Red Hat Directory Server 12

Tuning the performance of Red Hat Directory Server

Improving the server and database performance

Last Updated: 2024-01-25

Red Hat Directory Server 12 Tuning the performance of Red Hat Directory Server

Improving the server and database performance

Legal Notice

Abstract

To improve Directory Server performance, you can tune the number of locks, resource limits, and transaction log. You can also monitor the local disk and shut down Directory Server if the disk space available on a system becomes insufficient.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For submitting feedback through Jira (account required):
  1. Log in to the Jira website.
  2. Click **Create** in the top navigation bar
  3. Enter a descriptive title in the **Summary** field.
  4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
  5. Click **Create** at the bottom of the dialogue.
- For submitting feedback through Bugzilla (account required):
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. MONITORING THE DATABASE ACTIVITY

Administrators should monitor the database activity to ensure that tuning settings, such as caches, are properly configured.

1.1. MONITORING THE DATABASE ACTIVITY USING THE COMMAND LINE

To display the monitoring activity using the command line, display the dynamically-updated read-only attributes stored in the **cn=monitor,cn=*database_name*,cn=ldbm database,cn=plugins,cn=config**.

Procedure

- To display the current activity of a database, enter:

  ```
  # dsconf -D "cn=Directory Manager" ldap://server.example.com monitor backend userRoot
  ```

  This command displays the activity of the **userRoot** database.

Additional resources

- Database monitoring attributes

1.2. MONITORING THE DATABASE ACTIVITY USING THE WEB CONSOLE

In the web console, Directory Server displays the values of the dynamically-updated read-only monitoring attributes from the **cn=monitor,cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** in the `Monitoring´ tab.

Procedure

1. Navigate to **Monitoring → Database →** ***database name***.
2. Display the cache values on the **Entry Cache** and **DN Cache** tabs.

Additional resources

- Database monitoring attributes

1.3. DATABASE MONITORING ATTRIBUTES

Table 1.1. Inheritance settings

Attribute	Description
readonly	Indicates whether the database is in read-only mode (**1**) or in read-write mode (**0**).

Attribute	Description
entrycachehits	The total number of successful entry cache lookups. The value is the total number of times the server could retrieve an entry from the entry cache without reloading it from the database.
entrycachetries	The total number of entry cache lookups since you started the instance. The value is the total number, since the instance has been started, Directory Server tried to retrieve entry from the entry cache.
entrycachehitratio	The number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since you last started the instance. The closer the entry cache hit ratio is to 100%, the better. Whenever an operation attempts to find an entry that is not present in the entry cache, the server needs to access the database to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance decreases. To improve this ratio, increase the size of the entry cache of the database. 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.
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.
maxentrycachesize	The maximum size, in bytes, of directory entries that Directory Server can maintain 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.
currententrycachecount	The current number of entries stored in the entry cache of a given backend.
maxentrycachecount	The maximum number of entries stored in the entry cache of a database. To tune this value, increase the value of the **nsslapd-cachesize** attribute in the **cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** entry.

Attribute	Description
dncachehits	The number of times the server could process a request by obtaining a normalized distinguished name (DN) from the DN cache rather than normalizing it again.
dncachetries	The total number of DN cache accesses since you started the instance.
dncachehitratio	The ratio of cache tries to successful DN cache hits. The closer this value is to 100%, the better.
currentdncachesize	The total size, in bytes, of DN 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.
maxdncachesize	The maximum size, in bytes, of DNs that Directory Server can maintain 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.
currentdncachecount	The number of DNs currently present in the DN cache.
maxdncachecount	The maximum number of DNs allowed in the DN cache.

CHAPTER 2. IMPROVING THE PERFORMANCE OF VIEWS

The performance of view-based hierarchies depends on the construction of the hierarchy itself and the number of entries in the directory tree (DIT).

In general, there may be a marginal change in performance (within a few percentage points of equivalent searches on a standard DIT) if you use virtual DIT views. If you do not invoke a view in a search, then there is no performance impact. Test the virtual DIT views against expected search patterns and loads before deployment.

Red Hat recommends indexing the attributes used in view filters if you intend to use the views as general-purpose navigation tools in the organization.

Further, you can configure a virtual list view (VLV) index to be used in evaluation of sub-filters in views.

There is no need to tune any other part of the directory specifically for views.

2.1. CREATING INDEXES TO IMPROVE THE PERFORMANCE OF VIEWS USING THE COMMAND LINE

Views are derived from search results based on a given filter. Part of the filter are the attributes given explicitly in the **nsViewFilter**; the rest of the filter is based on the entry hierarchy, looking for the **entryid** and **parentid** operational attributes of the actual entries included in the view.

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

If any of the searched attributes – **entryid**, **parentid**, or the attributes in the **nsViewFilter** – are not indexed, then the search becomes partially unindexed and Directory Server searches the entire directory tree for matching entries.

To improve views performance, create the indexes as follows:

- Create *equality index* (**eq**) for **entryid**. The **parentid** attribute is indexed in the system index by default.
- If a filter in **nsViewFilter** tests presence (**attribute=***), then create *presence index* (**pres**) for the attribute being tested. You should use this index type only with attributes that appear in a minority of directory entries.
- If a filter in **nsViewFilter** tests equality (**attribute=value**), create *equality index* (**eq**) for the attribute being tested.
- If a filter in **nsViewFilter** tests a substring (**attribute=value***), create *substring index* (**sub**) for the attribute being tested.
- If a filter in **nsViewFilter** tests approximation (**attribute~=value**), create *approximate index* (**approximate**) for the attribute being tested.

For example, when you use the following view filter:

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

you should index **objectClass** with the *equality index*, which is done by default, and **roomNumber** with the *substring index*.

Prerequisites

- You are aware of the attributes that you use in a view filter.

Procedure

1. Optional: List the back ends to determine the database to index:

```
# dsconf -D "cn=Directory Manager" instance_name backend suffix list
dc=example,dc=com (userroot)
```

   Note the selected database name (in parentheses).

2. Create index configuration with the **dsconfig** utility for the selected back-end database. Specify the attribute name, index type, and, optionally, matching rules to set collation order (OID), especially in case of an internationalized instance.

```
# dsconf -D "cn=Directory Manager" instance_name backend index add --attr
roomNumber --index-type sub userroot
```

   Repeat this step for each attribute used in the view filter.

3. Reindex the database to apply the new indexes:

```
# dsconf -D "cn=Directory Manager" instance_name backend index reindex userroot
```

Verification

1. Perform a search that is based on the standard directory tree with the same filter that you use in the view:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com (&(objectClass=inetOrgPerson)(roomNumber=*66))
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com "(&(objectClass=inetOrgPerson)(roomNumber=*66))"
```

2. View the *access* log in **/var/log/dirsrv/slapd-*instance_name*/access**.
   The **RESULT** of your search should not contain **note=U** or **Partially Unindexed Filter** in the details.

Additional resources

- Managing indexes

2.2. CREATING INDEXES TO IMPROVE THE PERFORMANCE OF VIEWS USING THE WEB CONSOLE

Views are derived from search results based on a given filter. Part of the filter are the attributes given explicitly in the **nsViewFilter**; the rest of the filter is based on the entry hierarchy, looking for the **entryid** and **parentid** operational attributes of the actual entries included in the view.

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

If any of the searched attributes – **entryid**, **parentid**, or the attributes in the **nsViewFilter** – are not indexed, then the search becomes partially unindexed and Directory Server searches the entire directory tree for matching entries.

To improve views performance, create the indexes as follows:

- Create *equality index* (**eq**) for **entryid**. The **parentid** attribute is indexed in the system index by default.
- If a filter in **nsViewFilter** tests presence (**attribute=***), then create *presence index* (**pres**) for the attribute being tested. You should use this index type only with attributes that appear in a minority of directory entries.
- If a filter in **nsViewFilter** tests equality (**attribute=value**), create *equality index* (**eq**) for the attribute being tested.
- If a filter in **nsViewFilter** tests a substring (**attribute=value***), create *substring index* (**sub**) for the attribute being tested.
- If a filter in **nsViewFilter** tests approximation (**attribute~=value**), create *approximate index* (**approximate**) for the attribute being tested.

For example, when you use the following view filter:

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

you should index **objectClass** with the *equality index*, which is done by default, and **roomNumber** with the *substring index*.

Prerequisites

- You are logged in to the instance in the web console.
- You are aware of the attributes that you use in a view filter.

Procedure

1. Under **Database**, select a suffix from the configuration tree for which you want to create an index.
2. Navigate to **Indexes** and **Database Indexes**.
3. Click the **Add Index** button.
4. Type the name of the attribute and select the attribute.
5. Select the **Index Types** that should be created for this attribute.
6. Optionally, add **Matching Rules** to specify collation order (OID), especially in case of an internationalized instance.
7. Select **Index attribute after creation** to rebuild the index afterwards.
8. Click **Create Index**.
9. Repeat the steps for each attribute to be indexed.

Verification

- **Filter Indexes** by typing the name of the added attribute.
- The newly indexed attribute should appear in the results.

Additional resources

- Managing indexes

CHAPTER 3. SETTING AN INDEX SCAN LIMIT TO IMPROVE THE PERFORMANCE WHEN LOADING LONG LISTS OF IDS

In large directories, the search results list can be huge. For example, a directory with one million entries with **inetorgperson** attributes would return all these entries in a search with a filter, such as **(objectclass=inetorgperson)**.

Loading a long ID list from the database significantly reduces search performance. An ID list scan limit sets a limit on the number of IDs Directory Server reads before a key is considered to match the entire primary index. This means that Directory Server treats the search 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, the entire directory, to process results rather than searching through an index that is nearly the size of a directory, plus the directory itself.

You can set an index scan limit globally or for specific databases.

3.1. SETTING A GLOBAL INDEX SCAN LIMIT USING THE COMMAND LINE

By default, the ID list scan limit in Directory Server is **4000**. In most scenarios, this value provides good performance for a common range of database sizes and access patterns, and you do not need to change the default value. If the database index is slightly larger than 4000 entries, but still significantly smaller than the overall directory, raising the ID list scan limit improves searches.

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.

Procedure

1. Update the ID list scan limit:

   ```
   # dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --idlistscanlimit=8000
   ```

   This command sets the limit to **8000** entries.

2. Restart the instance:

   ```
   # dsctl instance_name restart
   ```

3.2. SETTING A GLOBAL INDEX SCAN LIMIT USING THE WEB CONSOLE

By default, the ID list scan limit in Directory Server is **4000**. In most scenarios, this value provides good performance for a common range of database sizes and access patterns, and you do not need to change the default value. If the database index is slightly larger than 4000 entries, but still significantly smaller than the overall directory, raising the ID list scan limit improves searches.

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.

Procedure

1. Navigate to **Database → Global Database Configuration**.
2. Update the **ID List Scan Limit** field.
3. Click **Save Config**.
4. Click **Actions** in the top right corner, and select **Restart Instance**.

3.3. SETTING AN INDEX SCAN LIMIT TO A DATABASE USING THE COMMAND LINE

In some cases, it is useful to define a limit for certain indexes, or to not use an ID list at all. You can configure individual settings for ID list scan limits for different types of search filters.

For example, in a large database with 10 million entries that contain the object class **inetOrgPerson**, the **(&(objectClass=inetOrgPerson)(uid=user))** filter 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 case, it is useful to define a limit for certain indexes, or to not use an ID list at all.

This procedure demonstrates how to configure Directory Server to note create an ID list for **objectClass=inetOrgPerson** conditions in **AND** clauses.

Procedure

- Set the **nsIndexIDListScanLimit** parameter:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://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
```

With these settings, Directory Server does not create any ID list for **objectClass=inetOrgPerson** conditions in **AND** clauses. In all other situations, Directory Server applies the global ID list scan limit value.

The **nsIndexIDListScanLimit** parameter uses the following syntax:

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

- **limit**: Sets the maximum size of the ID list. Valid values are:
  - **-1**: Unlimited
  - **0**: Do not use the index
  - **1** to the maximum of the 32-bit integer (**2147483647**): Maximum number of IDs
- **type**: Optional: Sets flags that alter the scan limit's behavior. 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: A 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.
  Use the values only with one type at a time. The values must correspond to the index type and 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 the **eq** type, you cannot use **type=eq values=*abc***.

  If the value contains spaces, commas, NULL, or other values which require escaping, use the LDAP filter escape syntax: A backslash (\) followed by the 2 hex digit code of 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
  ```

CHAPTER 4. TUNING THE NUMBER OF LOCKS

Lock mechanisms in Directory Server control how many copies of Directory Server processes can run at the same time. For example, during an import job, Directory Server sets a lock in the **/run/lock/dirsrv/slapd-instance_name/imports/** directory to prevent the **ns-slapd** Directory Server process, another import, or export operations from running.

If the server runs out of available locks, Directory Server logs the following error in the **/var/log/dirsrv/slapd-instance_name/errors** file:

```
libdb: Lock table is out of available locks
```

However, the Directory Server default settings try to prevent the server from running out of locks to avoid data corruption. For details, see Avoiding data corruption by monitoring free database locks

4.1. AVOIDING DATA CORRUPTION BY MONITORING FREE DATABASE LOCKS

Running out of database locks can lead to data corruption. To avoid this, Directory Server, by default, monitors the remaining number of free database locks every 500 milliseconds and, if the number of active database locks is equal or higher than the 90%, Directory Server aborts all searches.

This procedure changes the interval to **600** milliseconds and the threshold to **85** percent.

NOTE

If you set a too high interval, the server can run out of locks before the next monitoring check happens. Setting a too short interval can slow down the server.

Procedure

1. Set the interval and threshold:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --locks-monitoring-enabled on --locks-monitoring-pause 600 --locks-monitoring-threshold 85
```

2. Restart the instance:

```
# dsctl instance_name restart
```

Verification

- Display the locks monitoring settings:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com backend config get | grep "nsslapd-db-locks-monitoring"
nsslapd-db-locks-monitoring-enabled: on
nsslapd-db-locks-monitoring-threshold: 85
nsslapd-db-locks-monitoring-pause: 600
```

4.2. MANUALLY MONITORING THE NUMBER OF LOCKS

Directory Server tracks the current number of locks in the **nsslapd-db-current-locks** and **nsslapd-db-max-locks** attributes in **cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config**.

Procedure

- To display the number of locks, enter:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -s sub -b
"cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config" nsslapd-db-
current-locks nsslapd-db-max-locks
...
nsslapd-db-current-locks: 37
nsslapd-db-max-locks: 39
```

4.3. SETTING THE NUMBER OF LOCKS USING THE COMMAND LINE

Use the **dsconf backend config set** command to update the number of locks Directory Server can use.

Procedure

1. Set the number of locks:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --
locks=20000
```

   This command sets the number of locks to **20000**.

2. Restart the instance:

```
# dsctl instance_name restart
```

Verification

- Display the value of the **nsslapd-db-locks** parameter:

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com backend config get |
grep "nsslapd-db-locks:"
nsslapd-db-locks: 20000
```

4.4. SETTING THE NUMBER OF LOCKS USING THE WEB CONSOLE

You can set the number of locks Directory Server uses in the global database configuration in the web console.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Database → Global Database Configuration**.

2. Click **Show Advanced Settings**.
3. Select **Enable Monitoring**, and enter the threshold percentage and pause milliseconds.
4. Click **Save Config**.
5. Click **Actions → Restart Instance**.

Verification

1. Navigate to **Database → Global Database Configuration**.
2. Click **Show Advanced Settings**.
3. Verify if the lock monitoring settings.

CHAPTER 5. 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.

By default, Directory Server uses an auto-tuning setting that calculates the number of threads. This number is based on the hardware resources of the server when the instance starts.

WARNING

Avoid setting the number of threads manually. Use the auto-tuning setting instead.

With enabled automatic thread tuning, Directory Server uses the following optimized number of threads:

CPU threads number	Directory Server threads number
1-16	16
17-512	The Directory Server thread number matches the CPU thread number in the system. For example, if your system has 24 CPU threads, Directory Server uses 24 threads. The maximum number of Directory Server threads is 512.
512 and more	512. Directory Server applies the recommended maximum number of threads.

5.1. ENABLING AUTOMATIC THREAD TUNING USING THE COMMAND LINE

By default, Directory Server automatically sets the number of threads based on the available hardware. However, in certain cases, you can manually enable this auto-tuning feature by using the command line.

Procedure

- To enable the auto-tuning feature, set the **nsslapd-threadnumber** attribute value to **-1** by the command:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-threadnumber="-1"
```

```
Successfully replaced "nsslapd-threadnumber"
```

Verification

- Verify the number of treads that Directory Server now uses by the command:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-threadnumber

nsslapd-threadnumber: 16
```

NOTE

The command retrieves the number of threads that Directory Server calculated based on the correct hardware resources.

Additional resources

- The **nsslapd-threadnumber** attribute description.

5.2. ENABLING AUTOMATIC THREAD TUNING USING THE WEB CONSOLE

By default, Directory Server automatically sets the number of threads based on the available hardware. However, in certain cases, you can manually enable this auto-tuning feature by using the web console.

Prerequisites

- You are logged in to the instance in the web console. For more details, see Logging in to the Directory Server by using the web console.

Procedure

1. Navigate to **Server → Tuning & Limits**.
2. In the **Number Of Worker Threads** field, set the number of threads to **-1**.
3. Click **Save Settings**.

Additional resources

- The **nsslapd-threadnumber** attribute description.

5.3. MANUALLY SETTING THE NUMBER OF THREADS USING THE COMMAND LINE

In certain situations, it is necessary to manually set a fixed number of Directory Server threads. For example, if you do not use the auto-tuning setting and change the number of CPU cores in a virtual machine, adjusting the number of Directory Server threads can improve the performance.

You can also use this procedure to re-enable the auto-tuning setting if you set a specific number of threads earlier.

Procedure

- Set the number of threads Directory Server should use:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-threadnumber="64"

Successfully replaced "nsslapd-threadnumber"
```

  Set the **nsslapd-threadnumber** parameter to **-1** to enable the auto-tuning setting.

5.4. MANUALLY SETTING THE NUMBER OF THREADS USING THE WEB CONSOLE

In certain situations, it is necessary to manually set a fix number of Directory Server threads. For example, if you do not use the auto-tuning setting and change the number of CPU cores in a virtual machine, adjusting the number of Directory Server threads can improve the performance.

Note that you can use the web console to re-enable the auto-tuning setting if you set a specific number of threads earlier.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Server → Tuning & Limits**.
2. In the **Number Of Worker Threads** field, set the number of threads.
3. Click **Save Settings**.

CHAPTER 6. TUNING RESOURCE LIMITS

Directory Server provides several settings to tune the amount of resources an instance uses. You can change them using the command line or the web console.

6.1. UPDATING RESOURCE LIMIT SETTINGS USING THE COMMAND LINE

This section provides a general procedure how to change resource limit settings. Adjust the settings according to your environment.

Procedure

1. Update the performance settings:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
parameter_name=value
```

   You can set the following parameters:

   - **nsslapd-threadnumber**: Sets the number of worker threads.
   - **nsslapd-maxdescriptors**: Sets the maximum number of file descriptors.
   - **nsslapd-timelimit**: Sets the search time limit.
   - **nsslapd-sizelimit**: Sets the search size limit.
   - **nsslapd-pagedsizelimit**: Sets the paged search size limit.
   - **nsslapd-idletimeout**: Sets the idle connection timeout.
   - **nsslapd-ioblocktimeout**: Sets the input/output (I/O) block timeout.
   - **nsslapd-ndn-cache-enabled**: Enables or disables the normalized DN cache.
   - **nsslapd-ndn-cache-max-size**: Sets the normalized DN cache size, if nsslapd-ndn-cache-enabled is enabled.
   - **nsslapd-outbound-ldap-io-timeout**: Sets the outbound I/O timeout.
   - **nsslapd-maxbersize**: Sets the maximum Basic Encoding Rules (BER) size.
   - **nsslapd-maxsasliosize**: Sets the maximum Simple Authentication and Security Layer (SASL) I/O size.
   - **nsslapd-listen-backlog-size**: Sets the maximum number of sockets available to receive incoming connections.
   - **nsslapd-max-filter-nest-level**: Sets the maximum nested filter level.
   - **nsslapd-ignore-virtual-attrs**: Enables or disables virtual attribute lookups.
   - **nsslapd-connection-nocanon**: Enables or disables reverse DNS lookups.
   - **nsslapd-enable-turbo-mode**: Enables or disables the turbo mode feature.

For further details, see the descriptions of the parameters in the Configuration and schema reference

2. Restart the instance:

```
# dsctl instance_name restart
```

6.2. UPDATING RESOURCE LIMIT SETTINGS USING THE WEB CONSOLE

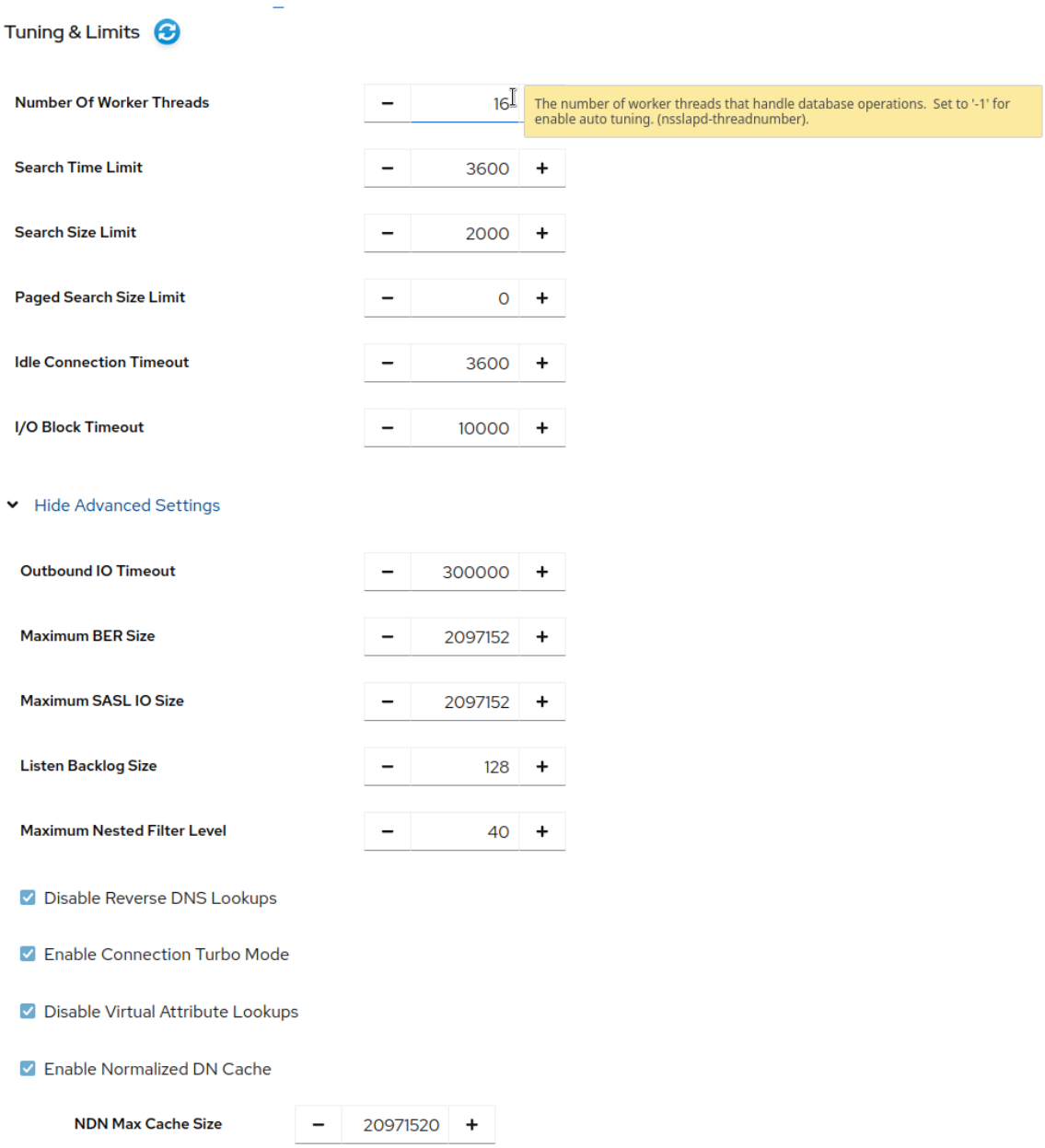
This section provides a general procedure how to change resource limit settings. Adjust the settings according to your environment.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Server → Tuning & Limits**.
2. Update the settings. Optionally, click **Show Advanced Settings** to display all settings.



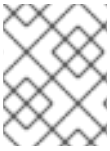
3. Click **Save Settings**.

4. Click **Actions → Restart Instance**.

6.3. DISABLING THE TRANSPARENT HUGE PAGES FEATURE

Transparent Huge Pages (THP) is the memory management feature in Linux, which reduces the overhead of Translation Lookaside Buffer (TLB) checks on machines with large amounts of memory by using larger memory pages. The THP feature is enabled by default on RHEL systems and supports 2 MB memory pages.

The THP feature, however, works best when enabled on large, contiguous allocation patterns and can degrade performance on small, sparse allocation patterns that are typical to the Red Hat Directory Server. The resident memory size of the process might eventually exceed the limit and impact performance or get terminated by the out of memory (OOM) killer.



NOTE

It is recommended to disable THP on RHEL systems with Red Hat Directory Server installed to avoid performance and memory consumption problems.

Procedure

1. Check the current status of transparent huge pages by running the following command:

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
```

2. If the transparent huge pages feature is active, disable it either at boot time or run time:

   - Disable the transparent huge pages at boot time by appending the following to the kernel command line in the **grub.conf** file:

```
transparent_hugepage=never
```

   - Disable transparent huge pages at run time by running the following commands:

```
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
# echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

Additional resources

- The negative effects of Transparent Huge Pages (THP) on RHDS article.
- Configuring Transparent Huge Pages.

CHAPTER 7. LOGGING STATISTICS PER SEARCH OPERATION

During some search operations, especially with filters such as **(cn=user*)**, the time the server spends for receiving the tasks and then sending the result back (**etime**) can be very long.

Expanding the access log with information related to indexes used during search operation helps to diagnose why **etime** value is resource expensive.

Use the **nsslapd-statlog-level** attribute to enable collecting statistics, such as a number of index lookups (database read operations) and overall duration of index lookups for each search operation, with minimal impact on the server.

Prerequisites

- You enabled access logging.

Procedure

1. Enable search operation metrics:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-statlog-level=1
```

2. Restart the instance:

```
# dsctl instance_name restart
```

Verification

1. Perform a search operation:

```
# ldapsearch -D "cn=Directory Manager" -H ldap://server.example.com -b
"dc=example,dc=com" -s sub -x "cn=user*"
```

2. View the access log file and find the search statistics records:

```
# cat /var/log/dirsrv/slapd-instance_name/access
...
[16/Nov/2022:11:34:11.834135997 +0100] conn=1 op=73 SRCH
base="dc=example,dc=com" scope=2 filter="(cn=user)"* attrs=ALL
[16/Nov/2022:11:34:11.835750508 +0100] conn=1 op=73 STAT read index:
attribute=objectclass key(eq)=referral --> count 0
[16/Nov/2022:11:34:11.836648697 +0100] conn=1 op=73 STAT read index: attribute=cn
key(sub)=er_ --> count 25
[16/Nov/2022:11:34:11.837538489 +0100] conn=1 op=73 STAT read index: attribute=cn
key(sub)=ser --> count 25
[16/Nov/2022:11:34:11.838814948 +0100] conn=1 op=73 STAT read index: attribute=cn
key(sub)=use --> count 25
[16/Nov/2022:11:34:11.841241531 +0100] conn=1 op=73 STAT read index: attribute=cn
key(sub)=^us --> count 25
[16/Nov/2022:11:34:11.842230318 +0100] conn=1 op=73 STAT read index: duration
0.000010276
```

```
[16/Nov/2022:11:34:11.843185322 +0100] conn=1 op=73 RESULT err=0 tag=101
nentries=24 wtime=0.000078414 optime=0.001614101 etime=0.001690742
...
```

Additional resources

- TBA the link to nsslapd-statlog-level description

CHAPTER 8. MONITORING THE LOCAL DISK TO SHUT DOWN DIRECTORY SERVER ON LOW DISK SPACE

When the disk space available on a system becomes too small, the Directory Server process terminates. As a consequence, there is a risk of corrupting the database or losing data. To prevent this problem, you can configure Directory Server to monitor the free disk space on the file systems that contain the configuration, transaction log, and database directories. If the free space reaches the configured threshold, Directory Server shuts down the instance.

8.1. BEHAVIOR OF DIRECTORY SERVER DEPENDING ON THE AMOUNT OF FREE DISK SPACE

How Directory Server behaves when when you configure monitoring depends on the amount of remaining free space:

- If the free disk space reaches the defined threshold, Directory Server:
  - Disables verbose logging
  - Disables access access logging
  - Deletes archived log files

NOTE

Directory Server always continues writing error logs, even if the threshold is reached.

- If the free disk space is lower than the half of the configured threshold, Directory Server shuts down within a defined grace period.
- If the available disk space is ever lower than 4 KB, Directory Server shuts down immediately.

If disk space is freed up, then Directory Server aborts the shutdown process and re-enables all of the previously disabled log settings.

8.2. CONFIGURING LOCAL DISK MONITORING USING THE COMMAND LINE

Directory Server can monitor the free disk space on the file systems that contain the configuration, transaction log, and database directories. Depending on the remaining free space, Directory Server disables certain logging features or shuts down.

Procedure

1. Enable the disk monitoring feature, set a threshold value and a grace period:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-disk-monitoring=on nsslapd-disk-monitoring-threshold=3221225472 nsslapd-
disk-monitoring-grace-period=60
```

   This command sets the threshold of free disk space to 3 GB (3,221,225,472 bytes) and the grace period to 60 seconds.

2. Optional: Configure Directory Server not to disable access logging or delete archived logs:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-disk-monitoring-logging-critical=on
```

3. Restart the instance:

```
# dsctl instance_name restart
```

8.3. CONFIGURING LOCAL DISK MONITORING USING THE WEB CONSOLE

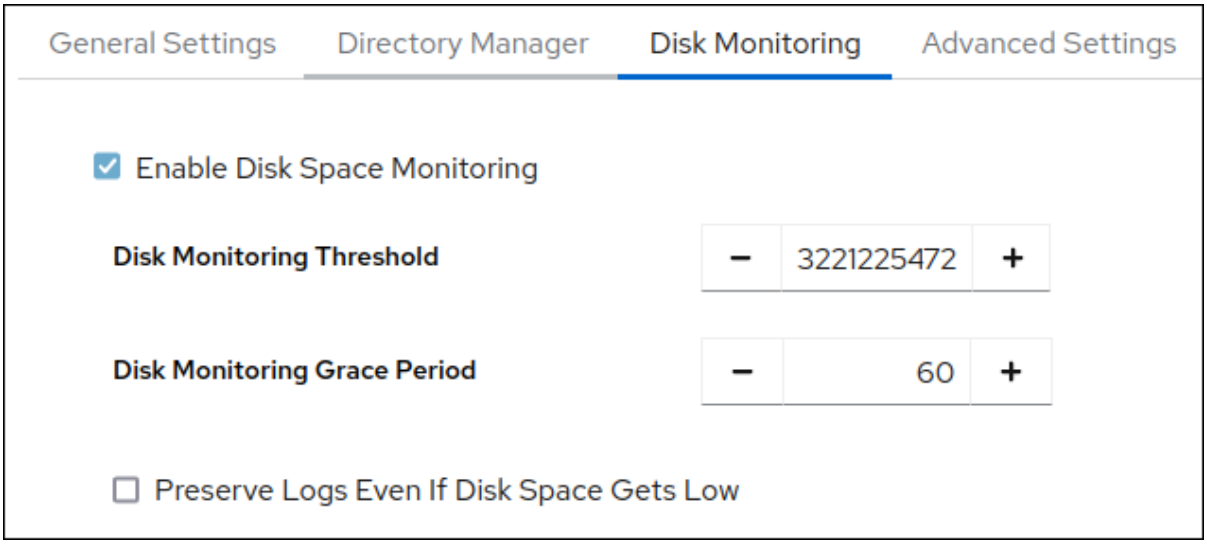
Directory Server can monitor the free disk space on the file systems that contain the configuration, transaction log, and database directories. Depending on the remaining free space, Directory Server disables certain logging features or shuts down.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Server → Server Settings → Disk Monitoring**.
2. Select **Enable Disk Space Monitoring**. Set the threshold in bytes and the grace period in minutes:



   This example sets the monitoring threshold to 3 GB (3,221,225,472 bytes) and the time before Directory Server shuts down the instance after reaching the threshold to 60 minutes.

3. Optional: Select **Preserve Logs Even If Disk Space Gets Low**
4. Click **Save Settings**.
5. Click **Actions** in the top right corner, and select **Restart Instance**.

CHAPTER 9. TUNING TRANSACTION LOGGING

Every Directory Server instance contains a transaction log which logs updates of databases it manages. Whenever a directory database operation, such as a modify operation, 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 record in the database file containing the entries and updating all 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 guarantees that an update operation is atomic. Either the entire operation succeeds permanently and irrevocably, or it fails.

Periodically, Directory Server 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 server automatically detects the error condition and uses the database transaction log to recover the database.

Although database transaction logging, flush the database, trimming, 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 the performance.

9.1. CHANGING THE DATABASE CHECKPOINT INTERVAL USING THE COMMAND LINE

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

By default, Directory Server sends a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval can increase the performance of directory write operations. However, it can 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.

Procedure

- Change the checkpoint interval, for example, to 120 seconds:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --
checkpoint-interval=120
```

9.2. CHANGING THE DATABASE CHECKPOINT INTERVAL USING THE WEB CONSOLE

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

By default, Directory Server sends a checkpoint entry to the database transaction log every 60 seconds. Increasing the checkpoint interval can increase the performance of directory write operations. However,

it can 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.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Database → Global Database Configuration**.
2. Click **Show Advanced Settings**.
3. Update the value in the **Database Checkpoint Interval** field.
4. Click **Save Configuration**.

9.3. 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 updates, each LDAP operation is treated as a single database transaction. Each LDAP operation is both atomic and durable.

WARNING

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

When you disable durable transaction logging, Directory Server writes every directory database operation to the database transaction log file but it 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.

Note that you cannot change the **nsslapd-db-durable-transaction** parameter if Directory Server is running.

Procedure

1. Stop the instance:

```
# dsctl instance_name stop
```

2. Edit the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file, and set the **nsslapd-db-durable-transaction** parameter in the **cn=config,cn=ldbm database,cn=plugins,cn=config** entry to **off**:

```
dn: cn=config,cn=ldbm database,cn=plugins,cn=config
...
nsslapd-db-durable-transaction: off
...
```

3. Start the instance:

```
# dsctl instance_name start
```

CHAPTER 10. MANAGING CACHE SETTINGS

Directory Server uses the following caches:

- The entry cache, which contains individual directory entries.
- The distinguished name (DN) cache is used to associate DNs and relative distinguished names (RDN) with entries.
- The database cache, which contains the database index files ***.db** 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.

10.1. HOW THE CACHE-AUTOSIZE AND CACHE-AUTOSIZE-SPLIT PARAMETERS INFLUENCE THE DATABASE AND ENTRY CACHE SIZES

By default, Directory Server uses an auto-sizing feature to optimize the size of both the database and entry cache on the hardware resources of the server when the instance starts.

IMPORTANT

Red Hat recommends to use the auto-sizing feature and not to set cache sizes manually.

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

nsslapd-cache-autosize

These settings control 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 Directory Server uses for the database cache. The server uses the remaining percentage for the entry cache.
- Using more than 1.5 GB RAM for the database cache does not improve the performance. Therefore, Directory Server limits the database cache to 1.5 GB.

By default, Directory Server uses the following defaults values:

- **nsslapd-cache-autosize: 25**
- **nsslapd-cache-autosize-split: 25**

- **nsslapd-dbcachesize: 1,536 MB**

Using these settings, 25% of the system's free RAM is used (**nsslapd-cache-autosize**). From this memory, the server uses 25% for the database cache (**nsslapd-cache-autosize-split**) and the remaining 75% for the entry cache.

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

Table 10.1. Cache sizes if both nsslapd-cache-autosize and nsslapd-cache-autosize-split use their default values

GB of free RAM	Database cache size	Entry cache size
1 GB	64 MB	192 MB
2 GB	128 MB	384 MB
4 GB	256 MB	768 MB
8 GB	512 MB	1,536 MB
16 GB	1,024 MB	3,072 MB
32 GB	1,536 MB	6,656 MB
64 GB	1,536 MB	14,848 MB
128 GB	1,536 MB	31,232 MB

10.2. REQUIRED CACHE SIZES

The **dsconf monitor dbmon** command enables you to monitor cache statistics at runtime. To display the statistics, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com monitor dbmon
DB Monitor Report: 2022-02-24 10:25:16
--------------------------------------------------------
Database Cache:
 - Cache Hit Ratio:     50%
 - Free Space:          397.31 KB
 - Free Percentage:     2.2%
 - RO Page Drops:       0
 - Pages In:            2934772
 - Pages Out:           219075

Normalized DN Cache:
 - Cache Hit Ratio:     60%
 - Free Space:          19.98 MB
 - Free Percentage:     99.9%
 - DN Count:            100000
 - Evictions:           9282348
```

```
Backends:
 - dc=example,dc=com (userroot):
  - Entry Cache Hit Ratio:       66%
  - Entry Cache Count:           50000
  - Entry Cache Free Space:      2.0 KB
  - Entry Cache Free Percentage: 0.8%
  - Entry Cache Average Size:    8.9 KB
  - DN Cache Hit Ratio:          21%
  - DN Cache Count:              100000
  - DN Cache Free Space:         4.29 MB
  - DN Cache Free Percentage:    69.8%
  - DN Cache Average Size:       130.0 B
```

Optionally, pass the **-b *back_end*** or **-x** option to the command to display the statistics for a specific back end or the index.

If your caches are sufficiently sized, the number in **DN Cache Count** matches the values in the **Cache Count** backend entries. Additionally, if all of the entries and DNs fit within their respective caches, the **Entry Cache Count** count value matches the **DN Cache Count** value.

The output of the example shows:

- Only 2.2% free database cache is left:

  ```
  Database Cache:
   ...
   - Free Space:         397.31 KB
   - Free Percentage:    2.2%
  ```

  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** 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 Setting the database cache size using the command line .

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

  ```
  Backends:
   - dc=example,dc=com (userroot):
     ...
     - DN Cache Count:              100000
     - DN Cache Free Space:         4.29 MB
     - DN Cache Free Percentage:    69.8%
     - DN Cache Average Size:       130.0 B
  ```

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

  To set the DN cache, see Setting the DN cache size using the command line .

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

  ```
  Backends:
   - dc=example,dc=com (userroot):
  ```

```
...
 - Entry Cache Count:           50000
 - Entry Cache Free Space:      2.0 KB
 - Entry Cache Free Percentage: 0.8%
 - Entry Cache Average Size:    8.9 KB
```

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, the cache must be increased to minimum of 890 MB (100000 DNs * 8,9 KB average entry size). However, Red Hat recommends to round the minimum required size to the next highest GB and double the result. In this example, the entry cache should be set to 2 Gigabytes.

To set the entry cache, see Setting the entry cache size using the command line .

10.3. SETTING THE DATABASE CACHE SIZE USING THE COMMAND LINE

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.

Procedure

1. Disable automatic cache tuning

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --cache-autosize=0
```

2. Manually set the database cache size:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --dbcachesize=268435456
```

   Specify the database cache size in bytes. In this example, the command sets the database cache to 256 MB.

3. Restart the instance:

```
# dsctl instance_name restart
```

10.4. SETTING THE DATABASE CACHE SIZE USING THE WEB CONSOLE

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.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Database → Global Database Configuration**.
2. Deselect **Automatic Cache Tuning**.
3. Click **Save Config**.
4. Enter the database cache size in bytes, such as **268435456** for 256 MB, into the **Database Cache Size** field.
5. Click **Save Config**.
6. Click **Actions** in the top right corner, and select **Restart Instance**.

10.5. SETTING THE DN CACHE SIZE USING THE COMMAND LINE

Directory Server uses the **entryrdn** index to associate distinguished names (DN) and relative distinguished names (RDN) with entries. It enables the server to efficiently rename subtrees, move entries, and perform **moddn** operations. The server uses the DN cache to cache the in-memory representation of the **entryrdn** index to avoid expensive file I/O and transformation operations.

If you do not use the auto-tuning feature, for best performance, especially with but not limited to renaming entries and moving 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.

Procedure

1. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com suffix list
dc=example,dc=com (userroot)
```

   This command displays the name of the back end database next to each suffix. You require the suffix's database name in the next step.

2. Set the DN cache size:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --dncache-memsize=20971520 userRoot
```

   This command sets the DN cache for the **userRoot** database to 20 megabytes.

3. Restart the instance:

```
# dsctl instance_name restart
```

10.6. SETTING THE DN CACHE SIZE USING THE WEB CONSOLE

Directory Server uses the **entryrdn** index to associate distinguished names (DN) and relative distinguished names (RDN) with entries. It enables the server to efficiently rename subtrees, move entries, and perform **moddn** operations. The server uses the DN cache to cache the in-memory representation of the **entryrdn** index to avoid expensive file I/O and transformation operations.

If you do not use the auto-tuning feature, for best performance, especially with but not limited to renaming entries and moving 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.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Database** → **Suffixes** → *suffix_name*.
2. Enter the DN cache size in bytes to the **DN Cache Size** field.
3. Click **Save Configuration**.
4. Click **Actions** in the top right corner, and select **Restart Instance**.

10.7. SETTING THE ENTRY CACHE SIZE USING THE COMMAND LINE

Directory Server uses the entry cache to store directory entries that are used during search and read operations. Setting the entry cache to a size that enables 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 distinguished names (DN) 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.

Procedure

1. Disable automatic cache tuning:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --cache-autosize=0
```

2. Display the suffixes and their corresponding back end:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com suffix list
dc=example,dc=com (userroot)
```

   This command displays the name of the back end database next to each suffix. You require the suffix's database name in the next step.

3. Set the entry cache size in bytes for the database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --
cache-memsize=2147483648 userRoot
```

This command sets the entry cache for the **userRoot** database to 2 gigabytes.

4. Restart the instance:

```
# dsctl instance_name restart
```

10.8. SETTING THE ENTRY CACHE SIZE USING THE WEB CONSOLE

Directory Server uses the entry cache to store directory entries that are used during search and read operations. Setting the entry cache to a size that enables 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 distinguished names (DN) 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.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Database** → **Suffixes** → *suffix_name* → **Settings**.
2. Disable the **Automatic Cache Tuning** setting.
3. Click **Save Configuration**.
4. Click **Actions** in the top right corner, and select **Restart Instance**.
5. Navigate to **Database** → **Suffixes** → *suffix_name* → **Settings**.
6. Set the size of the database cache in the `Entry Cache Size field.
7. Click **Save Configuration**.
8. Click **Actions** in the top right corner, and select **Restart Instance**.

CHAPTER 11. IMPROVING IMPORT PERFORMANCE

Very large attribute sizes or a large number of entries can negatively impact server performance during import operations. This section describes how to tune Directory Server settings to improve the import performance.

11.1. TUNING DIRECTORY SERVER FOR LARGE DATABASE IMPORTS AND IMPORTS WITH LARGE ATTRIBUTE VALUES

Use the import cache autosizing feature for operations like:

- importing a very large database
- importing a database with large attributes, such as binary attributes that store certificate chains or images

NOTE

Offline imports are faster than online ones. Consider using offline imports where possible.

You can configure the import cache autosizing feature with the **nsslapd-import-cache-autosize** attribute. By default, Directory Server enables the import cache autosizing for the **ldif2db** operation only, automatically allocating 50% of the free physical memory for the import cache.

For further details, see the description of the nsslapd-import-cache-autosize attribute in the Configuration, Command, and File Reference document.

CHAPTER 12. TUNING SERVER CONNECTION MANAGEMENT

The **nsslapd-numlisteners** attribute specifies the number of listener threads the Directory Server can use to monitor established connections. You can improve the response times when the server experiences a large number of client connection by increasing the attribute value.

12.1. MANAGING THE NUMBER OF CONNECTION LISTENER THREADS BY USING THE COMMAND LINE

You can manage the number of connection listener threads by using the command line. The default value is **1**.

Procedure

1. List the number of connection listener threads:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-numlisteners
```

2. Modify the number of connection listener threads:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-numlisteners=4
```

3. Restart the instance:

```
# dsctl instance_name restart
```