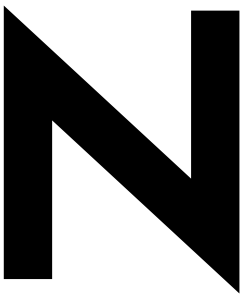
Novell® eDirectory™ 8.7:

Performance Tuning for Linux* and UNIX*

www.novell.com

GUIDE



Novell®

ABOUT THIS GUIDE

This guide describes how to tune Novell® eDirectory™ on Linux* and UNIX* platforms to improve its performance, providing both eDirectory tuneables and OS specific tuneables to optimize performance when deploying eDirectory.

This guide is intended for network administrators.

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, TM, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as UNIX, should use forward slashes as required by your software.

Table of Contents

Novell eDirectory 8.7:
Performance Tuning
for Linux and UNIX

Introduction

Novell® eDirectory™ 8.7 is a standards-compliant, cross-platform, highly scalable, fault-tolerant and high-performance directory service.

Performance tuning of any software application is a complex job. It requires an understanding of various components and subsystems of the software, knowledge of operating system and other system resources like file system, memory, storage media and bandwidth.

Prerequisites

- Ensure that you are running Novell eDirectory 8.7 with the latest patches and updates.

  To check the version of eDirectory, enter the following command:

  ndsd --version

- Ensure that your OS is updated with the latest patch levels.

  Linux*:

  — Red Hat* Linux 7.2 or 7.3

  Ensure that the latest glibc patches are applied from Red Hat Errata (**http://www.redhat.com/apps/support/errata**) on Red Hat systems.

  Solaris*:

  — Solaris 7 on Sun* SPARC (with patch 106327-13 or later for 32-bit systems)

  — Solaris 7 on Sun SPARC (with patch 106300-07 or later for 64-bit systems)

  — Solaris 8 on Sun SPARC (with patch 108827-20 or later)

  AIX*:

  — AIX 4.3.3 with Maintenance Level 10, JVM* 1.3.1, and the latest AIX V5.0 Runtime Libraries (**http://www-1.ibm.com/support/docview.wss?uid=swg24001173**)

  — AIX 5L with Maintenance Level 2, JVM 1.3.1, and the latest AIX V6.0 Runtime Libraries (**http://www-1.ibm.com/support/docview.wss?uid=swg24001467**)

For more information about system requirements, refer to the *eDirectory Administration Guide* at: **http://www.novell.com/documentation/lg/edir87/index.html**.

TUNING THE CACHE SUBSYSTEM

Novell eDirectory uses a state-of-the-art cache subsystem to reduce disk access and deliver better performance. Adequate amount of cache memory is critical for the performance of eDirectory servers.

Hence, cache sizing is one of the most important factors affecting the overall performance of eDirectory.

This section describes how eDirectory uses the cache internally. Understanding this is important for tuning the cache for a specific deployment. Novell eDirectory 8.5 and later versions provide a *block cache* and an *entry cache* to boost certain areas of eDirectory performance.

There is a certain amount of redundancy between the two caches, but each cache is designed to boost performance for different types of operations.

- The block cache holds the physical disk blocks to minimize frequent access to the disk; whereas, the entry cache contains logical entries from the directory.
- Generally, block cache is more useful for update operations, and entry cache is more useful for operations that tend to browse or traverse the directory tree by reading through entries like name-resolution operations.
- Both block cache and entry cache are useful in boosting query performance—block cache helps searching indexes, and entry cache helps retrieving the entries that are referenced from an index.
- With both an entry cache and a block cache, the total memory available for caching is effectively split between the two caches. By default, eDirectory splits the cache equally by giving 50% of available cache to each cache.

eDirectory normally creates logical entries in memory by getting the data from the block cache. The entry cache reduces the processing time required to do this. This time saving can be significant in some applications.

The rule of the thumb here is that the larger the number of items (blocks and entries) you cache, the better the overall performance. The ideal is to cache the entire database in both the entry cache and the block cache. This would not be possible for extremely large databases.

The amount of memory required to cache the entire database in the block cache is roughly the size of the database on disk, which is a 1:1 ratio. On the other hand, the amount of memory needed to cache the entire database in the entry cache is roughly two to four times the database size on disk, that is, a 1:2 or 1:4 ratio.

NOTE: *This is only a very broad and very general rule of thumb and could vary significantly based on your deployment. Set the cache between 100MB to 2.5GB. You would not need more than three to four times the size of the DIB. For large DIBs, limit the cache to 2GB.*

In order to address the wide range of needs and different deployments and configurations possible, the mechanisms for regulating cache memory consumption in eDirectory 8.7 have been made to be intelligent, dynamic, and automated.

eDirectory provides the following two types of methods to control cache memory consumption:

- "Dynamic Sizing of the Cache"
- "Hard Memory Limit"

Both of these methods are mutually exclusive. You can use either one at any time, but the last one used always replaces any earlier settings.

Dynamic Sizing of the Cache

This method dynamically adjusts the cache limit to regulate memory consumption. A dynamically adjusting limit causes eDirectory to periodically adjust its memory consumption in response to the flow of memory consumption by other processes. You need to specify the limit as a percentage of the available physical memory. Using this percentage, eDirectory recalculates a new memory limit at fixed intervals. While this works well in typical user scenarios, because of large differences in memory usage patterns and memory allocators on UNIX platforms, this mechanism is not recommended for optimal performance of eDirectory on UNIX platforms.

UNIX gives the impression of having less available memory than other operating systems because of the following reasons:

- UNIX uses the free memory for internal caching of file system blocks, frequently run programs, libraries, etc.
- Libraries in UNIX normally do not return the freed memory back to the OS.

NOTE: *Physical memory excludes machine swap space. This convention is followed throughout this document.*

Hard Memory Limit

This is the second method provided for regulating memory consumption. The Hard Memory Limit was present in earlier versions of eDirectory also. Once this is set, a limit is not changed until you either set a different hard limit or dynamically adjust the limit.

You are allowed to specify a hard memory limit in one of three ways:

- Method 1: As a fixed number of bytes
- Method 2: As a percentage of physical memory
- Method 3: As a percentage of available physical memory

A hard limit specification using the second and third methods is always translated to a fixed number of bytes. Thus, for method two, the number of bytes will be the percentage of physical memory that is detected when eDirectory is started. For method three, the number of bytes will be the percentage of available physical memory that is detected when eDirectory queries the OS at regular intervals of time.

The default mechanism for regulating memory consumption is as follows: if the server contains a replica, eDirectory uses a dynamically adjusting limit of 51% of available memory, with a minimum of 8 megabytes, leaving a minimum of 24 megabytes for other applications. Otherwise, eDirectory uses a hard limit setting of 16 megabytes, with 8 megabytes for block cache and 8 megabytes for entry cache.

Interaction of File System Buffer Cache

On UNIX platforms, the OS tries to cache file system blocks in its internal buffer cache. You must normally tune the OS to flush this internal buffer cache as fast as possible; even bypass it completely, if it is feasible. If this is not possible, do not specify more than 50-75% of the total physical memory for the eDirectory cache.

Dirty Cache

Novell eDirectory 8.7 introduced a new method for specifying the maximum dirty cache and the low dirty cache for the eDirectory cache. The purpose is to keep the amount of dirty cache at any given instant below a particular value. This value is configurable. This evens out the disk writing, instead of burdening the checkpoint thread in the forced mode, which will essentially write the whole cache to the disk, thereby creating an I/O bottleneck.

Refer to section "Setting the Maximum and Low Dirty Cache" on page 6 for more details.

Database Tuneable Parameter Settings

At startup, eDirectory looks for the database options file, _ndsdb.ini, in the directory the DIB files are stored in. This file is a simple text file that can be created or modified with any text editor. The syntax for controlling cache memory consumption is given in the sections below.

NOTE: *Although you may alter the _ndsdb.ini file at any time, the changes do not take effect in eDirectory until the server is restarted.*

Set the Hard Memory Limit

Add the following line to the _ndsdb.ini file to set the maximum amount of cache that eDirectory will consume (including both the entry cache and the block cache):

cache=<*cache bytes*>

Set a Dynamically Adjusting Limit

Add the following line to the _ndsdb.ini file.

cache=<*cache options*>

The *cache options* are described below. Multiple options may be specified, in any order, separated by commas:

CACHE OPTIONS	DESCRIPTION
DYN or HARD	Dynamically adjust the limit or hard limit. (For optimal performance use only hard on UNIX platforms). **NOTE**: *We recommend you to use HARD limits.*
%:<percentage>	Percentage of available or physical memory to use.
AVAIL or TOTAL	Percentage is for the available physical memory or total physical memory. This option is ignored for a dynamically adjusting limit, because a dynamically adjusting limit is always calculated based on the available physical memory.
MIN:<bytes>	Minimum number of bytes.
MAX:<bytes>	Maximum number of bytes.
LEAVE:<bytes>	Minimum number of bytes to leave for the OS and other applications.

Examples:

- Set cache to 75% of total memory, minimum of 16 megabytes as follows:

  cache=HARD,%:75,MIN:16000000

- Set cache to 65% of total memory, leaving at least 32 megabytes for the OS, minimum of 32 megabytes

  cache=HARD,%:65,MIN:32000000,LEAVE:32000000

Setting the Cache Adjust Interval and Cache Cleanup Interval

In addition to the cache setting for regulating memory consumption, eDirectory also provides settings to control the dynamic adjust interval, and the interval for cleaning up older versions of entries and blocks.

These settings are as follows:

cacheadjustinterval=<*seconds*>

cachecleanupinterval=<*seconds*>

Default is 15 seconds, if it is not set in the .ini file.

Setting the Cache Ratios

The following setting allows you to control the percentage split between the entry and block cache:

blockcachepercent=<*percent*>

Default is 50 percent, if it is not set in the .ini file.

Where, <*percent*> should be a value between 0 and 100 (inclusive).

A value of 70 means that 70 percent of cache memory would be used for block cache and 30 percent for entry cache.

We do not recommend you to set this percentage to 0.

Set blockcachepercent between 70% and 90% depending on the proportion of updates in the total operations.

Set it to 90% for operations like bulk create or delete.

Set to 50% if you do not expect too many update bursts.

Setting the Database Block Size

Novell eDirectory uses a default database block size of 4096 bytes. In FLAIM, *block* is a buffer used to aggregate disk images of an entry. It is used to model a portion of a single file on disk. Larger the block, more the number of entries in a block and longer is the time taken to search within a block and extract an entry. But a short block means more I/O calls to the OS.

You can configure the database block size as follows:

blocksize=<*4096 or 8192*>

Note that this parameter is effective only during the installation of eDirectory and has no effect once a database has been created. It is also important to understand that increasing the blocksize from 4096 may adversely impact the performance of other eDirectory operations (like search) even though update performance will improve. A block size of 4K gives better performance in all cases.

Setting the Maximum and Low Dirty Cache

The two parameters we recommend to set while bulkloading the database are *maxdirtycache* and

lowdirtycache. You need to set them in the _ndsdb.ini file.

eDirectory 8.7, by default sets the value of maxdirtycache to the unlimited value and the lowdirtycache value is set to zero.

You can specify the maxdirtycache and lowdirtycache as follows:

maxdirtycache=<*value*>

lowdirtycache=<*value*>

Modifying Database Cache Settings at Runtime

You can set the amount of cache to be used while eDirectory is running by using the ndstrace utility. In the ndstrace window, enter the following command:

set dstrace=!mb<*amount of RAM to use in bytes*>

To set a simple hard limit, enter the following command:

set dstrace=!mb<*cache options*>

To set a dynamically adjusting limit, see "Set a Dynamically Adjusting Limit" on page 5. The settings are effective only as long as the current instance of eDirectory is running.

DATABASE INDEXING

To improve the performance of LDAP searches, index the attributes on which a search is done. There are three types of indexes:

- Presence
- Value
- Substring

You can add an index using ConsoleOne®. Indexes can dramatically speed up the performance of applications based on the search expressions being used. Indexes need to be created judiciously, because, while indexes increase the search performance, each additional index adds to the update time for a new object; this is especially true for substring. The Novell eDirectory database is set up to select one optimal index per complex search, and then apply the other filter criteria to the results pulled from the index.

NOTE: *For massive bulkloading (millions of objects), we do not recommend you to disable the eDirectory server indexes while bulkloading the directory.*

Suspending the Indexes

When you are bulkloading a large amount of objects, you can suspend the substring indexes to speed up the bulk loading operation. You can enable them later after the loading is complete; eDirectory will automatically complete the indexing in the background. However, please note that the indexes will not be used for search operations until the background indexing is complete and the indexes are brought online.

Bulkloading users with passwords

Bulkloading users with passwords is much slower than adding users without passwords because eDirectory has to create a "RSA* Key Pair" for each password. This is a CPU-intensive cryptographic operation and cannot be bypassed because of the security needs of eDirectory. It is recommended that you first start a bulkload of the users without password and add the passwords in parallel.

TUNEABLES FOR BULKLOADING DATA

Novell Import Convert and Export (ICE) utility uses an optimized bulk update protocol called LBURP to upload data into the directory eDirectory. This protocol is significantly faster than uploading using a simple ldapmodify command.

NOTE: *For faster bulkloading, use the -C -n options with ICE.*

n4u.ldap.lburp.transize

ICE loads multiple objects in a single transaction to improve update performance. You can increase the performance of ICE bulkload even further by increasing the transaction size from the default of 25. The recommended range of this transaction size is 25 to 350 depending on the size of the whole transaction and system resources. However, note that an increase in the transaction size will increase the memory usage in eDirectory because all the records must be buffered in memory. If the system is running low on memory, this can cause a slowdown due to swapping. The transaction size can be modified by specifying the required value for the n4u.ldap.lburp.transize parameter in the /etc/nds.conf file.

The LBURP transaction size determines the number of records that will be sent from the ICE utility to the LDAP server in a single LBURP packet. However, even if a single error exists in the transaction, (including cases where the object to be added already exists in the directory), the LBURP optimization will be disabled and objects will be added to eDirectory individually for that transaction. In addition, the LBURP optimization currently works only for leaf objects; so the optimization is lost if the transaction contains both a container and it's subordinate objects. Therefore, we recommend you to add the containers first, using a separate LDIF file.

blockcachepercent

When you are bulkloading a large number of objects (e.g. greater than one million), you can set the blockcachepercent parameter to 90% or higher in the _ndsdb.ini file. Remember to change the parameter back to the original value after you complete the bulkload.

maxdirtycache and lowdirtycache

While loading a large number of objects, a burst in Disk I/O is observed, which slows down the bulkloading rate. To smoothen the disk I/O pattern we need to set the maxdirtycache and lowdirtycache to appropriate values.

NOTE: *Setting of maxdirtycache and lowdirtycache is useful only for bulkloading for less than 1.5 million objects. For higher values, there might be a performance degradation.*

Guidelines to Set the Value of maxdirtycache and lowdirtycache

Measure the random I/O write speed to the disk and set the maxdirtycache such that all modified buffers in a 3-minute interval can be flushed to the DIB volume in ten seconds (10000ms) or less and set lowdirtycache to about half this value.

For example, if the random write speed is 10ms per block (4KB), then set maxdirtycache to (10000ms*4K/10) or 4000KB. On most systems,

the maxdirtycache will be between 1 and 10MB. Fibre Channel SANs offer higher rates, so you can go up to 20MB.

If you are not sure, set it to 5MB and observe the max update response time during a burst of updates. Adjust this value upwards until the response time is acceptable.

TUNING A REPLICA RING FOR MINIMUM REPLICATION LATENCY

Novell eDirectory uses a slow, but sure convergence algorithm to replicate changes on a single replica server to its peers in a replication ring. A replica server can manage only a single DIB, but a DIB may contain replicas of multiple partitions.

eDirectory uses a batch update mechanism for replication. The period for which changes are accumulated in a replica server is adjustable from one second to a few hours, but defaults to 30 minutes.

Important changes like passwords will schedule a sync immediately. However, sync operations are handled by a background thread which would yield or postpone its operation if a request for a create, modify or delete operation is received.

Replication latencies can be minimized by partitioning a tree such that update operations are spread across multiple partitions and placing these volatile partitions on DIBs such that the peak update load on each DIB is minimized. For example, if a tree has three containers that are volatile, then isolate each container into a partition and place them in separate DIBs. Larger the peak update rate, smaller the ring, but a ring should be designed with at least two DIBs. If the entire server farm is front-ended by a load balancing switch, configure the switch to direct all requests to the primary servers and failover to the secondary.

TUNING eDIRECTORY THREADS

Novell eDirectory uses an internal pool of threads to service client requests and internal operations. This thread pool avoids the overhead of starting or stopping a new thread for every request. Maximum performance is achieved by using the minimum number of threads required to service the requests. eDirectory 8.7 automatically tries to use a lesser number of threads and starts or stops threads as needed. This delivers optimum performance in most cases. This may need some tuning under heavy client loads.

n4u.server.active-interval

The parameter n4u.server.active-interval controls when a new thread is started. A thread should be considered *busy* on another job if it does not return back to the thread pool within the time interval (in milliseconds) specified by the parameter. This parameter is scaled based on the number of processors available on the machine and can be increased to its maximum value (25000) to get the maximum performance.

n4u.server.idle-threads

This may be specified depending upon the average client load. The idea is to minimize the time required to produce new threads during normal client activity. The parameter specifies the minimum number of threads regardless of any activity.

n4u.server.start-threads

This specifies the number of threads that start when eDirectory starts. This also depends upon the average client load, in order to minimize the time required to produce new threads during normal client activity.

n4u.server.max-threads

The number of threads in eDirectory also influences the memory used by the process (in addition to the database cache). Each thread uses approximately 200 KB of memory during intensive search operations. As a general rule, assume that eDirectory will internally need about 16 threads for its internal operations. Add an additional thread for every 255 clients that need to be serviced simultaneously. Finally, add approximately 8 threads for each processor configured on the machine to service client search requests (the actual number will depend on the time taken for each request).

Number of eDirectory server threads = 16
(needed internally by eDirectory)
+ (Number of simultaneous clients service requests) / 255
+ (8 * number of processors)

Use this value to set the parameter n4u.server.max-threads. A value of 128 works well in most cases and will not require more tuning except when servicing a very large number of clients. The default value for this parameter is 64.

TUNING FILE SYSTEM

The choice of the file system can influence the bulk update performance significantly, though search performance is less affected because of aggressive caching in Novell eDirectory. Using Veritas* File System with a block size of 4KB (eDirectory default database block size) can give significantly improved performance. If you are installing over UFS, you may also set the blocksize parameter to 8192 in _ndsdb.ini.

As mentioned earlier, dynamic resizing of the eDirectory database cache does not inter-operate well with Solaris internal caching and the user level memory allocation algorithms. Therefore, we recommend that you always use a hard limit for the cache to get optimal performance.

TUNING OPERATING SYSTEM FOR eDIRECTORY

The operating system on which Novell eDirectory is installed plays a crucial part in its performance. This section gives general guidelines on tuning your OS and then gives specific tuneables based on your target platform.

Operating System Version

The version of the OS that you are running on may affect the performance of eDirectory significantly. In general, you must update your OS to the latest patch level and in some instances upgrade to a newer version of your OS to get optimal performance.

For more information, see "Prerequisites" on page 2.

Disk Performance

Update operations in eDirectory can be disk intensive. Spread out the I/O bandwidth over

multiple disks with RAID striping. You can have a stripe width of 16KB, 32KB, 64KB (based on your disk/controller) for maximum performance. The choice of the file system and the file system block size also affects performance.

Tuning Solaris for eDirectory

Note that some of these parameters may be superseded or modified for newer versions of Solaris. This information applies for Solaris 7 and Solaris 8.

Set the following system tuneables in the /etc/system file:

```
set priority_paging=1 (Not needed on Solaris 8
    onwards)
set maxphys=1048576
set ufs:ufs_LW=<1/128 of available memory>
set ufs:ufs_HW=<1/64 of available memory>
set tcp:tcp_conn_hash_size=8192
```

Ensure to backup your original file before making these changes and to reboot your system to get the parameters in effect.

- **priority_paging**. The priority paging algorithm allows the system to place a boundary around the file cache, so that file system I/O does not cause paging of applications. Setting priority_ paging value to 1 enables priority paging. You should not set the system variable priority_paging in the Solaris 8 operating environment, and you should remove the variable from the /etc/system file when systems are upgraded to the Solaris 8 operating environment.
- **maxphys**. Maximum size of physical I/O requests. It is specified when doing I/O to and from a UFS file system where large amounts of data (greater than 64 Kbytes) are being read or written at any one time.
- **ufs_HW**. The number of bytes outstanding on a single files barrier value. If the number of bytes outstanding is greater than this value and ufs_WRITES is set, then the write is deferred. The write is deferred by putting the thread issuing the write to sleep on a condition variable.
- **ufs_LW**. The barrier for the number of bytes outstanding on a single file below which the condition variable on which other sleeping processes is toggled. When a write completes and the number of bytes is less than ufs_LW, then the condition variable is toggled, which causes all threads waiting on the variable to awaken and try to issue their writes.

  ufs_LW and ufs_HW have meaning only if ufs_WRITES(set in /etc/system) is not equal to zero. ufs_HW and ufs_LW should be changed together to avoid needless churning when processes awake and find that they either cannot issue a write (when ufs_LWand ufs_HWare too close) or when they might have waited longer than necessary (when ufs_LWand ufs_HW are too far apart).
- **tcp_conn_hash_size**. Controls the hash table size in the TCP module for all TCP connections. If the system consistently has tens of thousands of TCP connections, increase the

value accordingly. With the default value, TCP performs well up to a few thousand active connections. Note that increasing the hash table size means more memory consumption so set an appropriate value to avoid wasting memory unnecessarily. The parameter can only be changed at boot time.

The number of connections to an eDirectory replica also influences the search performance. Many improvements were made to make eDirectory performance scalable with growing number of connections. However, many TCP parameters, like transmission and reception queue size and transmission and reception window size, influence the behavior of the networking subsystem in the operating system. These may affect eDirectory performance. For optimizing search performance, set the following networking tuneables using ndd:

```
ndd -set /dev/tcp tcp_conn_req_max_q 1024
ndd -set /dev/tcp tcp_close_wait_interval 60000
    (obsolete in Solaris 8 onwards, instead use
    tcp_time_wait_interval)
ndd -set /dev/tcp tcp_xmit_hiwat 32768
ndd -set /dev/tcp tcp_xmit_lowat 32768
ndd -set /dev/tcp tcp_slow_start_initial 2
```

NOTE: *Please note that these ndd settings will not survive a reboot. Please add them in a script that will be run at boot time. You may also need to set these parameters on your LDAP client machine to get the best results.*

- **tcp_conn_req_max_q**. The default maximum number of pending TCP connections for a TCP listener waiting to be accepted by accept. For applications such as Web servers that might receive several connection requests, the default value might be increased to match the incoming rate.
- **tcp_xmit_hiwat**. This is the default send window size in bytes.
- **tcp_close_wait_interval**. The time in milliseconds a TCP connection stays in TIME-WAIT state (tcp_close_wait_interval is obsolete in Solaris 8 onward; use tcp_time_wait_interval in Solaris 8). On a busy Web server, there can be too many TCP connections in TIME-WAIT state, consuming too much memory. In this situation, you can decrease the value for performance reasons. Do not set the value lower than 60 seconds.
- **tcp_slow_start_initial**. The maximum initial congestion window (cwnd) size in MSS of a TCP connection. If the initial cwnd size causes network congestion under special circumstances, decrease the value.

Tuning Linux for eDirectory

We strongly recommend that you upgrade to Linux kernel versions 2.4.9 or above for eDirectory. eDirectory performance is significantly better with 2.4.9 or above kernel versions, especially in large memory configurations.

There are no Virtual Memory tuneables for eDirectory on Linux. The page cache in Linux

kernel 2.4 gives excellent performance for database applications in comparison with 2.2 kernels. The default TCP/IP settings on Linux give satisfactory LDAP search performance and do not require further tuning.

If you use the ext2 file system storing the DIB files for eDirectory, the file system should be created with a block size of 4096 for optimum performance. You can do this by the following command before installing eDirectory:

```
mke2fs -b 4096 <device>
```

You can also disable updating access times by running the following command for all the DIB files:

```
chattr -A <filename>
```

The new reiserfs file system coming with the 2.4.1 and above kernels has not been extensively tested with eDirectory yet, so we cannot make any recommendation at this point in time.

As mentioned earlier, dynamic resizing of the eDirectory database cache does not inter-operate well with Linux internal caching and the glibc memory allocation algorithms. Therefore, we recommend that you always use a hard limit for the cache to get optimal performance.

Tuning AIX for eDirectory

Novell eDirectory 8.7 supports the AIX operating environments of AIX 4.3 and AIX 5.1. The AIX platform provides significant tools for optimizing performance on the AIX platforms from Virtual Memory and Network I/O to Disk I/O. Note that major changes occurred in the AIX environment between the AIX 4.3 and AIX 5 releases and tuning options in AIX 4.3 and AIX 5.1 are changing in AIX 5.2.

Tuning information for the various releases of the AIX operating environments are available at the following Web sites:

- **AIX 4.3**. **http://publib.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/toc.htm**
- **AIX 5.1**: **http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/prftungdtfrm.htm**
- **AIX 5.2**: **http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/prftungdtfrm.htm**

As mentioned earlier, we recommend using a minimum file system block size of 4096 bytes (or multiple thereof) which matches the default database block size. Dynamic resizing of the eDirectory database cache does not interoperate well with UNIX internal caching, therefore, we recommend you to always use a hard limit for the cache to get optimal performance.

Novell Product Training and Support Services

For more information about Novell's worldwide product training, certification programs, consulting and technical support services, please visit:
www.novell.com/ngage

For More Information

To access the online documentation for this and other Novell products, and to get updates, see:
www.novell.com/documentation

You may also call Novell at:

1 888 321 4272 US/Canada
1 801 861 4272 Worldwide
1 801 861 8473 Facsimile

Novell, Inc.
1800 South Novell Place
Provo, Utah 84606 USA

www.novell.com

Novell

462-001358-003