IBM Directory Server Version 4.1 Tuning Guide

IBM Directory Server Version 4.1 Tuning Guide

First Edition (April, 2002)

This edition applies to version 4, release 1, of The IBM® Directory and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

Welcome to the IBM Directory Tuning Guide. The purpose of this document is to provide performance tuning information for IBM Directory. The document is divided into sections dealing with IBM Directory, IBM Universal Database 2 (DB2®), operating system, and hardware tuning issues.

This document provides configuration data to help you improve performance. Tuning considerations for directory sizes ranging from a few thousand entries to millions of entries are given where applicable. Some advantages and disadvantages of different settings for the tuning parameters are also given. Some of these settings might affect resource usage, speed, and functionality.

For the most current and accurate tuning information, see the Web version of the Tuning Guide on the IBM Directory Web site:

http://www-4.ibm.com/software/network/directory/library

Who should read this guide

The target audience for this guide includes:

- System installation and deployment administrators
- Network system administrators
- Information Technology architects
- Application developers

Typeface conventions

This guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold	Command names and options, keywords, and other information that you must type as shown.
Italics	Variables and values you must provide appear in italics.
`Monospace`	Code examples, command lines, screen output, file names, programming keywords, message text or prompts addressed to the user, and text that the user must enter appear in monospace font.

Acronyms used in this document

- ACL – Access Control List
- DB2 – DB2 Universal Database™
- LDAP – Lightweight Directory Access Protocol
- SMP – Symmetric Multi-Processor

Chapter 1. IBM Directory tuning general overview

The IBM Directory is a Lightweight Directory Access Protocol (LDAP) directory that enables users to store and retrieve data for multiple purposes. The IBM Directory uses IBM Database 2 (DB2), a powerful and scalable database product, for its data storage facility. In the most optimal LDAP environments, directory data is fairly static and the access for LDAP "cached" data is repetitive. In more robust environments, where directory data is updated frequently and the access for "non-cached" data is random, the power and scale of DB2 is used to enhance performance.

IBM quick-start tuning

This section contains IBM's recommendations for basic tuning. Although it includes pointers to some of the more important and commonly used tuning setups, it is by no means a comprehensive list. We recommend that you read the entire document to learn about the tuning setups for your particular system.

Basic tuning recommendations:

- Allocate more than 60% of your physical memory to the DB2 buffer pools. See "Buffer pool tuning for IBM Directory version 4.1" on page 14 for more information.
- For AIX® operating systems on Symmetric Multi-Processor (SMP) machines, set MALLOCMULTIHEAP. See "Setting MALLOCMULTIHEAP" on page 19 for instructions on how to set MALLOCMULTIHEAP.
- Make sure all attributes used in searches are indexed. See "Indexes" on page 7 for instructions on how to define and verify indexes for IBM Directory.

Migration

See "Chapter 11. Migration" in the *IBM Directory Server 4.1 Installation and Configuration Guide for Multiplatforms* for information about migrating from an earlier version of SecureWay® Directory to IBM Directory 4.1

IBM Directory tuning

The IBM Directory allows users to set LDAP front end configuration variables that customize the IBM Directory for a specific environment in the `slapd32.conf` file. See "Setting front end configuration variables" on page 5 for more information.

The most important configuration variables for search performance are related to LDAP caches, which are fast storage buffers in memory used to store LDAP information such as queries, answers, and user authentication for future use. While LDAP caches are mostly useful for applications which frequently retrieve repeated "cached" information, they can greatly improve performance by avoiding calls to the database. The negative aspect of LDAP caches is the cache invalidation that occurs on update operations. In order to take advantage of the LDAP caches, it is useful to understand how they are used and maintained in detail. See "LDAP cache" on page 8 for more information.

As a general rule, you should define larger LDAP caches for the following cases:

- If there is no or low update activity and mostly cached searches

- If there is no or low update activity and enough memory to cache the entire directory

Specific DB2 tuning

The IBM Directory uses the IBM DB2 relational database as the data store and Structured Query Language (SQL) query retrieval mechanism. While LDAP caches LDAP queries, answers, and authentication information, DB2 has much more sophisticated and complex caching mechanisms which cache tables, indexes, and statements.

The DB2 data caches, which are called buffer pools, are important factors that can affect DB2 performance. Buffer pools, due to the removal of LONGVARCHAR columns, generally are more effective than the LDAP cache, especially in read/write environments. Each buffer pool is a data cache between the applications and the physical database files. If there are no buffer pools, then all database activity results in disk access. If the size of each buffer pool is too small, the buffer pool hit ratio will be low and the applications will wait for disk access activity to satisfy SQL queries. If one or more buffer pools are too large, memory on the server might be wasted. If the total amount of space used by all buffer pools is larger than the physical memory available on the server, then operating system paging (disk activity) will occur. See "DB2 buffer pool tuning" on page 14 for more information.

In general, increasing the DB2 buffer pool caches can be advantageous in the following cases:

- If there is high update activity
- If most queries are ″non-cached″ and there is not enough memory to cache the entire directory

DB2 has many other configuration parameters that can affect either the memory or disk resources. Since disk access is usually much slower than memory access, the key database performance tuning objective is to decrease the amount of disk activity. If you are able to eliminate input/output (I/O) wait time, the database requests are Central Processing Unit (CPU) bound and increasing performance typically would then require faster CPUs or multiple CPUs. For more information in detail see "Other DB2 configuration parameters" on page 15.

After initially loading a directory, or after a number of updates have been performed, it might be necessary to update database statistics and table organization for DB2 to perform optimally. See "Database optimization, statistics, reorganization check, and reorganization" on page 11 for more information.

Administrators must place the DB2 log on a physical disk drive separate from the data. While there might be some performance benefit to having the DB2 log and data on the same drive, data-integrity concerns require the separation. Use the following command to set the path to the DB2 log file directory:

```
UPDATE DATABASE CONFIGURATION FOR database_alias USING NEWLOGPATH path
```

Note: Be sure the database instance owner has write access to the specified path or the command fails.

Other DB2 tuning

There are numerous other DB2 methods and techniques that can be used to extend scalability and possibly improve performance. More detailed information on DB2 can be found at:

http://www.ibm.com/software/data/db2

Generic LDAP application tips

The following are some generic tips that can help improve performance:

- Perform searches on indexed attributes only. See "Indexes" on page 7 for instructions on how to define and verify indexes for IBM Directory.
- Open a connection only once and reuse it for many operations if possible.
- Minimize the number of searches by retrieving multiple attribute values at one time.
- Retrieve only the attributes you need. Do not use ALL by default. For example, when you search for the groups a user belongs to, ask for just the Distinguished Names (DNs), and not the entire group.
- Minimize updates (add, modify, modrdn, delete) when possible.

Chapter 2. IBM Directory tuning

This chapter discusses the following performance tuning tasks for the IBM Directory:

- Setting front end configuration variables
- Setting back end configuration variables
- Indexes
- Update performance and SMP systems
- LDAP cache

Setting front end configuration variables

There are severable tunable LDAP front end configuration variables that can affect performance in the `slapd32.conf` file.

To set front end configuration variables, add this line:

```
ibm-slapdSetEnv: variable_name= value
```

to the following section:

```
dn: cn=Front End,cn=Configuration
objectclass: top
objectclass: ibm-slapdFrontEnd
```

in the `slapd32.conf` file.

For example, to set the RDBM cache size to 100,000, you would add the following entry to `slapd32.conf`:

```
dn: cn=Front End,cn=Configuration
objectclass: top
objectclass: ibm-slapdFrontEnd
ibm-slapdSetEnv: RDBM_CACHE_SIZE=100000
```

Possible variables

Below are some of the front end configuration variables you might want to set.

Note: Starting with this release, the LDAP server is always enabled for concurrent read/write. Setting the LDAP_CONCURRENTRW variable is no longer required.

ACLCACHE=YES|NO
Determines whether or not there will be an Access Control List (ACL) cache on the server.
By default this variable is set to YES.

ACLCACHESIZE=<integer>
Specifies the maximum number of entries kept in the ACL cache.
By default the size is 25000.

DB2CP=<integer>
Specifies the Code Page of the directory database. ″1208″ is the code page for UTF-8 databases.

By default this variable is set to 1208.

RDBM_CACHE_SIZE=<integer>
Specifies the maximum number of entries to keep in the Entry cache. By default the size is 25000.

RDBM_FCACHE_SIZE=<integer>
Specifies the maximum number of entries to keep in the Search Filter cache. By default the size is 25000.

RDBM_CACHE_BYPASS_LIMIT=<integer>
Search filters that match more than this number of entries will not be added to the Search Filter cache. Because the list of entry IDs that matched the filter are included in this cache, this setting helps to limit memory use. A value of 0 indicates no limit. The default value is 100

RDBM_ENTRY_CACHE_BYPASS=[any value]
When set to any value, the RDBM_CACHE_BYPASS_LIMIT applies to the entry cache in addition to the filter cache. By default, the RDBM_CACHE_BYPASS_LIMIT does not apply to the entry cache.

Setting back end configuration variables

Back end configuration variables are set during DB2 configuration. To set back end configuration variables, add this line:

```
ibm-slapd <variable>:<value>
```

to the following section:

```
dn:cn=Directory,cn=RDBM Backends,cn=IBM SecureWay,cn=Schemas,cn=Configuration
cn:Directory
objectclass:top
objectclass:ibm-slapdRDBMBackend
```

in the `slapd32.conf` file.

The following table contains some examples of back end configuration variables:

Table 1. Back end configuration variables

Back end configuration variable	Default value
ibm-slapdDbInstance	ldapdb2
ibm-slapdDbName	ldapdb2
ibm-slapdDbUserId	ldapdb2

LDAP connections to DB2

The LDAP server maintains a certain number of connections to the DB2 servers. This number is controlled by the **ibm-slapdDbConnections** parameter in the `slapd32.conf` file. By increasing the number of DB2 connections, LDAP can increase its level of concurrency and can improve throughput performance. You can specify between 5 and 50 connections. The default setting for **ibm-slapdDbConnections** is 15.

Indexes

It is very important to index all attributes used in searches. The following DB2 commands can be used to verify that a particular index is defined. In the following example, the index being checked is **principalName**:

```
db2 connect to ldapdb2
db2 list tables for all | grep -i principalName
db2 describe indexes for table ldapdb2.principalName
```

If the second command fails or the last command does not return three entries, the index is not properly defined. The last command should return the following results:

```
IndexSchema      Index Name                  Unique Rule          Number of Columns
 -------------  -------------------          ----------          -------------
LDAPDB2         PRINCIPALNAMEI               D                    1
LDAPDB2         PRINCIPALNAME                D                    2
LDAPDB2         RPRINCIPALNAME               D                    2

              3 record (s) selected.
```

To have IBM Directory create an index for an attribute the next time IBM Directory is started, do one of the following:

- From the Database Management Tool:
  1. Click **Schema>Attributes**.
  2. Click **Edit Attribute**.
  3. On the **IBM Extensions** tab, select the **EQUALITY** checkbox under **Indexes**.
- Edit either `/etc/ldapschema/V3.ibm.at` or `/etc/ldapschema/V3.user.at`. Find the line containing the attribute of interest that looks similar to the following:

  ```
  ( 1.3.18.0.2.4.318 DBNAME ( 'principalName' 'principalName' )
   LENGTH 256 EQUALITY ORDERING SUBSTR APPROX )
  ```

  Add the word ″EQUALITY″ to the list as shown in the example.

  Note: If you choose this option, you must stop and restart the server for the index to be created.
- Issue the following command:

  ```
  ldapmodify -f /ldap/etc/addindex.ldif
  ```

  The ldif file should look like this:

  ```
  dn: cn=schema
  changetype: modify
  replace: attributetypes
  attributetypes: ( 1.3.18.0.2.4.318 NAME ( 'principalName'  'principal' ) DESC
  'A naming attribute that may be used to identify eUser object entries.' EQUALITY
   1.3.6.1.4.1.1466.109.114.2 ORDERING 2.5.13.3 SUBSTR 2.5.13.4 SYNTAX
   1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )-
  replace: ibmattributetypes
  ibmattributetypes: ( 1.3.18.0.2.4.318 DBNAME( 'principalName'  'principalName' )
  ACCESS-CLASS normal LENGTH 256 EQUALITY ORDERING SUBSTR APPROX )
  ```

Performance on SMP systems

Updates to the LDAP master server are not serialized in IBM Directory 4.1. Non-serialized updates can improve the performance of updates and searches on SMP machines.

LDAP cache

The LDAP cache is highly efficient in terms of size and speed. An LDAP search that accesses the LDAP cache generally is dramatically faster than one that requires a connection to DB2, even if the information is cached in DB2.

The disadvantage of using the LDAP cache is the cache invalidation that occurs on update operations. The DB2 cache is not as sensitive to update operations.

The LDAP cache has three components: a filter cache, an entry cache, and an ACL cache. The filter cache consists of actual queries on the requested attribute filters and resulting entry identifiers that matched. On an update operation, all filter cache entries are invalidated.

The entry cache contains the actual entry information for all entry IDs that have been cached.

The ACL cache caches ACL information for individual entries. Even though all filter cache entries are invalidated, the entry cache and ACL cache remain and can result in improved performance. Accessing the LDAP entry cache generally is faster than going to DB2 for the entire search, but it is not typically as fast as accessing the LDAP filter cache and avoiding DB2 altogether.

Following are some LDAP cache recommendations:

- If the available memory and number of entries make it possible to cache all entries, and updates are infrequent, you should define the cache large enough to cache all entries.
- If a subset of entries is significantly more active than the entire set of entries, and there are infrequent updates, you should define the cache large enough to cache the subset of entries.

In most cases, experimentation between LDAP cache size and DB2 cache size is necessary. To allow for experimentation it is best to install large amounts of physical memory in the LDAP servers. Several gigabytes of RAM is not unreasonable when the directory contains millions of entries.

Setting the LDAP cache

By default the Entry and ACL cache size (Entry, Filter, and ACL) is 25,000 entries. An administrator can change workload values in `slapd32.conf`. There is no optimal cache size for all LDAP servers; this must be determined on an individual basis. The LDAP monitor tool is a good tool to help administrators determine the appropriate cache sizes. The monitor search returns the size of the entry and filter cache as well as the number of cache hits and misses.

Keep in mind that in order for an entry to be added to the cache it must first be a cache miss. Analysis of this information over a period of time can help determine if the server needs a larger or smaller cache.

To prevent large uncommon searches from overwriting useful cache entries you might want to set the RDBM_CACHE_BYPASS_LIMIT and RDBM_ENTRY_CACHE_BYPASS environment variables. Queries that match a number of entries greater than the value of RDBM_CACHE_BYPASS_LIMIT will not be entered into the filter cache. By default the value is 100. When the RDBM_ENTRY_CACHE_BYPASS variable is set, the cache bypass limit applies to

the entry cache in addition to the filter cache. See “Setting front end configuration variables” on page 5 for more information about setting front end configuration variables.

Estimating LDAP cache size

The following procedure shows you how to estimate LDAP cache size:

Note: Beginning in SecureWay Directory 3.2.2, the amount of memory required to estimate LDAP cache size increased significantly from the amount required in SecureWay Directory 3.2.1. If you are migrating from SecureWay Directory 3.2.1 to IBM Directory 4.1, and need to preserve your memory resources, we recommend that you reduce the RDBM_CACHE_SIZE value by half. For example, if you had RDBM_CACHE_SIZE=5000 in SecureWay 3.2.1, you should set it to RDBM_CACHE_SIZE=2500 in IBM Directory 4.1.

1. Set the following front end configuration variables in the `slapd32.conf` file:

   ```
   RDBM_CACHE_BYPASS_LIMIT=0
   RDBM_CACHE_SIZE=150000
   RDBM_FCACHE_SIZE=1000
   RDBM_ACLCACHE_SIZE=150000
   ```

2. Start or restart the slapd server.
3. Execute the command to monitor LDAP performance and ascertain the number of entry cache misses. This will be the value for the **entry_cache_miss** attribute. See “Monitoring Performance” on page 23 for more information about the **entry_cache_miss** attribute. After starting the server, this value should always be zero.
4. Execute some LDAP search queries (no updates) which represent the search workload for the server. This will cause the number of cache misses to grow. Continue until the number of cache misses approaches 10,000 while running some scripts to continuously monitor the server helps. Assume the value of **entry_cache_miss** equals **m1**.
5. Find the process size. On Unix operating systems, you can use the ps command; on a Windows NT operating system, use the task manager or performance monitor. Assume the process size is **s1** KB.
6. Execute more LDAP queries until the number of cache misses grows to 50000 or more. Assume the value of entry_cache_miss equals **m2** at this point.
7. Find the process size again. Assume this process size is **s2** KB.
8. Estimated memory requirement = (s2-s1)/(m2-m1) KB/entry.

Chapter 3. DB2 tuning

IBM Directory uses DB2 to store directory data. Tuning DB2 can improve LDAP performance.

The following types of tuning are discussed:

- Database optimization, statistics, reorganization check, and reorganization
- DB2 buffer pool tuning
- Other DB2 configuration parameters
- Backup and restore

Notes:

1. In all DB2 command examples it is assumed that **ibm-slapdDbInstance** and **ibm-slapdDbName** are *ldapdb2*. It is also assumed that the user is logged in as **ibm-slapdDbUserId**. If logged in as the root user on a UNIX® operating system, it is possible to switch to the ldapdb2 user as follows:

   ```
   su - ldapdb2
   ```

   To log on as the database administrator on a Windows® operating system, run the following command:

   ```
   runas /user:ldapdb2 db2cm
   ```

   where *ldapdb2* is the defined user of the LDAP database.

   See "Setting back end configuration variables" on page 6 for more information.
2. If you have any trouble running the DB2 commands, check to ensure the following:
   - The ID trying to run the DB2 commands is a user in the dbsysadm group (UNIX operating systems) or a member of the Administrator group (Windows NT operating system.) Only users listed as database administrators can execute the DB2 commands. This includes the DB2 instance owner (the default is ldapdb2) and root.
   - DB2 environment variables have been established by running **db2profile** (if not, the **db2 get** and **db2 update** commands will not work). The script file **db2profile** is located in the sqllib subdirectory under the instance owner's home directory. If you need to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.)
   - For additional stability and performance enhancements, upgrade to the latest version of DB2.

Database optimization, statistics, reorganization check, and reorganization

The IBM Directory **Optimize** button uses DB2 ″runstats″ to update statistical information used by the query optimizer for all the LDAP tables. In many cases, performance can be improved significantly by running statistics first. This is especially true after a large amount of data has been loaded. In IBM Directory 4.1 , the command line equivalent of the **Optimize** button is the following command for all LDAP tables:

```
DB2 RUNSTATS ON TABLE table-name AND  DETAILED INDEXES ALL SHRLEVEL REFERENCE
```

Run the following commands for more detailed lists of runstats that improve performance:

```
DB2 RUNSTATS ON TABLE table-name WITH DISTRIBUTION AND DETAILED INDEXES ALL SHRLEVEL
REFERENCE
```

```
DB2 RUNSTATS ON TABLE ldapdb2.objectclass WITH DISTRIBUTION AND DETAILED INDEXES
ALL SHRLEVEL REFERENCE
```

Another important and often overlooked DB2 tuning command is **reorgchk**. The **reorgchk** command can improve both search and update operation performance. In addition to doing a "runstats", it also provides an indication of what results if a table gets reorganized.

After a number of updates have been performed against DB2, table indexes become sub-optimal and performance can degrade dramatically. This situation can be corrected by performing a DB2 **reorgchk** as follows:

```
db2 connect to ldapdb2
db2 reorgchk update statistics on table all
```

reorgchk, as shown above, does two things. It updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables.

reorgchk needs to be run periodically. For example, **reorgchk** needs to be run after a large number of updates have been performed. Note that LDAP tools such as **ldapadd**, **ldif2db** and **bulkload** can potentially do large numbers of updates that require a **reorgchk**. The performance of the database should be monitored and a **reorgchk** performed when performance starts to degrade. See "Monitoring Performance" on page 23 for more information.

reorgchk must be performed on all LDAP replicas, since each uses a separate database. The LDAP replication process does not include the propagation of database optimizations.

In general, reorganizing a table takes more time than running statistics. Therefore, performance might be improved significantly by running statistics first.

Because LDAP caches prepared DB2 statements, you must stop and restart slapd in order for DB2 changes to take effect.

Database table organization

The organization of the data in DB2 can be tuned. Tuning organizes the data on disk in a sorted order. Sorting the data on disk is beneficial only when accesses occur in a sorted order, which is not typically the case. For this reason, organizing the table data on disk typically yields little change in performance.

The first step in the reorganization procedure is to perform a **reorgchk db2** command. The command is as follows:

```
db2 reorgchk update statistics on table all >reorgchk.out
```

The output of this command is routed to a file named `reorgchk.out` in the previous example. This makes it easy to view the results after issuing the command.

This command gives organizational information about the database. The next step is an iterative process of finding the tables and indexes that need reorganizing and attempting to reorganize them. This can take a long time. The time it takes to perform the reorganization and the **reorgchk** increases as the DB2 database size increases.

After reorganizing a group of tables or indexes, a new **reorgchk** must be run to generate new statistics. The output from **reorgchk** can then be used to determine whether the reorganization worked and whether it introduced other tables and indexes that need reorganizing.

The following discussion explains the commands and guidelines for identifying and reorganizing the tables and indexes in this iterative process:

The **reorgchk** update statistics has two sections; the first section is the table information and the second section is the indexes. Use this command to reorganize the tables with an asterisk in the last column:

```
db2 reorg table <table name>
```

where <*table name*> is the name of the table to be reorganized, for example. LDAPDB2.LDAP_ENTRY.

Generally speaking, since most data in LDAP is accessed by index, reorganizing tables is usually not as beneficial as reorganizing indexes.

Use this command to reorganize the indexes with an asterisk in the last column:

```
db2 reorg table <table name> index <index name>
```

where <*table name*> is the name of the table, for example, LDAPDB2.LDAP_ENTRY.

And where <*index name*> is the name of the index, for example, SYSIBM.SQL000414155358130.

Here are some guidelines for performing a reorganization:

- If the number on the column that has an asterisk is close to the recommended value described in the header of each section and one reorganization attempt has already been done, it is probably okay to skip a reorganization on that table or index.
- In the table LDAPDB2.LDAP_ENTRY there exists a LDAP_ENTRY_TRUNC index and a SYSIBM.SQL index. Preference should be given to SYSIBM.SQL index if attempts to reorganize them seem to alternate between one or the other needing reorganization.
- Reorganize all the attributes that you want to use in searches. In most cases you will want to reorganize to the forward index, but in cases with searches beginning with '*', reorganize to the reverse index.

  For example:

```
Table: LDAPDB2.SECUUID

LDAPDB2 RSECUUID <— This is a reverse index

LDAPDB2 SECUUID <— This is a forward index

LDAPDB2 SECUUIDI <— This is an update index
```

DB2 buffer pool tuning

Along with the DB2 optimization tools (for example, **reorgchk**), DB2 buffer pool is one of the most significant DB2 performance tunings. Unlike the optimization tools, which need to be run periodically when significant changes have occurred to the data, buffer pool tuning typically only needs to be done once.

A DB2 buffer pool is a data cache between LDAP and the physical DB2 database files for both tables and indexes. If there are no buffer pool(s), then all database activity results in disk access. If the size of each buffer pool is too small, LDAP has to wait for DB2 disk activity to satisfy DB2 SQL requests. If one or more buffer pools are too large, memory on the LDAP server might be wasted. If the total amount of space used by all buffer pools is larger than physical memory available on the server, operating system paging (disk activity) will occur.

Although it is not possible to cache an entire table containing millions of rows, it is possible that the indexes for an entire table can be cached. Cached indexes can provide a significant boost to performance, because they can make locating the data on disk very fast.

Buffer pool tuning for IBM Directory version 4.1

In IBM Directory 4.1, the LDAP directory database is created with an additional tablespace (LDAPSPACE) and buffer pool (LDAPBP) using a 32K page size. Since IBMDEFAULTBP uses the default 4K page size, and the LDAPBP uses the 32K page size, it is no longer possible to use the single database configuration parameter for BUFFPAGE. If you have created a new database using IBM Directory version 4.1, you need to set each buffer pool size separately using the alter buffer pool commands.

The following examples shows two buffer pools being set to a total size of 1.2 GB:

```
db2 alter bufferpool ibmdefaultbp size 7900
db2 alter bufferpool ldapbp size 3300
db2 force applications all
db2stop
db2start
```

As a general guideline, a 3 to 1 ratio between memory allocated to the IBMDEFAULTBP (4K pages) and LDAPBP (32K pages) is good for performance. By default, the IBMDEFAULTBP is created with a size of 29500 (4K) pages. By default, the LDAPBP buffer pool is created with a size of 1230 (32K) pages. On an LDAP Server with minimal memory configuration, this allocates roughly 60% of physical memory to the DB2 buffer pools.

To get the current DB2 buffer pool sizes, run the following commands:

```
db2 connect to ldapdb2
db2 "select bpname,npages,pagesize from SYSIBM.SYSbufferpools"
```

The following example output shows the default settings for the example above:

```
BPNAME                 NPAGES        PAGESIZE
-----------------  -----------  -----------
IBMDEFAULTBP             29500         4096
LDAPBP                    1230        32768

  2 record(s) selected.
```

Attention: If you are using a machine that does not meet the minimum memory system requirement, or you have a machine with minimal memory configuration and a read-only directory server, you need to reduce the buffer pool size by issuing the following commands:

```
db2 alter bufferpool ibmdefaultbp size 9800
db2 alter bufferpool ldapdb size 400
db2 force applications all
db2stop
db2start
```

Guidelines for when to use LDAP Cache or buffer pool cache

Cached tables can provide performance benefits, but the LDAP cache is generally more efficient as a means of caching LDAP searches. On the other hand, parts of the LDAP cache get invalidated on updates and must be reloaded before performance benefits return. Some experimentation between the two caching schemes is probably appropriate.

If there are updates interspersed with authentications and searches, it might be best to allocate enough available physical memory in DB2 cache to hold the indexes and allocate the remaining available physical memory to LDAP cache.

Other DB2 configuration parameters

Performance benefits can come from setting other DB2 configuration parameters. The current setting of parameters can be obtained by issuing the following command:

```
db2 get database configuration for ldapdb2
```

This command returns the settings of other DB2 configuration parameters as well.

The following command also shows the DB2 configuration parameters for the entire ldapdb2 instance:

```
db2 get database manager configuration
```

To set the DB2 configuration parameters use the following syntax:

```
db2 update database configuration for ldapdb2 using \
<parm name> <parm value>
db2stop
db2start
```

where <*parm name*> is the parameter to change and <*parm value*> is the value it is to be assigned.

Changes to DB2 configuration parameters do not take effect until the database is restarted with **db2stop** and **db2start**.

The following are some DB2 parameters that you might want to modify to optimize performance:

- APPLHEAPSZ
- PCKCACHESZ
- SORTHEAP
- LOGFILSIZ
- DBHEAP
- APP_CTL_HEAP_SZ

- LOCKLIST

Backup and restore

Typically, the fastest way to backup and restore the database is with DB2 **backup** and **restore**. The LDAP alternatives, like **db2ldif** and **ldif2db**, are generally much slower in comparison. Although the IBM Directory **bulkload** tool is a fast alternative to **ldif2db**, the performance of DB2 **backup** and **restore** is still generally better and easier to use.

The disadvantage to using DB2 **backup** and **restore** is that the backed up database cannot be restored across dissimilar hardware platforms. For example, you cannot back up a database running on an AIX operating system and restore it to a machine running on a Solaris operating system. DB2 alternatives to **backup** and **restore** are **export** and **import**. **export** and **import** are not as fast as **backup** and **restore**, but work across dissimilar hardware platforms. Refer to DB2 documentation for more information on **export** and **import** usage.

Attention: Be aware that if you restore over an existing database, any tuning that has been done on that existing database is lost.

Check all DB2 configuration parameters after performing a restore. Also, run **reorgchk** after a restore, if it is not known whether a **reorgchk** was performed before the database was backed up. The DB2 commands to perform a **backup** and **restore** are as follows:

```
db2 force applications all
db2 backup db ldapdb2 to <directory or device>
db2 restore db ldapdb2 from  <directory or device> replace \
existing
```

where *<directory or device>* is the name of a directory or device where you want the backup to be placed or from where the restore is to come.

The most common error that occurs on a restore is a file permission error. Some possible reasons for this error might be:

- The DB2 instance owner does not have permission to access the specified directory and file. One way to solve this is to change directory and file ownership to the DB2 instance owner. A command similar to the following will do this:

  ```
  chown ldapdb2 <file or dir>
  ```

- The backed-up database is spread across multiple directories, and those directories do not exist on the target machine of the restore. Spreading the database across multiple directories is accomplished with a redirected restore.

  To solve this problem, you should either create the same directories on the target machine or do a redirected restore to specify the proper directories on the new machine. If creating the same directories, make sure the owner of the directories is ldapdb2.

backup and **restore** are required to get an LDAP replica initially synchronized with an LDAP master. **backup** and **restore** are also required any time the master and replica get out of sync. A replica can become out of sync if it is undefined to the master. The master will not save updates on a propagation queue for a replica that is not defined.

If a newly configured master LDAP directory is to be loaded with initial data, bulk loading utilities can be used to speed up the process. This is another case in which the replica is not informed of updates and a manual back up and restore is required to get the replica synchronized with the master.

Chapter 4. AIX Operating system tuning

This chapter discusses the following performance tuning tasks for the AIX operating system:

- Enabling large files
- Setting MALLOCMULTIHEAP
- Viewing slapd environment variables (AIX operating system only)

Enabling large files

The underlying AIX operating system files that hold the contents of a large directory can grow beyond the default size limits imposed by the AIX operating system. If the size limits are reached, the directory ceases to function correctly. The following steps make it possible for files to grow beyond default limits on an AIX operating system:

1. When you create the file systems that are expected to hold the directory's underlying files, you should create them as "Large File Enabled Journaled File Systems". The file system containing the DB2 instance's home directory (usually `/home/ldapdb2`), and, if **bulkload** is to be used, are file systems that can be created this way. The file system containing the **bulkload** temporary directory can be specified via the LDAPIMPORT environment variable.
2. Set the soft file size limit for the root, ldap, and the DB2 instance owner (usually ldapdb2) users to **-1**. A soft file size limit of -1 for a user specifies the maximum file size for that user as unlimited. The soft file size limit can be changed using the **smitty chuser** command. It is necessary for each user to log off and log back in for the new soft file size limit to take effect. You must also restart DB2.

Setting MALLOCMULTIHEAP

The MALLOCMULTIHEAP environment variable can improve LDAP performance on SMP systems. To set this variable, run the following command just before starting slapd:

```
export MALLOCMULTIHEAP=1
```

The disadvantage to using MALLOCMULTIHEAP is increased memory usage.

It might take less memory, yet have less of a performance benefit, if the variable is set as follows:

```
export MALLOCMULTIHEAP=heaps: <numprocs+1>
```

where <*numprocs*> is the number of processors in the multiprocessor system.

More information on MALLOCMULTIHEAP can be found in the AIX documentation.

Viewing slapd environment variables (AIX operating system only)

To view the environment settings and variables for your slapd process, run the following command:

```
ps ewww <PID>
```

where *PID* is the slapd process ID.

Example output for a PID of 20788:

```
$ ps ewww  20788
   PID    TTY STAT  TIME COMMAND
 20788  pts/0 A    20:04 /usr/bin/slapd -f /etc/slapd32.conf _=/usr/bin/slapd MA
NPATH=/usr/dt/man:/usr/share/man:/usr/lpp/info:/usr/lpp/ssp/man DSHPATH=/usr/loc
al/bin/setupserver:/usr/lpp/ssp/rcmd/bin:/usr/lpp/ssp/bin:/usr/lpp/ssp/kerberos/
bin:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin:/u/shared/bin:/usr/local
/bin:/usr/local/bin:/u/shared/bin LANG=en_US LOGIN=root IMQCONFIGCL=/etc/IMNSear
ch/dbcshelp PATH=/usr/local/bin/setupserver:/usr/lpp/ssp/rcmd/bin:/usr/lpp/ssp/b
in:/usr/lpp/ssp/kerberos/bin:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
:/u/shared/bin:/usr/local/bin:/usr/local/bin:/u/shared/bin:/var/ifor:/usr/opt/if
or/ls/conf:/db2home/instldap/sqllib/bin:/db2home/instldap/sqllib/adm:/db2home/in
stldap/sqllib/misc GMQ_XLAT_PATH=/usr/lpp/mqm/mqlsx/conv RPC_UNSUPPORTED_NETIFS=
en0:et0:fi0:en1 LC__FASTMSG=true CGI_DIRECTORY=/var/docsearch/cgi-bin IMQCONFIGS
RV=/etc/IMNSearch EDITOR=vi HISTFILE=/tmp/.root_history.carya LOGNAME=root SP_NA
ME=spgwa3e MAIL=/usr/spool/mail/root LOCPATH=/usr/lib/nls/loc PS1=?$FARM:$PWD?[!
]$HOST:$ID] PS2=->  HOST=f1n4e USER=root DOCUMENT_SERVER_MACHINE_NAME=spgwa1e AU
THSTATE=compat SHELL=/bin/ksh ODMDIR=/etc/objrepos JAVA_HOME=/usr/jdk_base HISTS
IZE=2000 DOCUMENT_SERVER_PORT=49213 TMOUT=3600 RDBM_CACHE_SIZE=0 HOME=/root DB2I
NSTANCE=instldap TERM=vt220 MAILMSG=[YOU HAVE NEW MAIL] K5MUTE=1 PWD=/db2home/ld
ap DOCUMENT_DIRECTORY=/usr/docsearch/html TZ=MST7MDT FARM=GWA C MALLOCMULTIHEAP=
4 A__z=! LOGNAME="*TMOUT LIBPATH=/usr/lib NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/li
b/nls/msg/%L/%N.cat:/usr/lib/nls/msg/En_US DB2CODEPAGE=1208 DB2CP=1208 LDAP_CONC
URRENTRW=TRUE RDBM_RESULT_SIZE_LIMIT=100 RDBM_ENTRY_CACHE_BYPASS=1 SLAPD_WORKERS
=35 SLAPD_DEFAULT_ACLS=TRUE RDBM_FCACHE_SIZE=1000 ACLCACHESIZE=10
```

Chapter 5. Hardware tuning

This chapter contains some suggestions for improving disk drive performance.

Disk speed improvements

With millions of entries in LDAP server, it can become impossible to cache all of them in memory. Even if a smaller directory size is cacheable, update operations must go to disk. The speed of disk operations is important. Here are some considerations for helping to improve disk drive performance:

- Use fast disk drives
- Use a hardware write cache
- Spread data across multiple disk drives
- Spread the disk drives across multiple I/O controllers
- Put log files and data on separate physical disk drives

Chapter 6. IBM Directory features

The sections in this chapter briefly describe the following additional performance-related IBM Directory features.

- Bulk loading (**bulkload**)
- Replication
- Monitoring Performance
- When to configure LDAP change log

Bulk loading (bulkload)

The **bulkload** utility loads directory data to the LDAP database using an ldif file. **bulkload** usually is significantly faster than **ldif2db** and **ldapadd** when loading approximately 100,000 to a million entries. Read the **bulkload** documentation in the *IBM Directory Server Version 4.1 Administration Guide* for information about using **bulkload**.

Replication

LDAP supports replication. Through replication, LDAP maintains multiple, synchronized copies of the LDAP master directory server. These copies are called LDAP replica servers. The process whereby the master server sends updates to its replicas is known as propagation.

A tuning parameter related to LDAP replication is the update interval. It specifies how long the LDAP master waits between propagations. Both immediate and delayed updates are supported.

The amount of CPU usage required in the replica server for propagations is the same with both immediate and delayed updates. The primary difference between the two choices is when the CPU is consumed, rather than how much CPU is consumed to do the updates. It might be beneficial to delay updates to reduce the cost of cache invalidation that results from updates.

The primary performance cost to replication is the increased time it takes to perform updates on the master. This increased time is the same regardless of the selection of immediate or delayed updates.

For best update performance in a replicated environment, run the LDAP master on the fastest processor available. If the environment is set up so that all searches go only to the replica servers, the LDAP master can be run on a uniprocessor.

Monitoring Performance

The following **ldapsearch** command can be used to monitor performance.

```
ldapsearch -h <ldap_host> -s base -b cn=monitor objectclass=*
```

where <*ldap_host*> is the name of the LDAP host.

The monitor search returns some of the following attributes of the server:

version	Version of the LDAP server
totalconnections	Total number of connections to the server
currentconnections	Total number of current connections
maxconnections	Configured maximum number of connections
writewaiter	Number of threads waiting to write
readwaiters	Number of threads waiting to read
opsinitiated	Operations initiated against the server
opscompleted	Number of operations completed
entriessent	Number of entries sent from the server
searchesrequested	Number of searches requested
searchescompleted	Number of searches completed
filter_cache_size	Configured maximum size of the filter cache
filter_cache_current	Current size of the filter cache
filter_cache_hit	Number of searches that have hit the filter cache
filter_cache_miss	Number of searches that have missed the filter cache
entry_cache_size	Configured maximum size of the entry cache
entry_cache_current	Current size of the entry cache
entry_cache_hit	Number of entries returned from entry cache
entry_cache_miss	Number of entries returned not from entry cache
currenttime	Current time of the search
starttime	Start time of the server
en_currentregs	Number of events currently registered
en_notificationssent	Number of event notifications sent

Example

The following example shows how to calculate the throughput of the server by monitoring the server statistic called **opsinitiated**, which is the number of operations initiated since the LDAP server started.

Suppose the values for the opsinitiated attribute obtained by issuing two ldapsearch commands to monitor the performance statistics, one at time t1 and the other at a later time **t2**, were opsinitiated (**t1**) and opsinitiated (**t2**.) Then, the average throughput at the server during the interval between **t1** and **t2** can be calculated as:

```
(opsinitiated(t2) - opsinitiated(t1) - 3)/(t2 -t1)
```

(3 is subtracted to account for the number of operations performed by the **ldapsearch** command itself.)

When to configure LDAP changelog

IBM Directory 4.1 has a function called **change log** that results in a significantly slower LDAP update performance. The **change log** function should be configured only if needed.

The **change log** function causes all updates to LDAP to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the LDAP server Directory Information Tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default.

One way to check for existence of the **change log** function is to look for the suffix CN=CHANGELOG. If it exists, the **change log** function is enabled.

Appendix.

Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX DB2 IBM SecureWay

Microsoft®, MS-DOS, Windows, and Windows NT® are registered trademarks of Microsoft Corporation

Other company, product, and service names may be trademarks or service marks of others.

Printed in U.S.A.