achieve higher performance, store the directory server databases and transaction log on a fast drive, such as an SSD.

For example, if you already run a Directory Server instance and want to mount the **/dev/sdb1** partition to the **/var/lib/dirsrv/slapd-*instance_name*/db/** directory:

1. Stop the instance:

```
# systemctl stop dirsrv@instance_name
```

2. Mount the **/dev/sdb1** partition to a temporary directory. For example:

```
# mount /dev/sdb1 /mnt/
```

3. Copy the content of the **/var/lib/dirsrv/slapd-*instance_name*/db/** directory to the temporary mount point:

```
# mv /var/lib/dirsrv/slapd-instance_name/db/* /mnt/
```

4. Unmount the temporary directory:

```
# umount /mnt/
```

5. If **/var/lib/dirsrv/slapd-*instance_name*/db/** is also a separate mount point, unmount the directory:

```
# umount /var/lib/dirsrv/slapd-instance_name/db/
```

6. Update the **/etc/fstab** file to mount the **/dev/sdb1** partition automatically to **/var/lib/dirsrv/slapd-*instance_name*/db/** when the system boots. For details, see the corresponding section in the *Red Hat System Administrator's Guide*.

7. Mount the file system. If you added the entry to **/etc/fstab**:

```
# mount /var/lib/dirsrv/slapd-instance_name/db/
```

8. If SELinux is running in **enforcing** mode, restore the SELinux context:

```
# restorecon -Rv /var/lib/dirsrv/slapd-instance_name/db/
```

9. Start the instance:

```
# systemctl start dirsrv@instance_name
```

5.2. CHANGING THE DATABASE CHECKPOINT INTERVAL

At regular intervals, the Directory Server writes operations logged in the transaction log to the database index files and logs a checkpoint entry in the database transaction log. By indicating which changes have already been written to the database indexes, checkpoint entries indicate where to begin recovery from the transaction log, thus speeding up the recovery process.

By default, the Directory Server is set up to send a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval may increase the performance of directory write operations. However, increasing the checkpoint interval may also increase the amount of time required to recover directory databases after a disorderly shutdown and require more disk space due to large database transaction log files. Therefore, only modify this attribute if you are familiar with database optimization and can fully assess the effect of the change.

To modify the checkpoint interval while the server is running, use the **ldapmodify** command-line utility to add the ***nsslapd-db-checkpoint-interval*** attribute to the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry.

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: modify
```

```
add: nsslapd-db-checkpoint-interval
nsslapd-db-checkpoint-interval: 120
```

For more information on the syntax of the ***nsslapd-db-checkpoint-interval*** attribute, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

5.3. DISABLING DURABLE TRANSACTIONS

Durable transaction logging means that each LDAP update operation, comprised of a sequence of database operations in a transaction, is physically written to disk. Even though each LDAP operation can be comprised of multiple database operations, each LDAP operation is treated as a single database transation. Each LDAP operation is both atomic and durable.

WARNING

Turning off durable transactions can improve Directory Server write performance at the risk of data loss.

When durable transaction logging is disabled, every directory database operation is written to the database transaction log file but may not be physically written to disk immediately. If a directory change was written to the logical database transaction log file but not physically written to disk at the time of a system crash, the change cannot be recovered. When durable transactions are disabled, the recovered database is consistent but does not reflect the results of any LDAP write operations that completed just before the system crash.

By default, durable database transaction logging is enabled. To disable durable transaction logging:

1. Use the **ldapmodify** command-line utility to add the ***nsslapd-db-durable-transactions*** attribute to the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry, and set the value of this attribute to **off**.

   ```
   # ldapmodify -D "cn=Directory Manager" -W -p 389 -h
   server.example.com -x

   dn: cn=config,cn=ldbm database,cn=plugins,cn=config
   changetype: modify
   add: nsslapd-db-durable-transactions
   nsslapd-db-durable-transactions: off
   ```

   For information on the syntax of the ***nsslapd-db-durable-transactions*** attribute, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

2. Restart the Directory Server.

   ```
   # systemctl restart dirsrv.target
   ```

5.4. SPECIFYING TRANSACTION BATCHING

To improve update performance when full transaction durability is not required, use the ***nsslapd-db-transaction-batch-val*** attribute to specify how many transactions will be batched before being committed to the transaction log. Setting this attribute to a value of greater than **0** causes the server to delay committing transactions until the number of queued transactions is equal to the attribute value. This is similar to disabling durable transaction logging (in the ***nsslapd-db-durable-transaction*** attribute), but setting the batch value gives more control over how many transactions can be potentially lost.

To specify or modify transaction batching while the server is running, use the **ldapmodify** command-line utility to add the ***nsslapd-db-transaction-batch-val*** attribute to the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: modify
add: nsslapd-db-transaction-batch-val
nsslapd-db-transaction-batch-val: 1
```

For more information on the syntax and values of the ***nsslapd-db-transaction-batch-val*** attribute, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

CHAPTER 6. MANAGING THE DATABASE CACHE SETTINGS

Directory Server uses the following caches:

- The *Entry cache*, which contains individual directory entries.
- The *DN cache* is used to associate DNs and RDNs with entries.
- The *Database cache*, which contains the database index files ***.db** and ***.db4** files.

For the highest performance improvements, all cache sizes must be able to store all of their records. If you do not use the recommended auto-sizing feature and have not enough RAM available, assign free memory to the caches in the previously shown order.

6.1. THE DATABASE AND ENTRY CACHE AUTO-SIZING FEATURE

Directory Server can automatically determine the optimized size for the database and entry cache. In instances created using Directory Server 10.1.1 or later, auto-sizing is enabled by default. Auto-sizing optimizes the size of both caches based on the hardware resources of the server when the instance starts.

If you upgraded from a version earier than 10.1.1 or if you manually set a database or entry cache size, re-enable auto-sizing for optimized performance. For details, see Section 6.1.1, “Manually Re-enabling the Database and Entry Cache Auto-sizing”.

IMPORTANT

Red Hat recommends to use the auto-tuning settings. Do not set the entry cache size manually.

6.1.1. Manually Re-enabling the Database and Entry Cache Auto-sizing

If you upgraded the instance from a version prior to 10.1.1, or previously manually set an entry cache size, you can enable the auto-tuning for the entry cache.

The following parameters in the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry control the auto-sizing:

nsslapd-cache-autosize

This settings controls if auto-sizing is enabled for the database and entry cache. Auto-sizing is enabled:

- For both the database and entry cache, if the ***nsslapd-cache-autosize*** parameter is set to a value greater than **0**.
- For the database cache, if the ***nsslapd-cache-autosize*** and ***nsslapd-dbcachesize*** parameters are set to **0**.
- For the entry cache, if the ***nsslapd-cache-autosize*** and ***nsslapd-cachememsize*** parameters are set to **0**.

nsslapd-cache-autosize-split

The value sets the percentage of RAM that is used for the database cache. The remaining percentage is used for the entry cache.

More than 512 MB RAM database cache do not improve the performance. Therefore, the database cache is limited to 512 MB.

To enable the database and entry cache auto-sizing:

1. Stop the Directory Server instance:

```
# systemctl stop dirsrv@instance_name
```

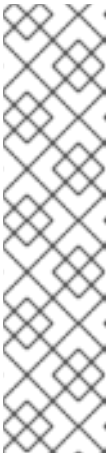
2. Backup the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file:

```
# cp /etc/dirsrv/slapd-instance_name/dse.ldif \
     /etc/dirsrv/slapd-instance_name/dse.ldif.bak.$(date "+%F_%H-%M-
%S")
```

3. Edit the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file:

   a. Set the percentage of free system RAM to use for the database and entry cache. For example, to set 10%:

```
nsslapd-cache-autosize: 10
```



NOTE

If you set the ***nsslapd-cache-autosize*** parameter to **0**, you must additionally set:

- the ***nsslapd-dbcachesize*** in the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry to **0** to enable the auto-sized database cache.
- the ***nsslapd-cachememsize*** in the **cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** entry to **0** to enable the auto-sized entry cache for a database.

   b. Optionally, set the percentage used from the free system RAM for the database cache. For example, to set 40%:

```
nsslapd-cache-autosize-split: 40
```

   Directory Server uses the remaining 60% of free memory for the entry cache.

   c. Save the changes.

4. Start the Directory Server instance:

```
# systemctl start dirsrv@instance_name
```

Example 6.1. The *nsslapd-cache-autosize* and *nsslapd-cache-autosize-split* Parameter

The following settings are the default for both parameters:

```
nsslapd-cache-autosize: 10
nsslapd-cache-autosize-split: 40
```

Using these settings, 10% of the system's free RAM is used (***nsslapd-cache-autosize***). From this memory, 40% are used for the database cache (***nsslapd-cache-autosize-split***) and the remaining 60% for the entry cache.

Depending on the free RAM, this results in the following cache sizes:

GB of Free RAM	Database Cache Size	Entry Cache Size
1 GB	40 MB	62 MB
2 GB	82 MB	122 MB
4 GB	164 MB	245 MB
8 GB	328 MB	492 MB
16 GB	512 MB [a]	1,126 MB
32 GB	512 MB [a]	2,764 MB
64 GB	512 MB [a]	6,042 MB
128 GB	512 MB [a]	12,596 MB

[a] Directory Server applies the 512 MB limit for the ***nsslapd-dbcachesize*** parameter.

6.2. DETERMINING THE REQUIRED CACHE SIZES

The **dbmon.sh** script enables you to monitor cache statistics at runtime and continuously outputs the statistics. To terminate the script, press the **Ctrl**+**C** key combination.

NOTE

The **dbmon.sh** requires you to pass the options as environment variables to the script. For further details see the *Directory Server Configuration, Command, and File Reference*.

Example 6.2. Using the dbmon.sh Script

To display the statistics, enter for example:

```
# BINDDN="cn=Directory Manager" BINDPW=secret SERVID=slapd-instance_name
dbmon.sh
dbcachefree 397310 free% 2.2 roevicts 9282348 hit% 50 pagein 2934772
pageout 219075
      dbname            count          free  free%    size
      userroot:ent      50000          2400    0.8  8972.7
      userroot:dn      100000       4294735   69.8   130.0
```

To list the total number of DNs in the **userroot** database, enter:

```
# dbscan -f /var/lib/dirsrv/slapd-instance_name/db/userRoot/id2entry.db
-t 200 | \
     grep -c rdn:
```

If your caches are sufficiently sized, the number returned by the previous command matches the value in the **count** column of the **dbmon.sh** script's output. Additionally, if all of the entries and DNs fit within their respective caches, the **userroot:ent** count value matches the **userroot:dn** count value.

The output of the **dbmon.sh** script example shows:

- Only 2.2% free database cache is left:

  ```
  dbcachefree 397310 free% 2.2 roevicts 9282348 hit% 50 pagein 2934772
  pageout 219075
  ```

  However, to operate efficiently, at least 15% free database cache is required. To determine the optimal size of the database cache, calculate the sizes of all ***.db** and ***.db4** files in the **/var/lib/dirsrv/slapd-*instance_name*/db/** directory including subdirectories and the changelog database, and add 12% for overhead.

  To set the database cache, see Section 6.5, “Setting the Database Cache Size”.

- The DN cache of the **userroot** database is well-chosen:

  ```
  dbname            count          free  free%    size
  userroot:dn      100000       4294735   69.8   130.0
  ```

  The DN cache of the database contains 100000 records. 69,8% of the cache is free. Based on the **count** value and the bytes used, each DN in memory requires 130 bytes on average.

  To set the DN cache, see Section 6.5, “Setting the Database Cache Size”.

- The statistics on the entry cache of the **userroot** database indicates that the entry cache value should be increased for better performance:

  ```
  dbname            count          free  free%    size
  userroot:ent      50000          2400    0.8  8972.7
  ```

  The entry cache contains in this database 50000 records and only 2 Kilobytes of free space are left. To enable Directory Server to cache all 100000 DNs, reported by the **dbscan** utility's

output, the cache must be increased to minimum of 856 Megabytes (100000 DNs * 8972,7 bytes average entry size). However, it is recommended to round the minimum required size to the next highest Gigabyte and double the result. In this example, the entry cache should be set to 2 Gigabytes.

To set the entry cache, see Section 6.3, “Setting the Entry Cache Size”.

6.3. SETTING THE ENTRY CACHE SIZE

The entry cache is used to store directory entries that are used during search and read operations. Setting the entry cache to a size that enables the Directory Server to store all records has the highest performance impact on search operations.

If entry caching is not configured, Directory Server reads the entry from the **id2entry.db** database file and converts the DNs from the on-disk format to the in-memory format. Entries that are stored in the cache enable the server to skip the disk I/O and conversion steps.

NOTE

Instead of manually setting the entry cache size Red Hat recommends the auto-sizing feature for optimized settings based on the hardware resources. For details, see Section 6.1.1, “Manually Re-enabling the Database and Entry Cache Auto-sizing”.

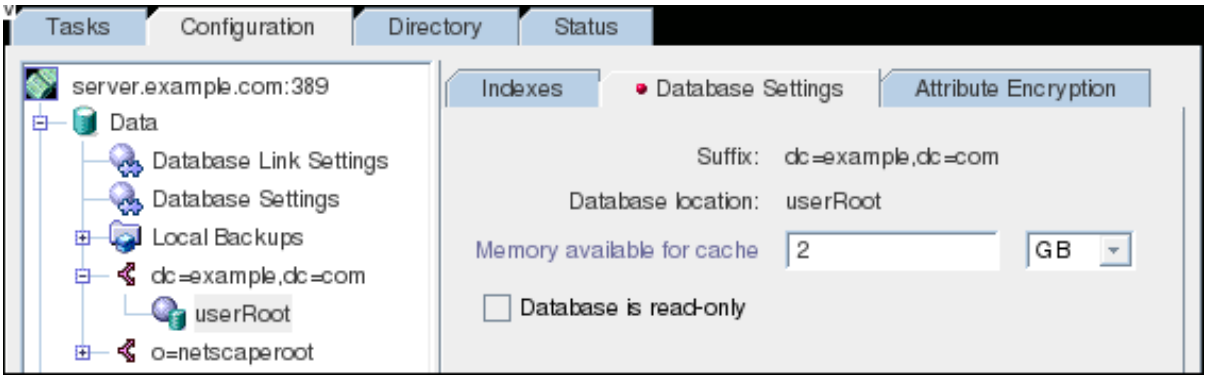
To manually set the entry cache size, you can use:

- the Directory Server Console (see the section called “Directory Server Console: Setting the Entry Cache Size”)
- the Command Line (see the section called “Command Line: Setting the Entry Cache Size”)

Directory Server Console: Setting the Entry Cache Size

For example, to set the entry cache for the **cn=userRoot** database to 2 GB:

1. Start the Directory Server Console.
2. Select the **Configuration** tab and, in the navigation tree, expand the **Data** icon.
3. Expand the suffix associated with the database, such as **dc=example,dc=com**, and then select the database.
4. In the **Database Settings** tab, fill the **Memory available for cache** field and select the unit. For example:



5. Click **Save**.

6. Restart the Directory Server instance:

```
# systemctl restart dirsrv.target
```

Command Line: Setting the Entry Cache Size
For example, to set the entry cache for the LDBM database to 2 GB:

1. Set the value in the Directory Server configuration:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-cachememsize
nsslapd-cachememsize: 2147483648
```

2. Restart the Directory Server instance:

```
# systemctl restart dirsrv.target
```

6.4. SETTING THE DN CACHE

The **entryrdn** index is used to associate DNs and RDNs with entries. It enables the server to efficiently perform subtree **rename**, entry **move**, and **moddn** operations. The DN cache is used to cache the in-memory representation of the **entryrdn** index to avoid expensive file I/O and transformation operations. For best performance, especially with but not limited to entry **rename** and **move** operations, set the DN cache to a size that enables Directory Server to cache all DNs in the database.

If a DN is not stored in the cache, Directory Server reads the DN from the **entryrdn.db** index database file and converts the DNs from the on-disk format to the in-memory format. DNs that are stored in the cache enable the server to skip the disk I/O and conversion steps.

To set the size of the DN cache for the LDBM database to 20 MB:

1. Update the value in the Directory Server configuration:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-dncachememsize
nsslapd-dncachememsize: 20971520
```

2. Restart the Directory Server instance:

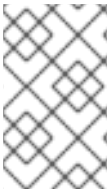
```
# systemctl restart dirsrv.target
```

6.5. SETTING THE DATABASE CACHE SIZE

The database cache contains the Berkeley database index files for the database, meaning all of the ***.db** and other files used for attribute indexing by the database. This value is passed to the Berkeley DB API function **set_cachesize()**.

This cache size has less of an impact on Directory Server performance than the entry cache size, but if there is available RAM after the entry cache size is set, increase the amount of memory allocated to the database cache.

The operating system also has a file system cache which may compete with the database cache for RAM usage. Refer to the operating system documentation to find information on file system cache settings and monitoring the file system cache.



NOTE

Instead of manually setting the entry cache size Red Hat recommends the auto-sizing feature for optimized settings based on the hardware resources. For details, see Section 6.1.1, “Manually Re-enabling the Database and Entry Cache Auto-sizing”.

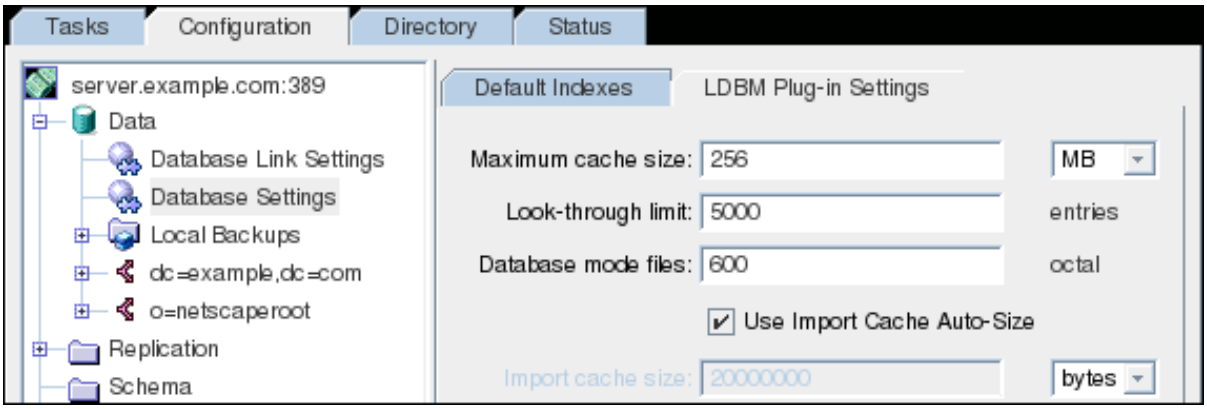
To manually set the database cache size, you can use:

- the Directory Server Console (see the section called “Directory Server Console: Setting the Database Cache Size”)
- the Command Line (see the section called “Command Line: Setting the Database Cache Size”)

Directory Server Console: Setting the Database Cache Size

For example, to set the database cache to 256 megabytes:

1. Start the Directory Server Console.
2. Select the **Configuration** tab and, in the navigation tree, expand the **Data** icon.
3. Select the **Database Settings** entry.
4. In the **LMDB Plug-in Settings** tab, fill the **Maximum cache size** field and select the unit.



5. Click **Save**.

   The Directory Server Console returns an **LDAP_UNWILLING_TO_PERFORM** error message when you set:

   - a value that is not a number.

- a value that is too big for a 32-bit signed integer (2147483647) on a 32-bit system.
- a value that is too big for a 64-bit signed integer (9223372036854775807) on a 64-bit system.

6. Restart the Directory Server instance:

```
# systemctl restart dirsrv.target
```

Command Line: Setting the Database Cache Size
For example, to set the database cache to 256 megabytes:

1. Update the value in the Directory Server configuration:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-dbcachesize
nsslapd-dbcachesize: 268435456
```

2. Restart the Directory Server instance:

```
# systemctl restart dirsrv.target
```

6.5.1. Storing the Database Cache on a RAM Disk

If your system running the Directory Server instance has enough free RAM, you can optionally store the database cache on a RAM disk for further performance improvements:

1. Create a directory for the database cache and metadata on the RAM disk:

```
# mkdir -p /dev/shm/slapd-instance_name/
```

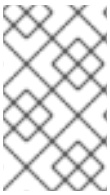
2. Set the path to the directory on the RAM disk in the ***nsslapd-db-home-directory*** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-db-home-directory
nsslapd-db-home-directory: /dev/shm/slapd-instance_name/
```

3. Restart the Directory Server instance:

```
# systemctl restart dirsrv.target
```



NOTE

When the database cache is stored on a RAM disk, Directory Server needs to recreate it after each reboot. As a consequence, the service start and initial operations are slower until the cache is recreated.

CHAPTER 7. SETTING THE NUMBER OF DIRECTORY SERVER THREADS

The number of threads Directory Server uses to handle simultaneous connections affects the performance of the server. For example, if all threads are busy handling time-consuming tasks (such as **add** operations), new incoming connections are queued until a free thread can process the request.

If the server provides a low number of CPU threads, configuring a higher number of threads can increase the performance. However, on a server with many CPU threads, setting a too high value does not further increase the performance.

In instances created using Directory Server 10.1.1 or later, the number of threads Directory Server creates is calculated automatically by default. This number is based on the hardware resources of the server when the instance starts.

If you upgraded from a version earier than 10.1.1 or if you manually set the number of threads, re-enable auto-sizing for optimized performance. For details, see Section 7.1, “Enabling Automatic Thread Tuning”.

NOTE

Red Hat recommends to use the auto-tuning settings. Do not set the number of threads manually.

7.1. ENABLING AUTOMATIC THREAD TUNING

Directory Server can automatically set the number of threads based on the available hardware threads:

1. Enable auto-setting the number of threads:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-threadnumber
nsslapd-threadnumber: -1
```

With this setting, Directory Server will use the following optimized number of threads:

Number of CPU Threads	Number of Directory Server Threads
1	16
2	16
4	24
8	32
16	48
32	64

Number of CPU Threads	Number of Directory Server Threads
64	96
128	192
256	384
512	512 [a]
1024	512 [a]
2048	512 [a]
[a] The recommended maximum number of threads is applied.	

NOTE

If you enabled the automatic setting, the ***nsslapd-threadnumber*** parameter shows the calculated number of threads while Directory Server is running.

2. Restart the Directory Server instance:

```
# systemctl start dirsrv@instance_name
```

7.2. MANUALLY SETTING THE NUMBER OF THREAD

In certain situations, it can be necessary to manually set a fixed number of Directory Server threads instead of using the automatic thread tuning. For example, to set **64** threads:

1. Set the number of threads:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-threadnumber
nsslapd-threadnumber: 64
```

2. Restart the Directory Server instance:

```
# systemctl start dirsrv@instance_name
```

NOTE

If the number of hardware threads changes, for example, because you increased the CPU cores of the virtual machine that runs the Directory Server instance, you must manually update this setting. To use the optimized and automatic setting, see Section 7.1, “Enabling Automatic Thread Tuning”.

CHAPTER 8. TUNING THE REPLICATION PERFORMANCE

8.1. IMPROVING THE MULTI-MASTER REPLICATION EFFICIENCY

The replication latency in a multi-master replication environment, especially if the servers are connected using a wide area network (WAN), can be high in case of multiple masters are receiving updates at the same time. This happens when one master exclusively accesses a replica without releasing it for a long time. In such situations, other masters cannot send updates to this consumer, which increases the replication latency

To release a replica after a fixed amount of time, set the ***nsds5ReplicaReleaseTimeout*** parameter on replication masters and hubs. For example, to set a **60** seconds timeout, enter:

```
[root@server ~]# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
replace: nsds5ReplicaReleaseTimeout
nsds5ReplicaReleaseTimeout: 60
```

The **60** second default value is ideal for most environments. A value set too high or too low can have a negative impact on the replication performance. If the value is set too low, replication servers are constantly reacquiring one another and servers are not able to send many updates. In a high-traffic replication environment, a longer timeout can improve situations where one master exclusively accesses a replica. However, in most cases, a value higher than **120** seconds slows down replication.

CHAPTER 9. TUNING DATABASE LINK PERFORMANCE

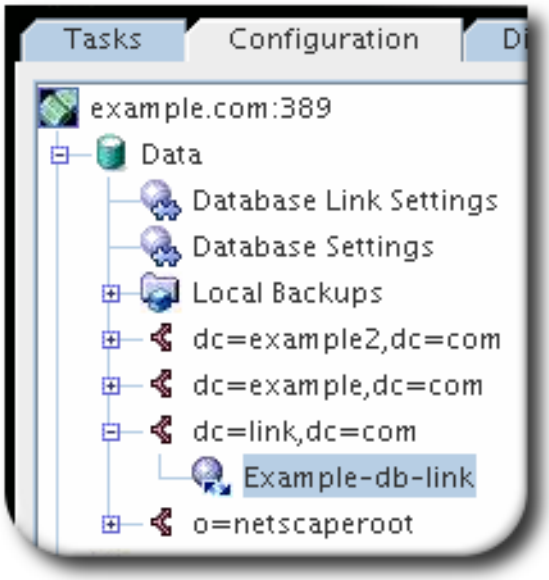
Database link performance can be improved through changes to the Directory Server's connection and thread management.

9.1. MANAGING CONNECTIONS TO THE REMOTE SERVER

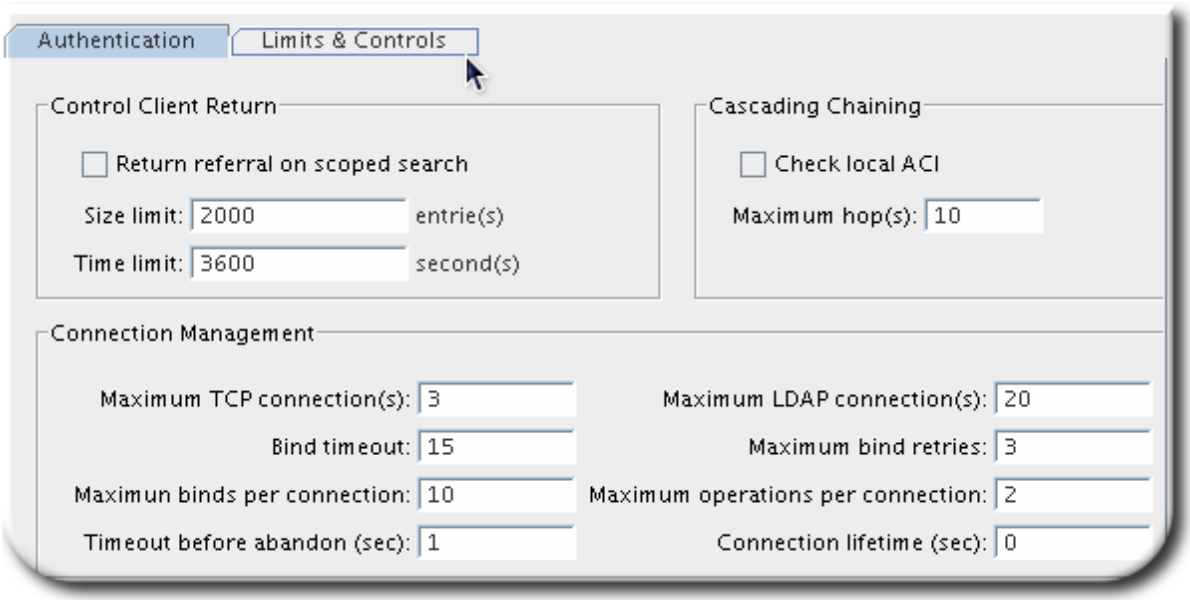
Each database link maintains a pool of connections to a remote server. The connections to optimize resources can be configured for the directory.

9.1.1. Managing Connections to the Remote Server Using the Console

1. Select the **`Configuration`** tab, expand the **`Data`** folder in the left pane, and select the suffix, then the database link to change.



2. Click the **`Limits and Controls`** tab in the right navigation pane.



3. In the **`Connection Management`** section, make changes to any of the following fields:

- *Maximum TCP connection(s).* The maximum number of TCP connections that the database link establishes with the remote server. The default value is **3** connections.
- *Bind timeout.* Amount of time, in seconds, before the database link's bind attempt times out. The default value is **15** seconds.
- *Maximum binds per connection.* Maximum number of outstanding bind operations per TCP connection. The default value is **10** outstanding bind operations per connection.
- *Time out before abandon (sec).* Number of seconds before the server checks to see if a timed-out connection should be abandoned. The default value is **1** second.
- *Maximum LDAP connection(s).* Maximum number of LDAP connections that the database link establishes with the remote server. The default value is **10** connections.
- *Maximum bind retries.* Number of times a database link attempts to bind to the remote server. A value of **0** indicates that the database link will try to bind only once. The default value is **3** attempts.
- *Maximum operations per connection.* Maximum number of outstanding operations per LDAP connection. The default value is **2** operations per connection.
- *Connection lifetime (sec).* How long a connection made between the database link and remote server remains open. Connections between the database link and the remote server can be kept open for an unspecified time or closed after a specific period of time. It is faster to keep the connections open, but it uses more resources. For slow connections, it may be desirable to limit the connection time. A value of **0** indicates there is no limit. By default, the value is set to **0**.

9.1.2. Managing Connections to the Remote Server from the Command Line

Use **ldapmodify** to add connection attributes to the database link entry.

The default connection management attributes are stored in the following entry:

```
cn=default instance config,cn=chaining database,cn=plugins,cn=config
```

The connection management attributes for a specific database link are stored in the following entry:

```
cn=database_link,cn=chaining database,cn=plugins,cn=config
```

The connection management attributes specified in this entry take precedence over the attributes specified in the **cn=default instance config** entry.

Table 9.1. Database Link Connection Management Attributes

Attribute Name	Description
nsOperationConnectionsLimit	Maximum number of LDAP connections that the database link establishes with the remote server. The default value is **20** connections per database link instance.

Attribute Name	Description
nsBindConnectionsLimit	Maximum number of TCP connections that the database link establishes with the remote server. The default value is **3** connections.
nsConcurrentOperationsLimit	Maximum number of outstanding operations per LDAP connection. The default value is **2** operations per connection.
nsConcurrentBindLimit	Maximum number of outstanding bind operations per TCP connection. The default value is **10** outstanding bind operations.
nsBindRetryLimit	Number of times a database link attempts to bind to the remote server. A value of zero (**0**) indicates that the database link will try to bind only once. The default value is **3** attempts.
nsConnectionLife	Connection lifetime, in seconds. Connections between the database link and the remote server can be kept open for an unspecified time or closed after a specific period of time. It is faster to keep the connections open, but it uses more resources. For example, it may be wise to limit the connection time for a slow connection. A value of **0** indicates there is no limit. By default, the value is set to **0**. When the value is **0** and there is a list of failover servers in the ***nsFarmServerURL*** attribute, the first server is never contacted after failover to the alternate server. The default value is **0** seconds.
nsBindTimeout	Amount of time, in seconds, before the bind attempt times out. The default value is **15** seconds.
nsAbandonedSearchCheckInterval	Number of seconds that pass before the server checks for abandoned operations. The default value is **1** second.

9.2. DETECTING ERRORS DURING NORMAL PROCESSING

Protect server performance by detecting errors during the normal chaining operation between the database link and the remote server. The database link has two attributes — ***nsMaxResponseDelay*** and ***nsMaxTestResponseDelay*** — which work together to determine if the remote server is no longer responding.

The first attribute, ***nsMaxResponseDelay***, sets a maximum duration for an LDAP operation to complete. If the operation takes more than the amount of time specified in this attribute, the database link's server suspects that the remote server is no longer online.

Once the ***nsMaxResponseDelay*** period has been met, the database link pings the remote server. During the ping, the database link issues another LDAP request, a simple search request for an object that does not exist in the remote server. The duration of the ping is set using the ***nsMaxTestResponseDelay***.

If the remote server does not respond before the ***nsMaxTestResponseDelay*** period has passed, then an error is returned, and the connection is flagged as down. All connections between the database link and remote server will be blocked for 30 seconds, protecting the server from a performance degradation.

After 30 seconds, operation requests made by the database link to the remote server continue as normal.

Both attributes are stored in the **cn=config,cn=chaining database,cn=plugins,cn=config** entry. The following table describes the attributes in more detail:

Table 9.2. Database Link Processing Error Detection Parameters

Attribute Name	Description
nsMaxResponseDelay	Maximum amount of time it can take a remote server to respond to an LDAP operation request made by a database link before an error is suspected. This period is given in seconds. The default delay period is **60** seconds. Once this delay period has been met, the database link tests the connection with the remote server.
nsMaxTestResponseDelay	Duration of the test issued by the database link to check whether the remote server is responding. If a response from the remote server is not returned before this period has passed, the database link assumes the remote server is down, and the connection is not used for subsequent operations. This period is given in seconds. The default test response delay period is **15** seconds.

CHAPTER 10. IMPROVING IMPORT PERFORMANCE

Very large entry sizes or a large number of entries can negatively impact server performance during import operations. Both Directory Server settings and operationg system settings can be tuned to improve performance for imports.

10.1. IMPORTING ENTRIES WITH LARGE ATTRIBUTES

The ***nsslapd-cachememsize*** attribute defines the size allowed for the entry cache.

The import buffer is automatically set to 80% of the cache memory size setting. If the memory cache is 1GB, for example, then the import buffer is 800MB.

When importing a very large database or entries with large attributes (often with values like binary data like certificate chains, CRLs, or images), then set the ***nsslapd-cachememsize*** attribute high enough so that the import buffer has enough memory to process the entries.

10.2. IMPORTING LARGE NUMBERS OF ENTRIES

When there are a large number of entries to be imported, the operating system itself may hit performance limits on what it allows the Directory Server to do. This is particularly true on x86 systems. This can cause import operations to fail because of resource constraints.

If necessary, set the system **ulimit** value to the maximum number of allows processes for the system user.

For example:

```
# ulimit -u 4096
```

Then run the import operation.

APPENDIX A. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Directory Server.

Revision 10.3-1 **Wed Oct 10 2018** **Marc Muehlfeld**
Red Hat Directory Server 10.3 release of the guide.

Revision 10.2-1 **Tue Apr 10 2018** **Marc Muehlfeld**
For version 10.2: Removed the *Optimizing System Performance* section.

Revision 10.1-5 **Tue Dec 05 2017** **Marc Muehlfeld**
Added the *Moving the Database Directory to a Separate Disk or Partition* section.

Revision 10.1-4 **Tue Aug 01 2017** **Marc Muehlfeld**
For version 10.1.1: Described database cache, entry cache, and number of threads auto-tuning.

Revision 10.1-3 **Fri Feb 24 2017** **Marc Muehlfeld**
Added dbmon.sh description. Added "Tuning the Replication Performance" chapter.

Revision 10.1-2 **Wed Dec 14 2016** **Marc Muehlfeld**
Updated Resetting the Host Machine's File Descriptors section.

Revision 10.1-0 **Wed Nov 02 2016** **Marc Muehlfeld**
Red Hat Directory Server 10.1 release of the guide.

Revision 10.0-1 **Wed Jun 22 2016** **Petr Bokoč**
Add information to avoid using owner nobody:nobody.

Revision 10.0-0 **Tue Jun 09 2015** **Tomáš Čapek**
Red Hat Directory Server 10 release of the guide.