univention
be open.

Univention Corporate Server - Performance guide

Release 5.0

May 03, 2024

Contents

By default UCS is suitable for environments with up to 5,000 users. This document describes configuration modifications which can increase performance in larger environments.

1 OpenLDAP and listener/notifier domain replication

As a core element in the operation and administration of a UCS domain, the performance of the LDAP server plays a central role in the overall performance.

1.1 Indexes

Comparable with other database systems, OpenLDAP uses indexes about commonly requested attributes. For indexed attributes a search isn't performed through the full database contents, but over an optimized subsection.

With recent UCS versions, the indexes are occasionally expanded and automatically activated. You can deactivate the automatic activation with the UCR variable `ldap/index/autorebuild`. In this case, you must set the indexes manually to ensure that there is no loss of performance as a result.

Optimizing an LDAP index requires some thoughts before hand. It isn't recommended to add all wanted attributes to the LDAP index. The index requires maintenance and updating an index costs performance on write and sometimes also on read operations. Evaluate your frequent searches and add the frequent used search attributes to the LDAP index.

See also:

Can more indexes improve performance?[1] **in the *OpenLDAP Faq-O-Matic***
for more information about indexes influence the performance in OpenLDAP

To add LDAP attributes to an index, use the following steps:

1. Stop the OpenLDAP server.
2. Add or remove the LDAP attributes in the indexes by changing the respective UCR variable.
3. Run the command **`slapindex`** to re-index the entries in the LDAP database.
4. Start the OpenLDAP server.

The UCR variables *`ldap/index/eq`* (page 3), *`ldap/index/pres`* (page 3), *`ldap/index/sub`* (page 3), and *`ldap/index/approx`* (page 2) control the LDAP index configuration for the OpenLDAP server. After changing one of those variables, UCR rewrites the OpenLDAP server configuration file.

Consider the following hints for your index:

- Negations don't use an index and therefore negation searches suffer performance.
- Range comparisons like >= and <= only work for non-string syntaxes like *integer* and *generalizedTime*. They use the *`eq`* (page 3) index for equality tests.

The following UCR variables control the LDAP index:

`ldap/index/approx`

This index tests for approximate matches. For example adding the `uid` attribute to the index, it corresponds to the LDAP filter `uid~=`*`value`* and finds all approximate objects.

See also:

[1] https://www.openldap.org/faq/data/cache/42.html

Ldapwiki: ApproxMatch[Page 3, 2]
Overview about approximate match in OpenLDAP search filters defined in **RFC 4511**[3].

ldap/index/eq
This index tests for equality. For example adding the `uid` attribute to the index, it corresponds to the LDAP filter `uid=value` and finds all objects with exact that `value`. LDAP uses this index as fallback for a missing presence index in *ldap/index/pres* (page 3).

ldap/index/pres
This index tests for the presence. For example adding the `uid` attribute to the index, it corresponds to the LDAP filter `uid=*` and finds all objects that have something within the `uid` attribute.

ldap/index/sub
This index runs a sub string search. For example adding the `uid` attribute to the index, it corresponds to the LDAP filter `uid=value*` and finds all objects that match with the filter including the wildcard.

To determine whether OpenLDAP uses not-indexed variables, you can activate OpenLDAP debug level `-1` and search for the string `not indexed` in the log file `/var/log/syslog`. For example:

```
$ ucr set ldap/debug/level=-1
$ systemctl restart slapd
$ grep 'not indexed' /var/log/syslog
```

1.2 Configuration of the database backend

The memory mapped database (MDB) has been used for new installations since UCS 4.0. If BDB is still in use, a migration to MDB should be performed for *amd64* systems. The database backend can be controlled via the UCR variable `ldap/database/type`. A migration can be performed as follows:

```
$ systemctl stop slapd
$ slapcat -f /etc/ldap/slapd.conf -l ldif
$ mkdir /var/lib/univention-ldap/ldap.BACKUP
$ mv /var/lib/univention-ldap/ldap/* /var/lib/univention-ldap/ldap.BACKUP
$ ucr set ldap/database/type=mdb
$ slapadd -f /etc/ldap/slapd.conf -l ldif
$ systemctl start slapd
```

By default the memory mapped database needs more I/O operations than the BDB backend. With the Univention Configuration Registry Variable `ldap/database/mdb/envflags` this behavior can be configured. The following flags can be set (multiple values are separated by spaces):

nosync
Specify that on-disk database contents should not be immediately synchronized with in memory changes. Enabling this option may improve performance at the expense of data security. In particular, if the operating system crashes before changes are flushed, some number of transactions may be lost. By default, a full data flush/sync is performed when each transaction is committed.

nometasync
Flush the data on a commit, but skip the sync of the meta page. This mode is slightly faster than doing a full sync, but can potentially lose the last committed transaction if the operating system crashes. If both `nometasync` and `nosync` are set, the `nosync` flag takes precedence.

writemap
Use a writable memory map instead of just read-only. This speeds up write operations but makes the database vulnerable to corruption in case any bugs in `slapd` cause stray writes into the memory mapped region.

mapasync
When using a writable memory map and performing flushes on each commit, use an asynchronous flush instead

[2] https://ldapwiki.com/wiki/ApproxMatch
[3] https://datatracker.ietf.org/doc/html/rfc4511.html

of a synchronous flush (the default). This option has no effect if `writemap` has not been set. It also has no effect if `nosync` is set.

`nordahead`
Turn off file read-ahead. Usually the OS performs read-ahead on every read request. This usually boosts read performance but can be harmful to random access read performance if the system's memory is full and the DB is larger than RAM.

1.3 OpenLDAP ACLs

Access to the information contained in the LDAP directory is controlled by access control lists (ACLs) on the server side. General information on the configuration of ACLs in UCS can be found in Access control for the LDAP directory[4] in *Univention Corporate Server - Manual for users and administrators* [1].

Nested groups are also supported. The Univention Configuration Registry Variable `ldap/acl/nestedgroups`[5] can be used to deactivate the nested groups function for LDAP ACLs, which will result in a speed increase for directory requests.

1.4 Univention Directory Listener

The Univention Directory Listener can perform safety checks to prevent a user name being added into a group twice. These checks add some overhead to replication and can be deactivated by setting the Univention Configuration Registry variables `listener/memberuid/skip` and `listener/uniquemember/skip` to `no`. Starting with UCS 3.1 the variables are not set and the checks are not activated any longer by default.

2 Name Service Cache Daemon (NSCD)

Name resolutions can be cached by the *Name Service Cache Daemon* (NSCD) in order to speed up frequently recurring requests for unchanged data. Thus, if a repeated request occurs, instead of querying the LDAP server, the data are simply drawn directly from the cache.

The size of the cache held by the NSCD is preconfigured for an environment with 5,000 users. If more users or hosts are created, the cache should be enlarged as otherwise it will not be possible to cache enough entries.

The following Univention Configuration Registry variables can be set:

- `nscd/hosts/size` should be at least the same as the number of all the computers entered in the DNS.
- `nscd/passwd/size` should be at least the same as the number of users.

To allow an efficient cache allocation, the value selected should always be a prime number, in case of doubt the next highest prime number should be selected.

A script can be downloaded from https://updates.software-univention.de/download/scripts/nscdCachesize.sh which suggests corresponding values based on the objects currently included in the system.

[4] https://docs.software-univention.de/manual/5.0/en/domain-ldap/ldap-directory.html#domain-ldap-acls
[5] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-ldap-acl-nestedgroups

3 Performance during initial provisioning of users and groups

There are several ways in which you can provision users and groups into UCS. Each method has its own performance implications and use cases. Especially for large environments it's important that you choose an efficient method.

The following recommendations can improve performance when creating large numbers of users and adding them to groups:

1. Use the *UDM Python library* in case you can do the provisioning locally on the UCS system.

   Use the UDM HTTP REST API[6] to provision users and groups to a remote UCS system.

   It isn't recommended to use the UDM command line interface, because it's significantly slower than the previously mentioned options.

2. Create the users first and then the groups. This prevents unnecessary LDAP operations, because after the creation of the users, LDAP only needs to update the groups one time.

3. For the duration of the provisioning, deactivate the automatic update of the primary group, typically `Domain Users`, when you create or remove a user. Set the Univention Configuration Registry variable `directory/manager/user/primarygroup/update` to `false`.

You can use the following example as a guide for using the *UDM Python library*:

```
#!/usr/bin/python3

from univention.admin import modules, uldap
from univention.config_registry import ucr


lo, position = uldap.getAdminConnection()
base = ucr['ldap/base']

modules.update()

users = modules.get('users/user')
modules.init(lo, position, users)

groups = modules.get('groups/group')
modules.init(lo, position, groups)

def create_user(name):
    position.setDn('cn=users,%s' % (base,))
    res = users.lookup(None, lo, "uid=%s" % name)
    if res:
        user = res[0]
    else:
        user = users.object(None, lo, position)
        user.open()
        user["lastname"] = name
        user["firstname"] = name
        user["password"] = "univention"
        user["username"] = name
        user.create()
    return user.dn

def create_or_modify_group(name, members=None):
    """
    Parameters:
        name (str): name of the group
        members (list[str]): list of user DNs
    """
```

(continues on next page)

[6] https://docs.software-univention.de/developer-reference/5.0/en/udm/rest-api.html#udm-rest-api

(continued from previous page)

```
    position.setDn('cn=groups,%s' % (base,))
    res = groups.lookup(None, lo, "cn=%s" % name)
    if res and members:
        group = res[0]
        group.open()
        group["users"].extend(members)
        group.modify()
    else:
        group = groups.object(None, lo, position)
        group.open()
        group["name"] = name
        if members:
            group["users"] = members
        group.create()

username_list = ["exampleuser1", "exampleuser2"]
userdn_list = []
for name in username_list:
    userdn = create_user(name)
    if userdn:
        userdn_list.append(userdn)

if userdn_list:
    create_or_modify_group("examplegroup1", userdn_list)
```

4 Performance issues during the join process

The size of the UCS domain can have an impact on the duration of the join process. Here is some information how to deal with such problems.

4.1 Samba

One of the join scripts for samba requires that the samba connector has synchronized all domain objects into samba. This script has a timeout of `3h` (from UCS 4.4-7 on). This is sufficient for normal sized environments. But in large environments this script may hit the timeout and abort the join process. To increase the timeout the Univention Configuration Registry Variable `create/spn/account/timeout` can be set prior to the join process.

The join scripts `97univention-s4-connector` and `98univention-samba4-dns` wait for the replication of the DNS record of the joining system to verify that the local Samba backed DNS server can answer requests for Active Directory related requests. By default the scripts wait for 600 seconds, but in case there are a lot of objects that need to be replicated (e.g. DNS zones) then this default may be too short. In that case the timeout can be adjusted by setting the Univention Configuration Registry Variable `join/samba/dns/replication/timeout` to a bigger value before joining.

Samba traditionally uses TDB as backend database storage, that has an internal 32 bit address space limitation. UCS supports provisioning Samba using LMDB instead, which doesn't have this strict limitation. For more information, see *Lightning Memory-Mapped Database Manager (LMDB) documentation* [2]. KB 18014[7] describes how to migrate a productive UCS domain.

To use LMDB instead of TDB, set the corresponding Univention Configuration Registry Variable `samba/database/backend/store` to `mdb` before you install the app **`Active Directory-compatible Domain Controller`** in your UCS domain.

The Univention Configuration Registry Variable `samba/database/backend/store/size` defines the current maximal size of the individual backend database store files and has the default value of `8GB`. Since there is one

[7] https://help.univention.com/t/18014

backend storage file per Active Directory naming context, this amounts to a total of `40GiB`. Take care that the storage can accommodate this amount of space.

If required, you can increase the value monotonically. After changing the Univention Configuration Registry Variable, you need to restart Samba, so that the value can take effect. You can check the number of used storage pages, `4KiB` each, by running the command **`mdb_stat -nef`** on the individual files and calculating `Number of pages used` minus `Free pages`. The value at `Max pages` shows the current effective limit. The backend storage files locate in `/var/lib/samba/private/sam.ldb.d/` and have the file extension `.ldb`.

5 Local group cache

By default the group cache is regenerated every time changes are made to a group. This avoids cache effects whereby group memberships only become visible for a service after the next scheduled group cache rewrite (by default once a day and after 15 seconds of inactivity in the Univention Directory Listener). In larger environments with a lot of group changes, this function should be deactivated by setting the Univention Configuration Registry Variable `nss/group/cachefile/invalidate_on_changes`[8] to `false`. This setting takes effect immediately and does not require a restart of the Univention Directory Listener.

When the group cache file is being generated, the script verifies whether the group members are still present in the LDAP directory. If only the Univention Management Console is used for the management of the LDAP directory, this additional check is not necessary and can be disabled by setting the Univention Configuration Registry Variable `nss/group/cachefile/check_member`[9] to `false`.

6 UCS management system

6.1 Disabling automatic search

By default all objects are automatically searched for in the domain management modules of the Univention Management Console. This behavior can be disabled by setting the Univention Configuration Registry Variable `directory/manager/web/modules/autosearch` to `0`.

6.2 Imposing a size limit for searches

The Univention Configuration Registry Variable `directory/manager/web/sizelimit` is used to impose an upper limit for search results. If, e.g., this variable is set to `2000` (as is the default), searching for more than 2000 users would not be performed and instead the user is asked to refine the search.

6.3 Adjusting the limit on open file descriptors

The Univention Configuration Registry Variable `umc/http/max-open-file-descriptors` is used to impose an upper limit on open file descriptors of the **`univention-management-console-web-server`**. The default is `65535`.

[8] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-nss-group-cachefile-invalidate_on_changes
[9] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-nss-group-cachefile-check_member

6.4 Vertical performance scaling

A single Univention Management Console instance does not use multiple CPU cores by design, therefore it can be beneficial to start multiple instances. Set the following Univention Configuration Registry Variable `umc/http/processes` and restart the Univention Management Console:

```
$ systemctl restart apache2 \
  univention-management-console-server
```

The number of instances to configure depends on the workload and the server system. As a general rule of thumb these should not be higher than the machines CPU cores. Good throughput values had resulted in tests with the following combinations:

- Automatically detect available CPU cores: `umc/http/processes=0`
- 6 CPU cores: `umc/http/processes=3`
- 16 CPU cores: `umc/http/processes=15`
- 32 CPU cores: `umc/http/processes=25`

Note that the number of Apache processes may also need to be increased for the customization to take effect.

7 Further services and components

7.1 Apache

In environments with many simultaneous accesses to the web server or Univention Portal and Univention Management Console, it may be advisable to increase the number of possible Apache processes or reserve processes. This can be achieved via the UCR variables `apache2/server-limit`, `apache2/start-servers`, `apache2/min-spare-servers` and `apache2/max-spare-servers`. After setting, the Apache process must be restarted via the command **`systemctl restart apache2`**.

Detailed information about useful values for the UCR variables can be found at ServerLimit Directive[10] and StartServers Directive[11] in *Apache HTTP Server Version 2.4 Documentation* [3].

7.2 SAML

By default, SAML assertions are valid for `300` seconds and must be renewed by clients no later than then to continue using them. In scenarios where refreshing SAML assertions at such short intervals is too expensive (for clients or servers), the lifetime of SAML assertions can be increased via the UCR variable `umc/saml/assertion-lifetime`. This can be achieved on each UCS system with the role Primary Directory Node or Backup Directory Node by executing the following commands:

```
$ ucr set umc/saml/assertion-lifetime=3600
$ cd /usr/share/univention-management-console/saml/
$ ./update_metadata --binddn "$USERDN" --bindpwdfile "$FILENAME"
```

`$USERDN` has to be replaced with a valid DN of a user, that is member of the group `Domain Admins` and the file specified by `$FILENAME` has to contain the corresponding password of that user.

It should be noted that increasing the lifetime has security implications that should be carefully considered.

[10] https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#serverlimit
[11] https://httpd.apache.org/docs/2.4/en/mod/mpm_common.html#startservers

7.3 Squid

If the Squid proxy service is used with NTLM authentication, up to five running NTLM requests can be processed in parallel. If many proxy requests are received in parallel, the Squid user may occasionally receive an authentication error. The number of parallel NTLM authentication processes can be configured with the Univention Configuration Registry Variable `squid/ntlmauth/children`.

7.4 BIND

BIND can use two different backend for its configuration: OpenLDAP or the internal LDB database of Samba/AD. The backend is configured via the Univention Configuration Registry Variable `dns/backend`[12]. On UCS Directory Nodes running Samba/AD, the backend **must not** be changed to OpenLDAP.

When using the Samba backend, a search is performed in the LDAP for every DNS request. With the OpenLDAP backend, a search is only performed in the directory service if the DNS data has changed. For this reason, using the OpenLDAP backend can reduce the load on a Samba/AD domain controller.

7.5 Kernel

In medium and larger environments the maximum number of open files allowed by the Linux kernel may be set too low by default. As each instance requires some unswappable memory in the Linux kernel, too many objects may lead to a resource depletion and denial-of-service problems in multi-user environments. Because of that the number of allowed file objects is limited by default.

The maximum number of open files can be configured on a per-user or per-group basis. The default for all users can be set through the following Univention Configuration Registry Variables:

`security/limits/user/`*`default`*`/hard/nofile`
: The hard limit defines the upper limit a user can assign to a process. The default is `32768`.

`security/limits/user/`*`default`*`/soft/nofile`
: The soft limit defines the default settings for the processes of the user. The default is `32768`.

A similar problem exists with the Inotify sub-system of the kernel, which can be used by all users and applications to monitor changes in file systems.

`kernel/fs/inotify/max_user_instances`
: The upper limit of inotify services per user ID. The default is `511`.

`kernel/fs/inotify/max_user_watches`
: The upper limit of files per user which can be watched by the inotify service. The default is `32767`.

`kernel/fs/inotify/max_queued_events`
: The upper limit of queued events per inotify instance. The default is `16384`.

7.6 Samba

Samba uses its own mechanism to specify the maximum number of open files. This can be configured through the Univention Configuration Registry Variable `samba/max_open_files`. The default is `32808`.

If the log file `/var/log/samba/log.smbd` contains errors like `Failed to init inotify - Too many open files`, the kernel and Samba limits should be increased and the services should be restarted.

[12] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-dns-backend

7.7 System statistics

The log file `/var/log/univention/system-stats.log` can be checked for further performance analyses. The system status is logged every *30 minutes*. If more regular logging is required, it can be controlled via the UCR variable `system/stats/cron`[13].

7.8 Dovecot high-performance mode

Univention Corporate Server configures Dovecot to run in *High-security mode* by default. Each connection is served by a separate login process. This security has a price: for each connection at least two processes must run.

Thus installations with 10.000s of users hit operating system boundaries. For this case Dovecot offers the *High-performance mode*. To activate it, login processes are allowed to serve more than one connection. To configure this run

```
$ ucr mail/dovecot/limits/imap-login/service_count=0
```

If `client_limit=1000` and `process_limit=100` are set, only 100 login processes are started, but each serves up to 1000 connections — a total of 100.000 connections.

The cost of this is that if a login process is compromised, an attacker might read the login credentials and emails of all users this login process is serving.

To distribute the load of the login processes evenly between CPU cores, `mail/dovecot/limits/imap-login/process_min_avail` should be set to the number of CPU cores in the system.

7.9 UDM HTTP REST API performance scaling

A single UDM HTTP REST API instance does not use multiple CPU cores by design, therefore it can be beneficial to start multiple instances. By setting the Univention Configuration Registry Variable `directory/manager/rest/processes` the number of processes can be increased. Afterwards the UDM HTTP REST API needs to be restarted:

```
$ systemctl restart univention-directory-manager-rest
```

The number of instances to configure depends on the workload and the server system. As a general rule of thumb these should not be higher than the machines CPU cores. With `directory/manager/rest/processes=0` all available CPU cores are used.

8 Bibliography

References

[1] *Univention Corporate Server - Manual for users and administrators*. Univention GmbH, 2021. URL: https://docs.software-univention.de/manual/5.0/en/.

[2] *Lightning Memory-Mapped Database Manager (LMDB) documentation*. Symas Corporation, 2015. URL: http://www.lmdb.tech/doc/.

[3] *Apache HTTP Server Version 2.4 Documentation*. The Apache Software Foundation. URL: https://httpd.apache.org/docs/2.4/en/.

[13] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-system-stats-cron

Index

S

U