opentext™

eDirectory Performance Tuning and Troubleshooting

EMEA TTP

March 2023 | Rodrigo Gomez

Agenda

Search Performance

Update/write performance

Avoiding resource starvation

Q&A

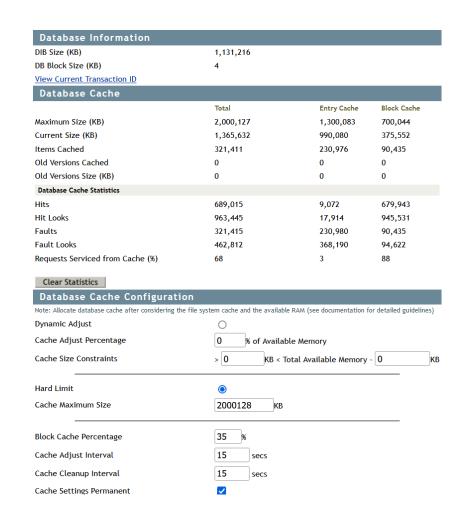# Search Performance

Search Performance

Searches in eDirectory can be done via LDAP or directly via NCP. Example of the later ones are searches with iManager or by IDM drivers.

The two main factors that influence the search performance are:
- eDirectory cache configuration
- Index configuration

Let's time some searches…

Search Performance – Sufficient cache

Database Information

DIB Size (KB)	1,131,216
DB Block Size (KB)	4

View Current Transaction ID

Database Cache

	Total	Entry Cache	Block Cache
Maximum Size (KB)	2,000,127	1,300,083	700,044
Current Size (KB)	1,365,632	990,080	375,552
Items Cached	321,411	230,976	90,435
Old Versions Cached	0	0	0
Old Versions Size (KB)	0	0	0

Database Cache Statistics

Hits	689,015	9,072	679,943
Hit Looks	963,445	17,914	945,531
Faults	321,415	230,980	90,435
Fault Looks	462,812	368,190	94,622
Requests Serviced from Cache (%)	68	3	88

[Clear Statistics]

Database Cache Configuration

Note: Allocate database cache after considering the file system cache and the available RAM (see documentation for detailed guidelines)

Dynamic Adjust	○
Cache Adjust Percentage	0 % of Available Memory
Cache Size Constraints	> 0 KB < Total Available Memory - 0 KB
Hard Limit	●
Cache Maximum Size	2000128 KB
Block Cache Percentage	35 %
Cache Adjust Interval	15 secs
Cache Cleanup Interval	15 secs
Cache Settings Permanent	☑

Lab setup: 100K users, 80K roles

DIB size is 1.1Gb, sufficiently large cache.

Search to be performed is:

```
time ldapsearch -x -D "cn=admin,ou=sa,o=system" -w
novell -LLL -b "" -s sub "directReports=*" 1.1
```

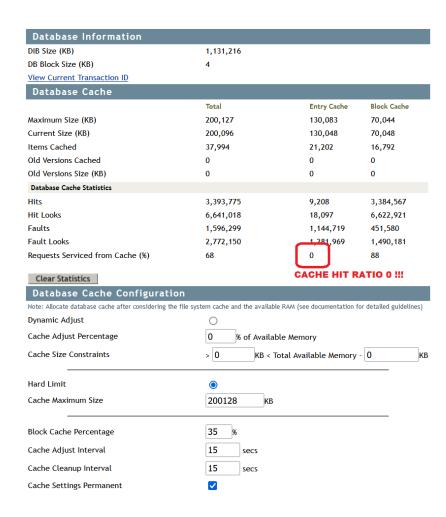Result with cold cache:
- **real   0m2.215s**

Result with warm cache:
- real   0m0.167s

The screenshot shows the cache view right after the first search is performed

Search Performance – Cache too small

Database Information			
DIB Size (KB)	1,131,216		
DB Block Size (KB)	4		
View Current Transaction ID			

Database Cache			
	Total	Entry Cache	Block Cache
Maximum Size (KB)	200,127	130,083	70,044
Current Size (KB)	200,096	130,048	70,048
Items Cached	37,994	21,202	16,792
Old Versions Cached	0	0	0
Old Versions Size (KB)	0	0	0

Database Cache Statistics

Hits	3,393,775	9,208	3,384,567
Hit Looks	6,641,018	18,097	6,622,921
Faults	1,596,299	1,144,719	451,580
Fault Looks	2,772,150	1,281,969	1,490,181
Requests Serviced from Cache (%)	68	0	88

CACHE HIT RATIO 0 !!!

Clear Statistics

Database Cache Configuration
Note: Allocate database cache after considering the file system cache and the available RAM (see documentation for detailed guidelines)

Dynamic Adjust	○
Cache Adjust Percentage	0 % of Available Memory
Cache Size Constraints	> 0 KB < Total Available Memory - 0 KB
Hard Limit	●
Cache Maximum Size	200128 KB
Block Cache Percentage	35 %
Cache Adjust Interval	15 secs
Cache Cleanup Interval	15 secs
Cache Settings Permanent	☑

Cache reduced to 200Mb

Result with cold cache:

- real   0m2.203s

Result with warm cache:

- real 0m2.203s

Requests serviced from cache drops to 0 and stays below 10

Performance is close to 15 times worst. The cache can never be fully initialized.

eDirectory cache configuration

Two subsystems:

- Entry cache: Caches entries as they are represented in eDirectory – Prefer for reads
- Block cache: Caches entries as they are stored in disk – Prefer for writes

It can be modified via iMonitor -> Agent Configuration -> Database Cache. Changes take effect immediately

Cache configuration is stored in the _ndsdb.ini file

Rule of thumb: start with 2 times dib size, up to 4 Gb. Increase/decrease based on performance.

Block cache percentage can be used to distribute the allocated space.

opentext™

Search Performance – Index added

Add relevant index:

> ndsindex add -D cn=admin,ou=sa,o=system
> -w novell -s cn=vm-
> demohb,ou=servers,o=system
> 'DirectReportsIx;directReports;value'

Check it's online:

Cache size is now less relevant

Search result:

- real   0m0.017s

Tracing that the index is being used with the +RECM (ndstrace) or Storage Manager tag

Adding an index – monitor progress

The limber process starts the index addition. Progress can then be followed with the +RECM trace

The index gets first added to the NCP Server object.

Then it gets copied over to the [Pseudo Server] object, which can be accessed via iMonitor -> Agent Configuration

From there, the operations is passed down to Flaim.

The index goes through different stages until it becomes Online

Indexes

An index is a set of keys arranged in a way that significantly speeds up the task of finding any particular key within the index.

Constructed based on values of attributes in the entries

Maintained in memory in the block cache

eDirectory defines a default set of indexes for system attributes like parentID or ancestorsID

Default indexes are defined for attributes such as CN, Surname and Given Name

Indexes – cont.

Indexes can be of type presence, value, and substring.

- Presence indexes are only used for presence filters (i.e. cn=*)
- Substring indexes create a value indexes with all substring combinations of each value, which makes them very expensive

System creates automatically an index if there are more than 25 values of a given attribute or if any value is larger than 2048 bytes

- This behavior can be disabled with disablemovetoattrcontainer =1 in the _ndsdb.ini file
- Attributes then are shown in the Pseudo Server object in the **dsContainerReadyAttrs**
- You can see if the index has been added automatically in iMonitor. Index type is "system"
- System indexes are stored in a separate location by Flaim

Compound indexes

Compound indexes can be created to combine attributes

- IDM (Identity Apps) creates by default a series of them
- Compound indexes are always value indexes
- It is not possible to combine an attribute that has been added as a System index

The parameter –a in ndsindex adds AncestorsID information to the index information (from eDir 9.1)

- The index is displayed as a compound index with the attributes defined plus AncestorsID as an extra attribute to the attribute list

Other factors that affect search performance

LDAP search controls like VLV, Server Side Sort and Paged Search force the use of specific indexes, impacting performance

Member queries on paths that contain dynamic groups will spawn a new search for each dynamic group

Alias objects can have a big performance impact. Reduce the use in search intensive trees

Complex ACL calculations can have an impact on the search performance as well

Update/write performance

Avoid unnecessary writes - Login Update Interval

On each login, attribute Login Time is updated and the previous value is copied to Last Login Time -> Overkill for LDAP Binds

It's possible to control this behavior with the attribute sasUpdateLoginInfo:

- **0 or off:** Do not update any login attributes.
- **1:** Only update attributes that are required by intruder detection.
- **2:** Update all login attributes except unused user password policy attributes.
- **3 or on:** Update all login attributes.

The frequency of update is controlled by sasUpdateLoginTimeInterval:

- If the value is between **1** and **1440** minutes, the Login Time attribute is updated after the specified interval. The Last Login Time attribute will not be updated.

opentext™

Login Update Interval

Can be specified at:

- User
- Container of the user
- Partition root
- Login Policy -> Tree wide

Or at server level:

```
#cat /var/opt/novell/eDirectory/data/nmas.config
nmas LoginInfo 2
Nmas UpdateLoginTimeInterval 30
```

NDSD_TRY_NMASLOGIN_FIRST needs to be set for LDAP Binds

Identity Manager – DirXMLPasswordSyncStatus

This attribute is used for the functionality present in iManager -> Check Password Sync status

- The function reads this attribute and reports the status for each connected system with this setting enabled.

If this feature is not being used, the tracking of password sync status can be disabled with an Engine Control Value on a per driver basis.

Engine Control Values

Engine Controls For Server: cn=vm-demoHB,ou=servers,o=system

Write pending associations	false
Use password event values	false
Enable password synchronization status reporting	true

Information

Name:
dirxml.engine.pwd-sync-status

Display name:
Enable password synchronization status reporting

Type:
boolean

Description:
This control determines whether the DirXML Engine will report the status of subscriber channel password change events.

Reporting the status of subscriber channel password change events allows applications such as the Identity Manager User Application to monitor the synchronization progress of a password change that should be synchronized to the connected application.

OK

Identity Manager - DirXMLStatusLog

Avoid updating the Dirxml-StatusLog attribute

The attribute stores in eDirectory the last status results, information that can be retrieved as well from a log at level 0

The attribute is not synchronized to other servers, but it causes the modifiersName and Revision attributes to be updated.

Modify Object: driverset1.system

| Identity Manager | General |

Named Passwords | Global Config Values | **Log Level** | Status Log | Activation | Misc | Inspector

Log Level

- ○ Log errors
- ○ Log errors and warnings
- ○ Log specific events
- ○ Log XDAS events
- ○ Only update the last log time
- ⦿ Logging off

- ☑ Turn off logging to Driver Set, Subscriber and Publisher logs.

Maximum number of entries in the log (50 - 500):

Update Performance – Write locks

iMonitor -> Agent Activity shows information on long write locks

The image shows a 42 second lock by process DSA Modify Entry

The Wait column indicates how long other threads have waited for the write lock

In this situation, it's possible that too many threads get spawned and the server runs out of threads.

Identify what is causing the write lock. Usual suspects: DirXML-EntitlementResult, DirXML-StatusLog, pwdFailureTime

Update Performance – High Value count report

Report

Value Count

Value Count More than 5000

Object Name or Attribute Name	EID or Value Count
CN=SOD role 2.CN=Level30.CN=RoleDefs.CN=RoleConfig.CN=AppConfig.CN=User Application Driver.CN=driverset1.O=system.T=VM_DEMOHB_TREE.	00039E98
Equivalent To Me	109595
CN=Casper-It02.CN=Level20.CN=RoleDefs.CN=RoleConfig.CN=AppConfig.CN=User Application Driver.CN=driverset1.O=system.T=VM_DEMOHB_TREE.	00039EFA
Equivalent To Me	27654
CN=BuschkePer03.CN=Level10.CN=RoleDefs.CN=RoleConfig.CN=AppConfig.CN=User Application Driver.CN=driverset1.O=system.T=VM_DEMOHB_TREE.	00047CFB
Equivalent To Me	27654
CN=BuschkePer04.CN=Level10.CN=RoleDefs.CN=RoleConfig.CN=AppConfig.CN=User Application Driver.CN=driverset1.O=system.T=VM_DEMOHB_TREE.	00047CFC
Equivalent To Me	27653
CN=bartgrp.OU=groups.O=data.T=VM_DEMOHB_TREE.	00039EB9
Member	20011
Equivalent To Me	20011
CN=bigdyngroup.OU=groups.O=data.T=VM_DEMOHB_TREE.	00048193
nrfDynamicMember	27653
CN=bigdyngroup2.OU=groups.O=data.T=VM_DEMOHB_TREE.	000481A6
nrfDynamicMember	27653
CN=bigdyngroup3.OU=groups.O=data.T=VM_DEMOHB_TREE.	000481A9
nrfDynamicMember	27653

iMonitor -> Reports -> Value count

Updating attributes with large value counts is very resource intensive. Updating indexes takes a long time, specially for large values

Latest versions of RRSD driver limit how long DirXML-EntitlementResult values are stored

It is best to avoid a high value count condition. Plan resource/group assignment accordingly.

Tuning eDirectory Background processes

iMonitor -> Agent Configuration -> Background Process Settings

Enable Async Outbound sync to split the process that builds the Change Cache from the one that does the actual sync.

Make sure ARC is enabled

Set Delay settings to a value between 10 and 20 ms. A value below 5 ms is not recommended.

Adjusting synchronization strategies

iMonitor -> Agent Configuration -> Agent Synchronization

Consider reducing the number of threads, to reduce contention

If one server updates more, set that server with a higher amount of threads

Priority sync can help for time sensitive data, like passwords, but it shouldn't be abused.

Avoiding resource starvation

Thread pool

Instead of creating and destroying threads, they are placed in a pool.

Two types of queues:
- Ready queue
- Waiting queue

Use `ndstrace -c threads` to gather:

- The total number of threads that are spawned, terminated, and idle.
- The total number of worker threads currently and the peak number of worker threads.
- The number of tasks and peak number of tasks in the Ready queue.
- The minimum, maximum and average number of microseconds spent in the Ready queue.
- The current and maximum number of tasks in the Waiting queue.

opentext™

Thread pool parameters:

n4u.server.max-threads: Maximum number of threads that can be available in the pool.

n4u.server.idle-threads: Maximum number of idle threads that can be available in the pool.

n4u.server.start-threads: Number of threads started.

Use ndsconfig get/set or modify nds.conf file directly

Thread pool

Default max threads is 256

Maximum with ndsconfig set is 512

If modified on nds.conf, there is no actual maximum

Consider that each thread spawned consumes a memory footprint

opentext™

Ephemeral ports

Default Linux range is from 32768 to 61000

- *To change it: echo 15000 65000 > /proc/sys/net/ipv4/ip_local_port_range*

You can use netstat –na to check how many ports are in use. Check for large amount of ports in TIME_WAIT status

Parameter NDSD_AGENT_CONTEXT_OPTIMIZATION=true should help reduce the need for this

What works even better is to allow the system to reuse threads in Time Wait status:

- Dynamic: echo 1 > */proc/sys/net/ipv4/tcp_tw_reuse*
- Permanent: *add net.ipv4.tcp_tw_reuse = 1 in file sysctl.conf*

eDirectory contexts

iMonitor -> Connections -> Contexts

Different processes inside eDirectory spawn eDirectory contexts

The table is limited by the NCP buffer size, which limits it to 32K

If the table is full, the server stops working

This is often a symptom of a software bug.

Transaction ID

Before it's exhausted in ndsd.log and with the flag +RECM:

FATAL: DB : WARNING: the last transaction ID 0xXXXXXXXX

Once exhausted, the error displayed is:

The current transaction ID is 0xXXXXXXXX. Run local database repair with rebuild database option enabled
Transaction ID has exceeded the allowed limit of 0xFFFFE000. Run local database repair

More information in TID 7002658

https://support.microfocus.com/kb/doc.php?id=7002658

Cleaning the oidpInstanceData attribute

oidpInstanceData stores session information for Identity Applications.

If a session is not closed cleanly (close browser without logout), it grows over time. If it grows too large, logins will fail with error -649: Insufficient buffer

New property in the UserApp driver will perform automatic cleanup:

- https://www.netiq.com/documentation/identity-manager-48/releasenotes_idm484/data/new-features-and-enhancements.html#t4dk24tdr44u

More information

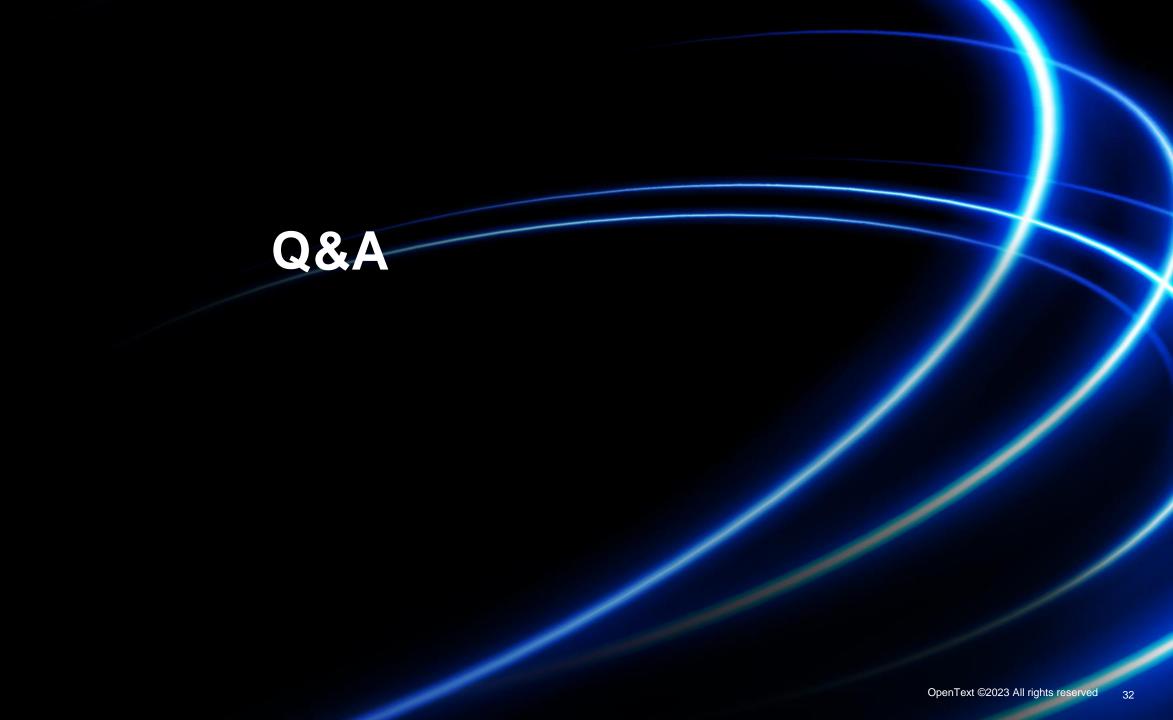Extended version of this presentation:

- https://www.thettp.org/edir-webinar

Tuning guide:

- https://www.netiq.com/documentation/edirectory-92/edir_tuning/data/bookinfo.html

History of issues fixed:

- eDir 9: https://support.microfocus.com/kb/doc.php?id=7016794
- eDir 8.8.x: https://support.microfocus.com/kb/doc.php?id=3426981

Q&A

opentext™

Thank you

twitter.com/opentext

linkedin.com/company/opentext

opentext.com

opentext™