IBM Tivoli Directory Server

Performance Tuning and Capacity Planning Guide

Version 6.3

SC27-2748-00

IBM Tivoli Directory Server

Performance Tuning and Capacity Planning Guide

Version 6.3

SC27-2748-00

This edition applies to version 6, release 3, of IBM Tivoli Directory Server and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this book

IBM® Tivoli® Directory Server is the IBM implementation of Lightweight Directory Access Protocol for supported Windows®, AIX®, Linux® (System x®, System z®, System p®, and System i®), Solaris, and Hewlett-Packard UNIX® (HP-UX) (Itanium®) operating systems.

IBM Tivoli Directory Server Version 6.3 Performance Tuning and Capacity Planning Guide contains information about tuning the directory server for better performance.

Intended audience for this book

This book is for system administrators, network administrators, information technology architects, and application developers.

Readers need to know how to use the operating system on which IBM Tivoli Directory Server will be installed.

Publications

This section lists publications in the IBM Tivoli Directory Server version 6.3 library and related documents. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

IBM Tivoli Directory Server version 6.3 library

The following documents are available in the IBM Tivoli Directory Server version 6.3 library:

- *IBM Tivoli Directory Server Version 6.3 What is New for This Release*, GC27-2746-00

  Provides information about the new features in the IBM Tivoli Directory Server Version 6.3 release.
- *IBM Tivoli Directory Server Version 6.3 Quick Start Guide*, GI11-9351-00

  Provides help for getting started with IBM Tivoli Directory Server 6.3. Includes a short product description and architecture diagram, as well as a pointer to the product Information Center and installation instructions.
- *IBM Tivoli Directory Server Version 6.3 System Requirements*, SC27-2755-00

  Contains the minimum hardware and software requirements for installing and using IBM Tivoli Directory Server 6.3 and its related software. Also lists the supported versions of corequisite products such as DB2® and GSKit.
- *IBM Tivoli Directory Server Version 6.3 Installation and Configuration Guide*, SC27-2747-00

  Contains complete information for installing, configuring, and uninstalling IBM Tivoli Directory Server. Includes information about upgrading from a previous version of IBM Tivoli Directory Server.
- *IBM Tivoli Directory Server Version 6.3 Administration Guide*, SC27-2749-00

  Contains instructions for performing administrator tasks through the Web Administration Tool and the command line.
- *IBM Tivoli Directory Server Version 6.3 Command Reference*, SC27-2753-00

Describes the syntax and usage of the command-line utilities included with IBM Tivoli Directory Server.

- *IBM Tivoli Directory Server Version 6.3 Server Plug-ins Reference*, SC27-2750-00

  Contains information about writing server plug-ins.
- *IBM Tivoli Directory Server Version 6.3 Programming Reference*, SC27-2754-00

  Contains information about writing Lightweight Directory Access Protocol (LDAP) client applications in C and Java™.
- *IBM Tivoli Directory Server Version 6.3 Performance Tuning and Capacity Planning Guide*, SC27-2748-00

  Contains information about tuning the directory server for better performance. Describes disk requirements and other hardware needs for directories of different sizes and with various read and write rates. Describes known working scenarios for each of these levels of directory and the disk and memory used; also suggests rough rules of thumb.
- *IBM Tivoli Directory Server Version 6.3 Problem Determination Guide*, GC27-2752-00

  Contains information about possible problems and corrective actions that can be taken before contacting IBM Software Support.
- *IBM Tivoli Directory Server Version 6.3 Messages Guide*, GC27-2751-00

  Contains a list of all informational, warning, and error messages associated with IBM Tivoli Directory Server 6.3.
- *IBM Tivoli Directory Server Version 6.3 White Pages*, SC27-2756-00

  Describes the Directory White Pages application, which is provided with IBM Tivoli Directory Server 6.3. Contains information about installing, configuring, and using the application for both administrators and users.

Related publications

The following documents also provide useful information:

- *Java Naming and Directory Interface™ 1.2.1 Specification* on the Sun Microsystems Web site at http://java.sun.com/products/jndi/1.2/javadoc/index.html.

  IBM Tivoli Directory Server Version 6.1 and above versions use the Java Naming and Directory Interface (JNDI) client from Sun Microsystems. See this document for information about the JNDI client.

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at the following Web address:

http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Information Center Web site at http://publib.boulder.ibm.com/tividd/td/link/tdprodlist.html.

In the Tivoli Information Center window, click **Tivoli product manuals**. Click the letter that matches the first letter of your product name to access your product library. For example, click **M** to access the IBM Tivoli Monitoring library or click **O** to access the IBM Tivoli OMEGAMON® library.

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Documentation Central Web site at http://www.ibm.com/tivoli/documentation.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File → Print** window that allows Adobe® Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.
2. Select your country from the list and click **Go**.
3. Click **About this site** in the main panel to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

Visit the IBM Accessibility Center at http://www.ibm.com/alphaworks/topics/accessibility/ for more information about IBM's commitment to accessibility.

For additional information, see the Accessibility Appendix in the *IBM Tivoli Directory Server Version 6.3 Installation and Configuration Guide*.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site at http://www.ibm.com/software/tivoli/education.

Tivoli user groups

Tivoli user groups are independent, user-run membership organizations that provide Tivoli users with information to assist them in the implementation of Tivoli Software solutions. Through these groups, members can share information and learn from the knowledge and experience of other Tivoli users. Tivoli user groups include the following members and groups:

- 23,000+ members
- 144+ groups

Access the link for the Tivoli Users Group at www.tivoli-ug.org.

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online

Access the Tivoli Software Support site at http://www.ibm.com/software/sysmgmt/products/support/index.html?ibmprd=tivman. Access the IBM Software Support site at http://www.ibm.com/software/support/probsub.html .

IBM Support Assistant

The IBM Support Assistant is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The Support Assistant provides quick access to support-related information and serviceability tools for problem determination. To install the Support Assistant software, go to http://www.ibm.com/software/support/isa.

Troubleshooting Guide

For more information about resolving problems, see the *IBM Tivoli Directory Server Version 6.3 Problem Determination Guide*.

Conventions used in this book

This book uses several conventions for special terms and actions, operating system-dependent commands and paths, and margin graphics.

Typeface conventions

This book uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:**, and **Operating system considerations**:)
- Keywords and parameters in text

Italic

- Citations (examples: titles of books, diskettes, and CDs)
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide: ... where *myname* represents....

`Monospace`

- Examples and code examples

- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This book uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace $*variable* with **%** *variable***%** for environment variables and replace each forward slash (**/**) with a backslash (\) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. IBM Tivoli Directory Server tuning general overview

This guide provides tuning information for IBM Tivoli Directory Server and the related IBM Database 2 (DB2) database. IBM Tivoli Directory Server is a Lightweight Directory Access Protocol (LDAP) directory that provides a layer on top of DB2, allowing users to efficiently organize, manipulate, and retrieve data stored in the DB2 database. Tuning for optimal performance is primarily a matter of adjusting the relationships between the LDAP server and DB2 according to the nature of your workload.

Because each workload is different, instead of providing exact values for tuning settings, guidelines are provided, where appropriate, for how to determine the best settings for your system.

Attention: Measurements in this guide were captured in a lab environment. The workload driving this test included a mixture of searches and binds, including wildcard searches which return multiple entries. Your results might differ from the lab results shown in this guide.

IBM Tivoli Directory Server application components

The following figure illustrates how IBM Tivoli Directory Server components interact with each other. Tuning these components can result in improved performance.

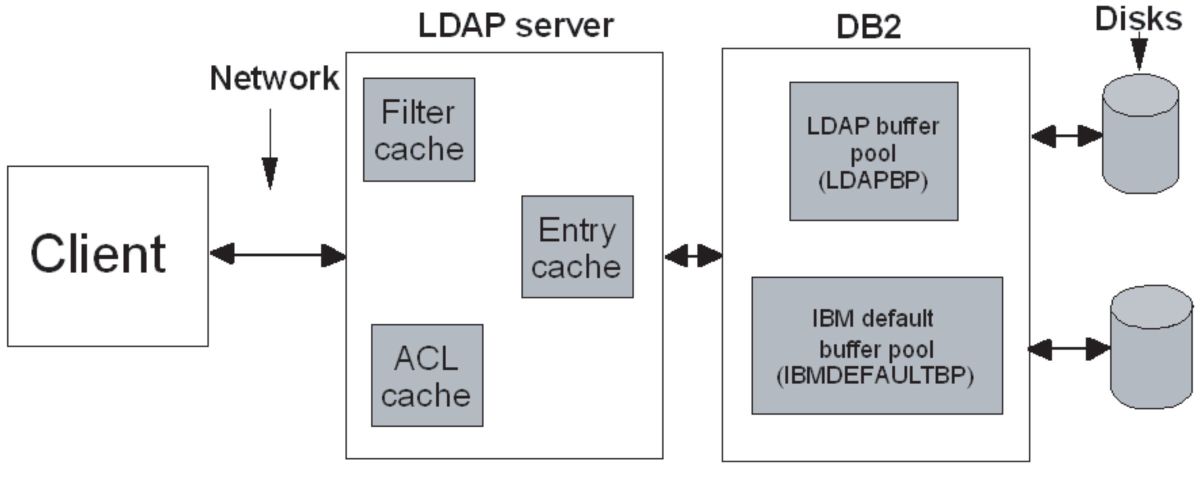


Figure 1. IBM Tivoli Directory Server

The arrows in Figure 1 represent the path of a query issued from a client computer. The query follows a path from the IBM Tivoli Directory Server client to the LDAP server, to DB2, to the physical disks in search of entries that match the query's search filter settings. The shorter the path to matching entries, the better overall performance you can expect from your system.

For example, if a query locates all the matching entries in the LDAP server, access to DB2 and the disks is not necessary. If the matching entries are not found in the

LDAP server, the query continues on to DB2 and, if necessary, to the physical disks as a last resort. Because of the time and resources it takes to retrieve data from disk, it is better from a performance standpoint to allocate a significant amount of memory to the LDAP server caches and setting the DB2 buffer pools to AUTOMATIC.

LDAP caches and DB2 buffer pools

Caches and buffer pools store previously retrieved data and can significantly improve performance by reducing disk access. When requested data is found within a cache or buffer pool, it is called a cache hit. A cache miss occurs when requested data is not located in a cache or buffer pool.

Because the type of information in each cache and buffer pool is different, it is useful to understand how and when each cache is accessed.

LDAP caches

Search operations attempt to use one or more caches when resolving the search filter as well as returning the individual matching entries. Most base-scoped searches can be resolved directly in memory by retrieving the base entry from the entry cache or the database buffer pool and performing the comparison of the entry with the filter.

If a base-scoped search cannot be resolved directly in memory or the search is not base-scoped, an attempt is made to use the attribute cache to resolve the filter in memory or to use the filter cache to retrieve the results of a previously run search operation. If LDAP caches cannot be used to resolve the filter, the filter will be resolved using DB2. When the individual entries are returned to the client, they are retrieved from memory using the entry cache, if possible. If the individual entries are not found in the entry cache, they are retrieved from DB2

The four LDAP caches are:

- LDAP attribute cache

  Note: Starting with the IBM Tivoli Directory Server 6.3 release, attribute cache is deprecated. Henceforth, users should avoid using attribute cache.
- LDAP filter cache
- Entry cache
- ACL cache

For more information on these caches, see "LDAP caches" on page 7.

DB2 buffer pools

There are two DB2 buffer pools:

LDAPBP

LDAPBP contains cached entry data (ldap_entry) and all of the associated indexes. LDAPBP is similar to the entry cache, except that LDAPBP uses different algorithms in determining which entries are cached. It is possible that an entry that is not cached in the entry cache is located in LDAPBP. If the requested data is not found in the entry cache or LDAPBP, the query must access the physical disks.

IBMDEFAULTBP

DB2 system information, including system tables and other LDAP information, is cached in the IBMDEFAULTBP.

IBM Tivoli Directory Server tuning overview

Tuning the LDAP server can significantly improve performance by storing useful data in the caches. It is important to remember, however, that tuning the LDAP server alone is insufficient. Some tuning of DB2 is also required for optimal performance.

To get best results it is advisable to tune Tivoli Directory Server instance immediately after the instance has been configured. If the Tivoli Directory Sever instance is not tuned, it is likely to perform poorly as the size of directory instance grows as the entries are added to the instance. Considerations that could be used as a trigger for tuning a Tivoli Directory Server instance:

- Poor or slow search and update response time, or slow execution time for the bulkload utility. To prevent poor performance, tune the Tivoli Directory Server at least one time and subsequently update the DB2 system statistics after any large number of updates to the server, for example after adding a large number of entries.

Before tuning Tivoli Directory Server, it is recommended to take backup of the instance, database, and configuration files. To know more backing up and restoring Tivoli directory Server instance, see the usage of idsdbback and idsdbrestore commands in *IBM Tivoli Directory Server Version 6.3 Command Reference*.

The most significant performance tuning related to the IBM Tivoli Directory Server involves the LDAP caches. LDAP caches 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 useful mostly for applications that frequently retrieve repeated cached information, they can greatly improve performance by avoiding calls to the database. See "LDAP caches" on page 7 for information about how to tune the LDAP caches.

DB2 tuning overview

DB2 serves as the data storage component of the IBM Tivoli Directory Server. Tuning DB2 results in overall improved performance.

This guide contains several recommendations for tuning DB2, but the most commonly tuned items are:

- **DB2 buffer pools –** Buffer pools are DB2 data caches. Each buffer pool is a data cache between the applications and the physical database files. In IBM Tivoli Directory Server V6.2 and above versions, automatic tuning of buffer pools are performed by default. See "Tuning DB2 buffer pool" on page 24 for information about buffer pool tuning.
- **Optimization and organization –** After initially loading a directory, or after a number of updates have been performed, it is very important to update database statistics and organization for DB2 to perform optimally. See "Optimization and organization (idsrunstats, reorgchk and reorg)" on page 40 for more information.
- **Indexes –** Indexes can make locating data on disk very fast, providing a significant boost to performance. For information about how to create indexes, see "DB2 indexes" on page 45.

Attention: You should place the DB2 log on a physical disk drive separate from the data. For improved data-integrity and performance, have the DB2 log and the data on separate drives. Use the following command to set the path to the DB2 log file directory:

```
DB2 UPDATE DATABASE CONFIGURATION FOR database_alias USING NEWLOGPATH path
```

Be sure the database instance owner has write access to the specified path or the command fails. For more information on using DB2 commands, see Chapter 3, “Tuning DB2 and LDAP caches,” on page 23.

Users can also use the idsperftune utility to improve the performance of the Tivoli Directory Server, see “The performance tuning tool (idsperftune)” on page 28.

Performance impact due to multiple password policy

To evaluate a user’s effective password policy, the directory server takes into consideration all the password policies associated with a user. This means that the directory server evaluates the individual, group, and global password policies to determine a user’s effective password policy.

If you have defined group password policies, the performance of bind operations may be degraded. This is because during authentication a user's effective password policy must be determined, and group membership must be resolved to properly apply group password policies.

Enforcing minimum ulimits

The directory server tries to enforce minimum ulimit values such process data size, process virtual memory, and process file size that are considered important for the smooth running of the server. To accomplish this, the directory server first checks if the ulimit values for the current processes are greater than or equal to the prescribed ulimit values specified in the configuration file. In case the ulimit values for the current processes are lesser than the prescribed values, the server attempts to set the ulimit values of the current processes to the prescribed values.

On Linux or UNIX operating systems, resource limits are defined for each user. When a process is started, that process inherits or takes the resource limits of the user context under which it was started. For example, if the Tivoli Directory Server process, idsslapd, is started under the root user context, the idsslapd process takes on the resource limits of the root user. This inheritance occurs even if the process switches user contexts as the idsslapd process does. The idsslapd process switches the user context to the DB2 instance owner if the need is to have the idsslapd process take on the resource limits of the DB2 instance owner. In this case, the idsslapd process must be started while the DB2 instance owner is logged in.

In Tivoli Directory Server v6.2 and later versions, the directory server sets the ulimits based on the configuration file. The following entries under the DN entry cn=Ulimits, cn=Configuration can be used to modify the existing limits:

```
dn: cn=Ulimits, cn=Configuration
cn: Ulimits
ibm-slapdUlimitDataSegment: 262144
ibm-slapdUlimitDescription: Prescribed minimum ulimit option values
ibm-slapdUlimitFileSize: 2097152
ibm-slapdUlimitNofile: 500
ibm-slapdUlimitStackSize: 10240
```

```
ibm-slapdUlimitVirtualMemory: 1048576
objectclass: top
objectclass: ibm-slapdConfigUlimit
objectclass: ibm-slapdConfigEntry
```

Note: An administrator can modify the minimum ulimit values using the web administration tool or through command line.

Generic LDAP application tips

The following are some tips that can help improve performance:

- Perform searches on indexed attributes only. See "DB2 indexes" on page 45 for instructions for defining and verifying indexes for IBM Tivoli Directory Server.
- 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 only the Distinguished Names (DNs), and not the entire group. Do not request the member or uniquemember attributes if possible.
- Minimize and batch updates (**add**, **modify**, **modrdn**, **delete**) when possible.
- Use base-scoped searches whenever possible rather than one-level or subtree searches.
- Avoid using wildcard searches where the wildcard is in any position other than the leading character in a term, or a trailing character. Use wildcard searches that are similar to the following (leading character):

  ```
  sn=*term
  ```

  or the following (trailing character):

  ```
  sn=term*
  ```

  Note: A filter such as `sn=*term*` is less efficient than the examples given.
- When using nested groups, keep the depth of nesting to 50 groups or less. Greater nesting depths can result in greater processing times when performing add or delete operations that involve updates to the nested group hierarchy.
- Set server search limits to prevent accidental long-running searches.
- Use the ldap_modify interface to add members to or delete members from a group. Do not do a search to retrieve all members, edit the returned list, then send the updated list as a modify-replace operation. This modify-replace scenario will not perform well with large groups.
- For a proxy server, do not set the value in the **Connection pool size** field to be less than **5**.

Chapter 2. IBM Tivoli Directory Server tuning

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

- Tuning LDAP caches
- Determining how directory size affects performance

LDAP caches

LDAP caches are fast storage buffers in memory used to store LDAP information such as queries, answers, and user authentication for future use. Tuning the LDAP caches is crucial to improving performance.

An LDAP search that accesses the LDAP cache can be faster than one that requires a connection to DB2, even if the information is cached in DB2. For this reason, tuning LDAP caches can improve performance by avoiding calls to the database. The LDAP caches are especially useful for applications that frequently retrieve repeated cached information. See Figure 1 on page 1 for an illustration of the LDAP caches.

The following sections discuss each of the LDAP caches and demonstrate how to determine and set the best cache settings for your system. Keep in mind that every workload is different, and some experimentation will likely be required in order to find the best settings for your workload.

Note: Cache sizes for the filter cache, ACL cache, and entry cache are measured in numbers of entries.

LDAP attribute cache

Note: Starting with the IBM Tivoli Directory Server 6.3 release, attribute cache is deprecated. Henceforth, users should avoid using attribute cache.

The attribute cache stores configured attributes and their values in memory. When a one-level or sub-tree search is performed, or a base-scoped search is performed that cannot be resolved directly in memory using the entry cache, the attribute cache manager resolves the search filter in memory if all attributes used in the filter are cached and the filter is a type supported by the attribute cache manager. Resolving filters in memory leads to improved search performance over resolving filters using DB2.

There are two ways to configure which attributes are to be kept in the attribute cache. It can be configured to automatically select the attributes expected to provide the most benefit, see "directory automatic attribute caching" in the section Configuring attribute caching. Alternatively, the administrator can choose specific attributes that they wish to be cached. The following section gives some tips on how to determine the best attributes to keep in the attribute cache.

There are two things that can happen when a query arrives at the attribute cache:

- **All attributes used in the search filter are cached and the filter is of a type that can be resolved by the attribute cache manager.** If this is the case, the list of matching entry IDs is resolved in memory using the attribute cache manager.

This list of matching IDs is then sent to the entry cache. For this reason, the attribute cache is most efficient when used in combination with the entry cache

The attribute cache manager can resolve simple filters of the following types:

– exact match filters
– presence filters

The attribute cache manager can also resolve complex filters that are conjunctive or disjunctive. Additionally, the subfilters within complex filters must be exact match, presence, conjunctive, or disjunctive.

– exact match filters
– presence filters
– conjunctive filters
– disjunctive filters

Filters containing attributes with language tags are not resolved by the attribute cache manager.

For example, if the attributes objectclass, uid, and cn are all cached, the following filters can be resolved in memory within the attribute cache manager:

– `(cn=Karla)`
– `(cn=*)`
– `(&(objectclass=eperson)(cn=Karla))`
– `(&(objectclass=eperson)(cn=*)(uid=1234567))`
– `(&(&(objectclass=eperson)(cn=*))(uid=1234567))`
– `(&(uid=1234567)(&(objectclass=eperson)(cn=*)))`

- **Some or all of the attributes used in the search filter are not cached or the filter is of a type that cannot be resolved by the attribute cache manager.** If this is the case, the query is sent to the filter cache for further processing.

  Note: If there are no attributes in the attribute cache, the attribute cache manager determines this quickly, and the query is sent to the filter cache.

  For example, if the attributes objectclass, uid, and cn are the only cached attributes, the following filters will not be able to be resolved in memory by the attribute cache manager:

  – `(sn=Smith)`
  – `(cn=K*)`
  – `(|(objectclass=eperson)(cn~=Karla))`
  – `(&(objectclass=eperson)(cn=K*)(uid=1234567))`
  – `(&(&(objectclass=eperson)(cn<=Karla))(uid=1234567))`
  – `(&(uid=1234567)(&(objectclass=eperson)(sn=*)))`

Note: Choosing to cache member, uniquemember, or ibm-membergroup can lead to slower performance of delete and modrdn operations. If the entry being deleted or renamed is a member of many groups, or large groups, then the attribute caches need to be updated to reflect this change for every group in which the entry was a member.

Determining which attributes to cache

To determine which attributes to cache, experiment with adding some or all of the attributes listed in the **cached_attribute_candidate_hit** attribute to the attribute cache. Then run your workload and measure the differences in operations per second. For information about the **cached_attribute_candidate_hit** attribute, see “ldapsearch with "cn=monitor"” on page 63.

Note: Choosing to cache member, uniquemember, or ibm-membergroup can lead to slower performance of delete and modrdn operations. If the entry being deleted or renamed is a member of many groups or large groups, the attribute caches are updated to reflect this change for every group in which the entry was a member. This additional processing can lead to slower performance of these types of operations.

Examples: Information about attributes that are cached, their individual sizes in kilobytes, and their hit counts can be retrieved during cn=monitor searches. Also, up to ten attributes that are most often used in search filters that can be processed by the attribute cache manager, but are not yet cached, can be retrieved during cn=monitor searches. Use a combination of the output from cn=monitor searches and knowledge of the types of searches your applications use to determine which attributes to cache.

Example 1: The following results are for a cn=monitor search for a server that had no attributes configured for attribute caching:

```
ldapsearch -h ldaphost -s base -b cn=monitor objectclass=*
 cached_attribute_total_size
cached_attribute_configured_size cached_attribute_hit cached_attribute_size
cached_attribute_candidate_hit
cn=monitor
cached_attribute_total_size=0
cached_attribute_configured_size=1200
cached_attribute_candidate_hit=mail:50000
cached_attribute_candidate_hit=uid:45000
cached_attribute_candidate_hit=givenname:500
cached_attribute_candidate_hit=sn:200
```

If this cn=monitor search produced these results, you can assume that the attributes to cache must be `uid` and `mail`. Even though `givenname` and `sn` were used in search filters that have been resolved by the attribute cache manager had those attributes been cached, their hit counts are very low in comparison to the attributes `uid` and `mail`, and using memory to store `givenname` and `sn` is not realistic.

After the attributes `uid` and `mail` are cached and the application or performance test is rerun, the cn=monitor search should be performed again to determine if there is enough memory configured to cache both attributes. If there is not enough memory, then additional memory must be configured, or the least-used attribute must be removed from the list of attributes to cache.

Example 2: In this example, `givenname` and `sn` are already cached. The hit count for `objectclass` is very high. Also, the hit rates for `uid` and `mail` are very high:

```
ldapsearch -h ldaphost -s base -b cn=monitor objectclass=*
 cached_attribute_total_size
cached_attribute_configured_size cached_attribute_hit cached_attribute_size
cached_attribute_candidate_hit
cn=monitor
cached_attribute_total_size=1000
cached_attribute_configured_size=1200
cached_attribute_hit=givenname:500
cached_attribute_size=givenname:300
cached_attribute_hit=sn:200
cached_attribute_size=sn:400
cached_attribute_candidate_hit=objectclass:110000
cached_attribute_candidate_hit=mail:90000
cached_attribute_candidate_hit=uid:85000
cached_attribute_candidate_hit=workloc:25000
```

Note: cached_attribute_total_size is the amount of memory used by the directory attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

As in the previous example, `givenname` and `sn` are not good choices for caching because of their relatively low hit count, in comparison to the other attributes listed. You can assume that `objectclass` is the best choice and that `uid` and `mail` are also excellent choices. If attribute caching is reconfigured to cache `objectclass`, `uid` and `mail`, you might discover after caching is complete and after rerunning your performance tests under the same conditions, that your performance isn't what you expect. Also, the cn=monitor search yields the following unexpected results which show that only objectclass is cached, and its hit count is much lower than when it was a candidate:

```
ldapsearch -h ldaphost -s base -b cn=monitor objectclass=*
 cached_attribute_total_size
cached_attribute_configured_size cached_attribute_hit cached_attribute_size
cached_attribute_candidate_hit
cn=monitor
cached_attribute_total_size=1000
cached_attribute_configured_size=1200
cached_attribute_hit=objectclass:10000
cached_attribute_size=objectclass:750
cached_attribute_candidate_hit=mail:90000
cached_attribute_candidate_hit=uid:85000
cached_attribute_candidate_hit=workloc:25000
cached_attribute_candidate_hit=givenname:300
cached_attribute_candidate_hit=sn:200
```

Two things occurred to cause these results:

1. The `objectclass` attribute table was large in comparison to the other attribute tables. Even though `objectclass`, `uid` and `mail` were all configured to be cached, `objectclass` was the only attribute that fit within the maximum memory configured for attribute caching.
2. Further analysis of the search filters used by your application reveals that `objectclass` was not used in search filters by itself very often. The attribute cache manager could not resolve many filters because not all attributes in the filter were cached. A combination of the cn=monitor output and analysis of the filters used by your application is necessary to determine which attributes to cache. The following search filters were used in this example:

```
(objectclass=*)                    10000 hits
(givenname=*)                        300 hits
(sn=*)                               200 hits
(mail=*)                           50000 hits
(uid=*)                            45000 hits
(workloc=*                          5000 hits
(&(objectclass=person)(mail=*))    40000 hits
(&(objectclass=person)(uid=*))     40000 hits
(&(objectclass=person)(workloc=*)) 20000 hits
```

You can see from the above filter analysis that `objectclass`, when used alone, had only 10000 hits. Therefore, if the only attribute cached is `objectclass`, the attribute cache manager can only resolve 10000 out of the 210500 total search filters. If the server is reconfigured to have enough memory to hold both the `objectclass` and `mail` attributes, 100000 of the search filters can be resolved in the attribute cache manager. If `objectclass`, `uid` and `mail` were all configured and enough memory was available, 185000 of the search filters can be resolved by the attribute cache manager. However, if memory is constrained and only one attribute can be cached, the best choice is `mail` with 50000 hits. If both `uid` and `mail` can be cached, 95000 filters can be resolved in the attribute cache manager, which is almost as many hits as caching `objectclass` and `mail` instead.

Because caching `uid` and `mail` likely consumes less memory than caching `objectclass` and `mail`, caching `uid` and `mail` instead of `objectclass` and `mail` might be a better choice if not enough memory is available on your server. Therefore, it is necessary to understand and consider the types of search filters used by your application in order to determine the appropriate attributes to cache as well as to consider the amount of memory that you want the attribute cache to be able to use.

LDAP filter cache

The filter cache contains cached entry IDs that match a search filter that was previously resolved in DB2. When the client issues a query for some data and that query is not a base-scoped search that can be resolved in memory nor is it a filter that can be resolved in memory by the attribute cache manager, the query goes to the filter cache. There are two things that can happen when a query arrives at the filter cache:

- **The IDs that match the filter settings used in the query are located in the filter cache.** If this is the case, the list of the matching entry IDs is sent to the entry cache.
- **The matching entry IDs are not cached in the filter cache.** In this case, the query must access DB2 in search of the desired data.

Filter cache size

To determine how big your filter cache should be, run your workload with the filter cache set to different values and measure the differences in operations per second. For example, Figure 2 on page 12 shows varying operations per second based on different filter cache sizes for one installation:

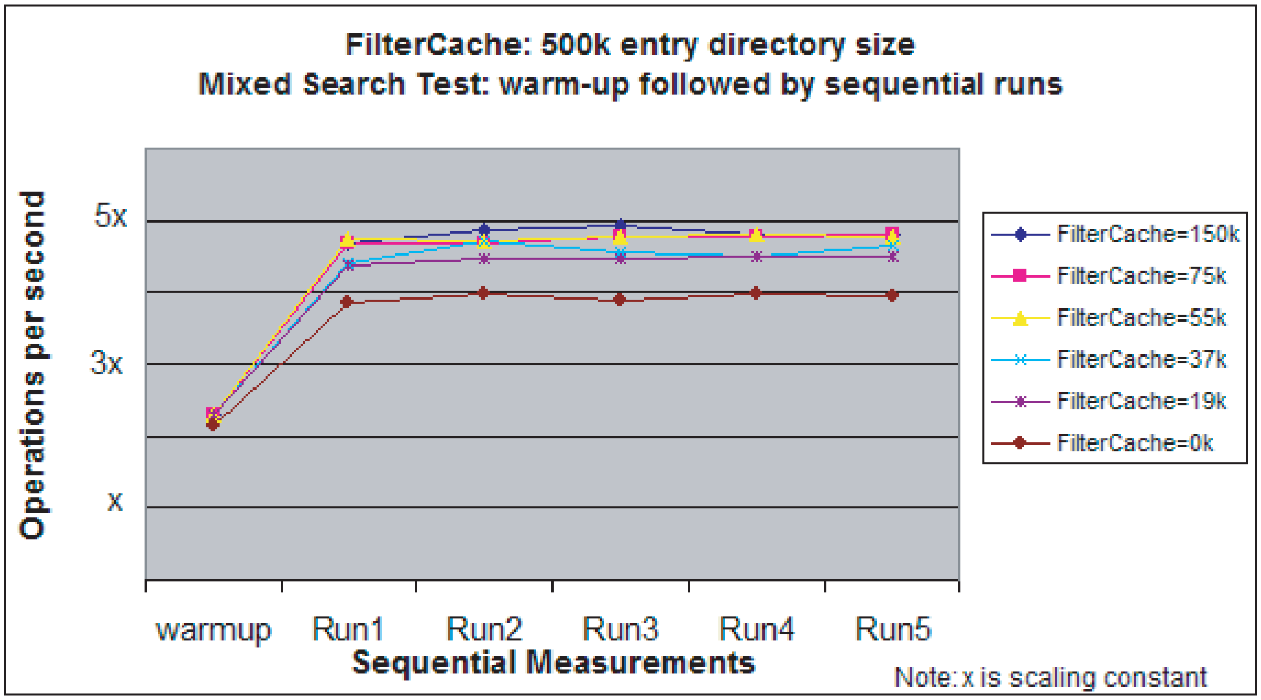


Figure 2. Varying the size of the filter cache

For this workload it appears that a filter cache large enough to hold 55,000 entries results in the best performance. There is no benefit in making the filter cache any larger than this. See "LDAP cache configuration variables" on page 15 to set the filter cache size.

Filter cache size with updates

Figure 3 on page 13 shows that, for the test installation, there is no performance benefit in allocating any memory to the filter cache if even a small fraction of the operations in the workload are updates.

If this proves to be the case for your workload, the only way to retain the performance advantage of a filter cache when updates are involved is to batch your updates. This allows long intervals during which there are only searches. If you cannot batch updates, specify a filter cache size of zero and allocate more memory to other caches. See "LDAP cache configuration variables" on page 15 for instructions on how to set configuration variables such as filter cache size.

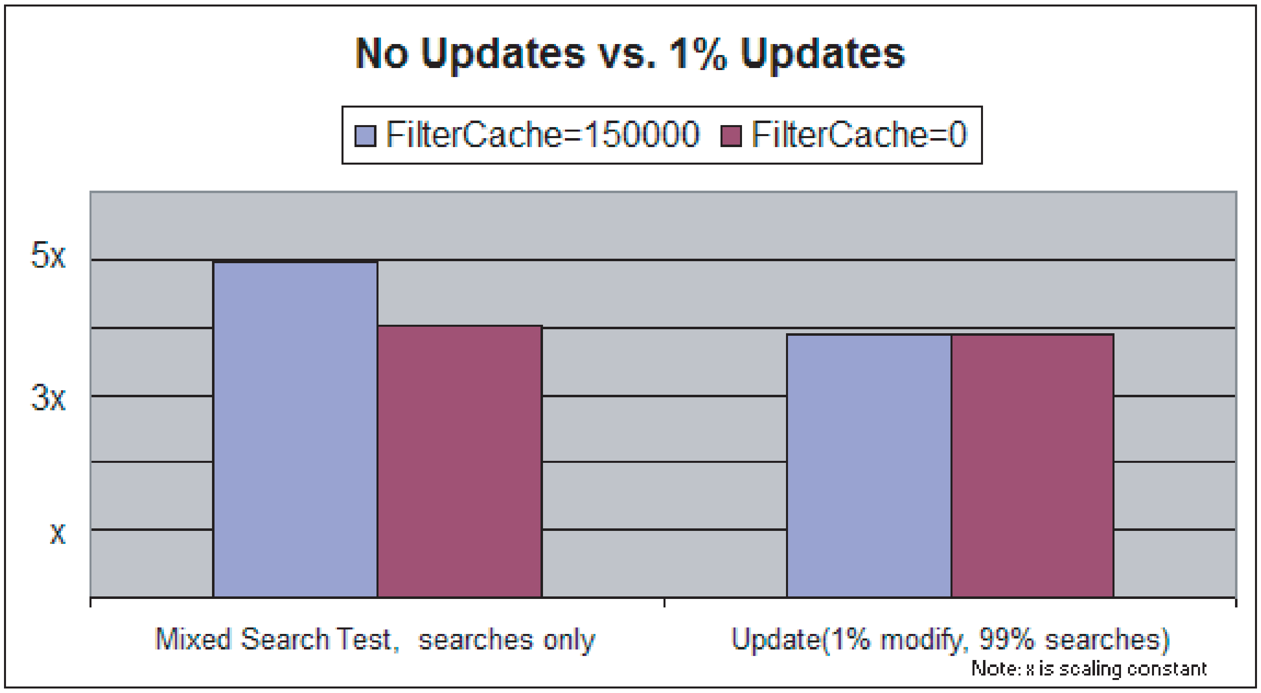


Figure 3. Effect of updates on the performance of the filter cache

Filter cache bypass limits

The filter cache bypass limit configuration variable limits the size of entries that can be added to the filter cache. For example, if the bypass limit variable is set to 1,000, search filters that match more than 1,000 entries are not added to the filter cache. This prevents large, uncommon searches from overwriting useful cache entries. See “LDAP cache configuration variables” on page 15 to set the filter cache bypass limit.

Entry cache

The entry cache contains cached entry data. Entry IDs are sent to the entry cache. If the entries that match the entry IDs are in the entry cache, then the results are returned to the client. If the entry cache does not contain the entries that correspond to the entry IDs, the query goes to DB2 in search of the matching entries.

Entry cache size

To determine how big your entry cache should be, run your workload with the entry cache set to different sizes and measure the differences in operations per second. For example, Figure 4 on page 14 shows varying operations per second based on different entry cache sizes:

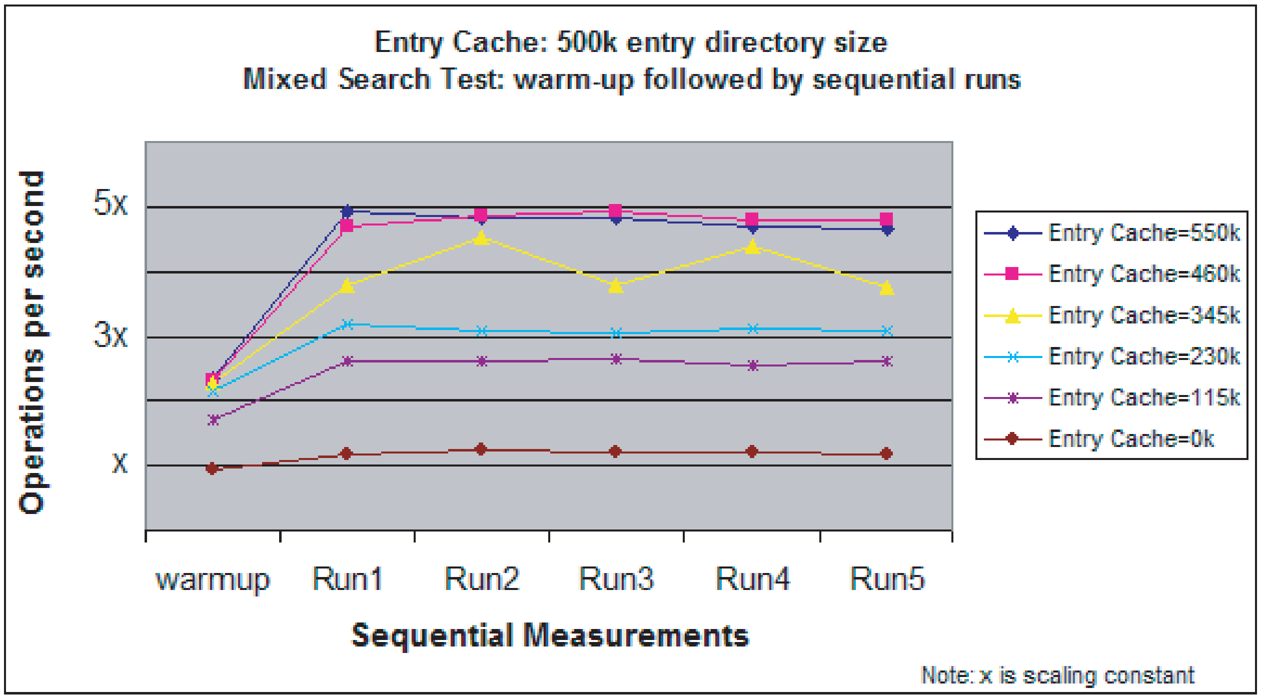


Figure 4. Varying the size of the entry cache

From the results in Figure 4, it appears that an entry cache large enough to hold 460,000 entries results in the best performance. There is no benefit to making the entry cache any larger than this. Setting the entry cache at 460,000 results in 4 times as many operations per second than if entry cache was set to zero. To find the best cache size for your workload, you must run your workload with different cache sizes. See "LDAP cache configuration variables" on page 15 to set the filter cache size.

Note: The test with Entry Cache size at 345k resulted in unpredictable performance due to the nature of the test case and the relationship to the chosen cache size. Certain parts of the workload were in cache while others not, resulting in a harmonics effect.

Group members cache

The group members cache is an extension of the Entry cache. This cache stores member and uniquemember attribute values with their entries. The group entries will only be a part of the group members cache if the entry structures actually have members and uniquemembers. Otherwise, they will be a part of the regular entry cache. Group member caching can be controlled using the two configuration options:

- **ibm-slapdGroupMembersCacheSize**: This defines the number of groups whose members will be cached. The default value for this configuration option is 25.
- **ibm-slapdGroupMembersCacheBypassLimit**: This defines the maximum number of members a group can have in order for it to be cached in the group members cache. The default value of this configuration option is 25000.

ACL cache

The Access Control List (ACL) cache contains information about the access permissions of recently queried entries, such as the entry owner and whether the entry's permissions are explicit or inherited. Having this information cached in memory can speed up the process of determining whether the user who submitted the query is authorized to see all, some, or none of its results.

Measuring cache entry sizes

Filter cache and entry cache sizes are measured in numbers of entries. When determining how many entries to allow in your LDAP caches, it can be useful to know how big the entries in your cache are.

The following example shows how to measure the size of cached entries:

Note: This example calculates the average size of an entry in a sample entry cache, but the average filter cache entry size can be calculated similarly.

1. From the LDAP server:
   a. Set the filter cache size to zero.
   b. Set the entry cache size to a small value; for example, 200.
   c. Start **ibmslapd**.
2. From the client:
   a. Run your application.
   b. Find the entry cache population (call this *population1*) using the following command:
      ```
      ldapsearch -h servername -s base -b cn=monitor objectclass=* | grep
       entry_cache_current
      ```
3. From the LDAP Server:
   a. Find the memory used by **ibmslapd** (call this *ibmslapd1*):
      - On AIX operating systems, use the following command:
        ```
        ps -e -o vsz -o command | grep ibmslapd
        ```
      - On Windows operating systems, use the **VM size** column in the **Task Manager**.
   b. Stop **ibmslapd**.
   c. Increase the size of the entry cache but keep it smaller than your working set.
   d. Start **ibmslapd**.
4. Run your application again and find the entry cache population (call this *population2*). See step 2b for the command syntax.
5. Find the memory used by **ibmslapd** (call this *ibmslapd2*). See step 3a for the command syntax.
6. Calculate the size of an entry cache entry using the following formula:
   ```
   (ibmslapd size2 - ibmslapd size1)  /
   (entry cache population2 - entry cache population1)
   ```

For example, using this formula with a 500,000-entry database results in the following measurement:

```
(192084 KB – 51736 KB) / (48485 – 10003) = 3.65 KB per entry
```

LDAP cache configuration variables

LDAP cache configuration variables allow you to set the LDAP cache sizes, bypass limits, and other variables that affect performance.

Configuring attribute caching

The attribute cache size is measured by the amount of memory the attribute cache requires. You can configure the maximum amount of memory allowed to be used for attribute caching. You can configure attribute caching for the directory database, the changelog database, or both. Typically, there is no benefit from configuring attribute caching for the changelog database unless you perform very frequent searches of the changelog.

Note: Starting with the IBM Tivoli Directory Server 6.3 release, attribute cache is deprecated. Henceforth, users should avoid using attribute cache.

Using the Web Administration Tool

To configure the attribute cache using the Web Administration Tool:

Expand the **Manage server properties** category in the navigation area of the Web Administration Tool, select the **Attribute cache** tab.

1. You can change the amount of memory in kilobytes available to the directory cache. The default is 16384 kilobytes (16 MB).
2. You can change the amount of memory in kilobytes available to the changelog cache. The default is 16384 kilobytes (16 MB).

   Note: This selection is disabled if a changelog has not been configured.

To enable directory automatic attribute caching, perform the following steps:

1. Select the **Enable directory automatic attribute cache** check box. This enables other elements within this group.
2. Enter the start time for directory automatic attribute caching in the **Start Time** text box.
3. From the **Interval** combo box, select the interval at which the directory automatic attribute caching is to be performed again.

To enable change log automatic attribute caching, perform the following steps:

1. Select the **Enable change log automatic attribute cache** check box. This enables other elements within this group.
2. Enter the start time for change log automatic attribute caching in the **Start Time** text box.
3. From the **Interval** combo box, select the interval at which the change log automatic attribute caching is to be performed again.

Note: Automatic attribute caching for change log should not be enabled unless frequent searches within the change log are required and the performance of these searches are critical.

To add an attribute:

1. Select the attribute that you want to cache from the **Available attributes** drop-down menu. Only those attributes that can be designated as cached attributes are displayed in this menu. For example, sn.

   Note: An attribute remains in the list of available attributes until it has been placed in both the Directory and the Changelog containers.
2. Click either **Add to Database** or **Add to Change log** button. The attribute is displayed in the appropriate list box. You can list the same attribute in both containers.

3. Repeat this process for each attribute you want to cache.

   Note: An attribute is removed from the drop-down list when it is added to both the **Cached attributes under Database** and **Cached attributes under Change log** listboxes. If changelog is not enabled, then the **Add to Change log** button is disabled and the entry cannot be added to Cached attributes under Change log list box. The attribute is removed from the available attributes list when it is added to **Cached attributes under Database** list box.
4. When you are finished, click **Apply** to save your changes without exiting, or click **OK** to apply your changes and exit, or click **Cancel** to exit this panel without making any changes.

Using the command line

To configure the attribute cache through the command line, issue the following command:

```
ldapmodify -D <adminDN> -w<adminPW> -i<filename>
```

where *<filename>* contains the following, for example.

- For configuring specific attributes for the directory database:

```
dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory,
  cn=Schemas, cn=Configuration
changetype: modify
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute:  sn
-
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute:  cn
-
replace: ibm-SlapdCachedAttributeSize
ibm-SlapdCachedAttributeSize: 262144
```

- For automatic attribute caching for the directory database:

```
dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory,
     cn=Schemas, cn=Configuration
changetype: modify
replace: ibm-SlapdCachedAttributeSize
ibm-SlapdCachedAttributeSize: 262144
-
replace: ibm-slapdCachedAttributeAutoAdjust
ibm-slapdCachedAttributeAutoAdjust: TRUE
```

- For the changelog database:

```
dn: cn=change log, cn=RDBM Backends, cn=IBM Directory,
  cn=Schemas, cn=Configuration
changetype: modify
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute:  changetype
-
replace: ibm-SlapdCachedAttributeSize
ibm-SlapdCachedAttributeSize: 32768
```

See the *IBM Tivoli Directory Server Version 6.3 Administration Guide* for more information.

Setting other LDAP cache configuration variables

You can set LDAP configuration variables using the Web Administration Tool or the command line.

Using the Web Administration Tool

To set LDAP configuration variables using the Web Administration Tool:

1. Expand the **Manage server properties** category in the navigation area of the Web Administration tool.
2. Click **Performance**.
3. You can modify any of the following configuration variables:
   - **Cache ACL information** — This option must be selected for the **Maximum number of elements in ACL cache** settings to take effect.
   - **Maximum number of elements in ACL cache** (ACL cache size) — The default is 25,000.
   - **Maximum number of elements in entry cache** (entry cache size) — Specify the maximum number of elements in the entry cache. The default is 25,000. See "Entry cache" on page 13 for more information about the entry cache.
   - **Maximum number of elements in search filter cache** (filter cache size) — The search filter cache consists of the requested search filters and resulting entry identifiers that matched. On an update operation, all filter cache entries are invalidated. The default is 25,000. See "LDAP filter cache" on page 11 for more information about the filter cache.
   - **Maximum number of elements from a single search added to search filter cache** (filter cache bypass limit) — If you select **Elements**, you must enter a number. The default is 100. Otherwise select **Unlimited**. Search filters that match more entries than the number specified here are not added to the search filter cache. See "Filter cache bypass limits" on page 13 for more information about bypass limits.
4. When you are finished, click **OK** to apply your changes, or click **Cancel** to exit the panel without making any changes.

Using the command line

To set LDAP configuration variables using the command line, issue the following command:

```
ldapmodify -DAdminDN -wAdminpassword  -ifilename
```

where the file *filename* contains:

```
dn: cn=Directory,cn=RDBM Backends,cn=IBM Directory,
  cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdDbConnections
ibm-slapdDbConnections:  15

dn: cn=Front End, cn=Configuration
changetype: modify
replace: ibm-slapdACLCache
ibm-slapdACLCache: TRUE
-
replace: ibm-slapdACLCacheSize
ibm-slapdACLCacheSize: 25000
-
replace: ibm-slapdEntryCacheSize
ibm-slapdEntryCacheSize: 25000
-
replace: ibm-slapdFilterCacheSize
ibm-slapdFilterCacheSize: 25000
```

```
-
replace: ibm-slapdFilterCacheBypassLimit
ibm-slapdFilterCacheBypassLimit: 100
```

Additional settings

There are several additional settings that affect performance by putting limits on client activity, minimizing the impact to server throughput and resource usage, such as:

- ibm-slapdSizeLimit: 500
- ibm-slapdTimeLimit: 900
- ibm-slapdIdleTimeOut: 300
- ibm-slapdMaxEventsPerConnection: 100
- ibm-slapdMaxEventsTotal: 0
- ibm-slapdMaxNumOfTransactions: 20
- ibm-slapdMaxOpPerTransaction: 5
- ibm-slapdMaxTimeLimitOfTransactions: 300
- ibm-slapdPagedResAllowNonAdmin: TRUE
- ibm-slapdPagedResLmt: 3
- ibm-slapdSortKeyLimit: 3
- ibm-slapdSortSrchAllowNonAdmin: TRUE

For more information about these settings, see "Appendix R. IBM Tivoli Directory Server configuration schema" in *IBM Tivoli Directory Server Version 6.3 Installation and Configuration Guide*.

Note: Default values are shown.

The IBM Tivoli Directory Server response time for searches with alias dereferencing option set to **always** or **searching** is significantly greater than that of searches with the dereferencing option set to **never**. A server-side configuration option ibm-slapdDerefAliases under dn: cn=Configuration can be used to override the dereference option specified in the client search requests. The allowed values are:

- **never**
- **find**
- **search**
- **always**

By setting the value to **never**, the server does not attempt to dereference possible aliases, and the response time for searches improves.

Setting SLAPD_OCHSELECT_USECS

An administrator can set the environment variable, SLAPD_OCHSELECT_USECS, to invoke the OCH select() call with a timeout value, where the timeout value is of microsecond granularity. If the SLAPD_OCHSELECT_USECS variable is not present, or is present and the value is zero, the OCH select() invocation will continue with an indefinite timeout value. If SLAPD_OCHSELECT_USECS is set to a positive integer value, the OCH select() invocation occurs with the set timeout value, which is considered in microseconds.

You can set the environment variable SLAPD_OCHSELECT_USECS or modify the value of the variable by altering or adding the **ibm-slapdSetenv** attribute under the "cn=Front End, cn=Configuration" entry of the ibmslapd.conf file. For example, to set the value of SLAPD_OCHSELECT_USECS to 1000 microseconds, issue the ldapmodify command in the following format:

```
idsldapmodify -D <adminDN> -w <adminPW>
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetEnv
ibm-slapdSetenv: SLAPD_OCHSELECT_USECS=1000
```

You must restart the LDAP server to effect the changes made. On restarting the LDAP server, SLAPD_OCHSELECT_USECS is configured with a value of 1000 microseconds (or one millisecond).

Directory size

It is important when you run your workload that you consider several measurements. For example, measuring the number of operations per second as shown in Figure 5, it appears that performance degrades significantly as the database size grows.

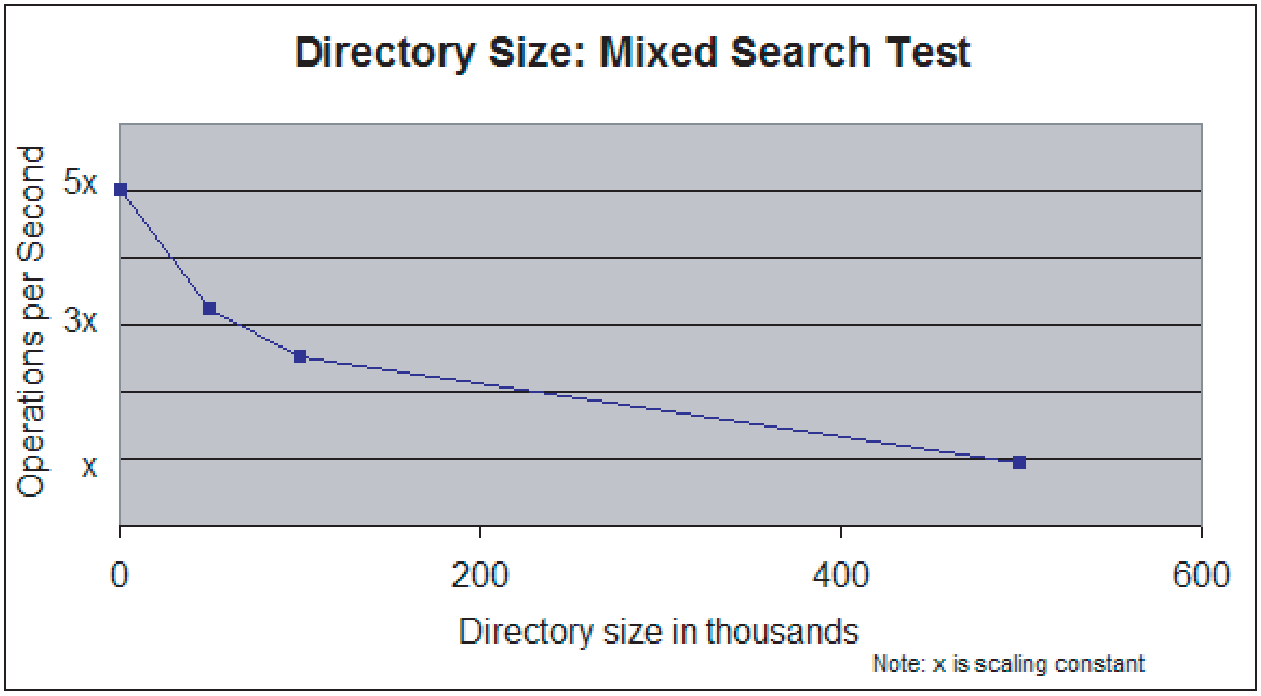


Figure 5. Operations per second

However, the benchmark tool test includes a large fraction of wildcard searches and exact-match searches, such as "(sn=Smith)" that return all entries where the sn value is "Smith". Both of these types of searches typically return multiple entries in response to a single search request. As Figure 6 on page 21 shows, as the size of the directory grows, so does the number of entries returned in response to wildcard and exact-match search requests.

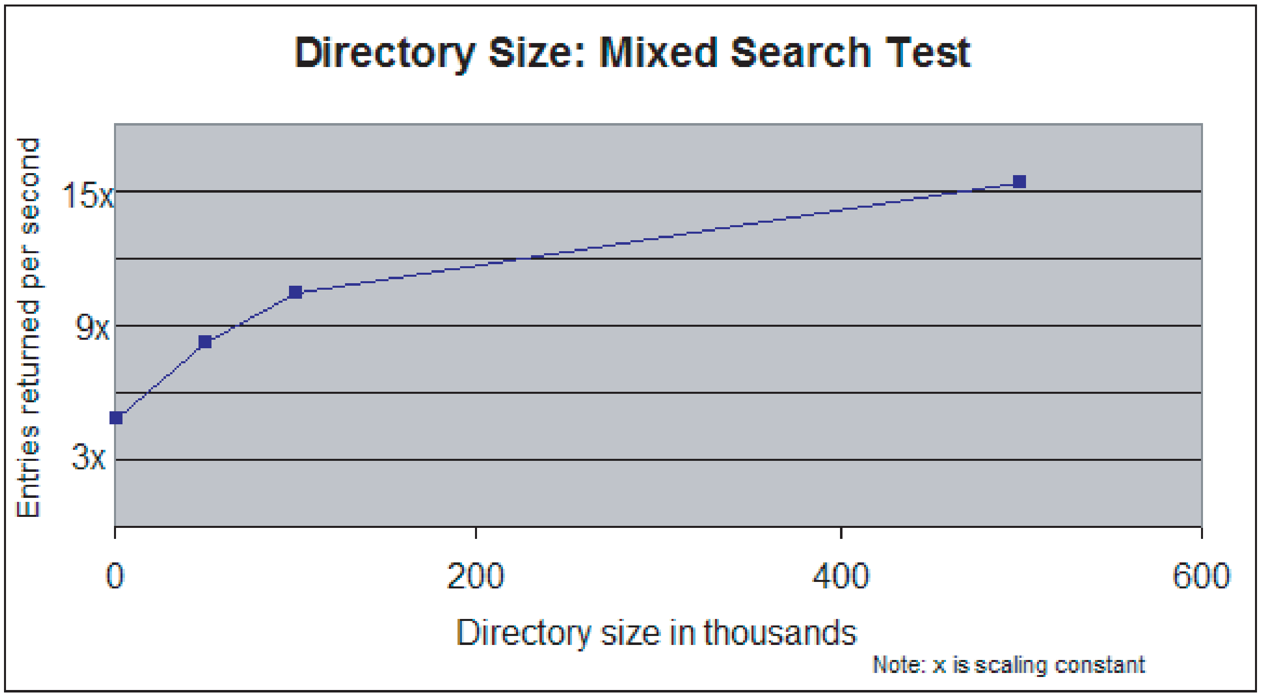


Figure 6. Entries returned per second

In this situation, the number of entries returned per second is a truer measure of throughput than operations per second, because each operation requires more work to be performed as the size of the database grows.

Note: As your directory grows, it might become necessary to readjust the sizes of the LDAP caches. You can determine the optimal sizes for your caches and buffer pools using the guidelines in "LDAP caches" on page 7 and "Tuning DB2 buffer pool" on page 24. The DB2 buffer pool tuning is performed automatically in Tivoli Directory Server V6.2 and above versions.

Chapter 3. Tuning DB2 and LDAP caches

IBM Tivoli Directory Server uses DB2 as the data store and Structured Query Language (SQL) as the query retrieval mechanism. While the LDAP server caches LDAP queries, answers, and authentication information, DB2 caches tables, indexes, and statements.

Many DB2 configuration parameters affect either the memory (buffer pools) 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. In DB2 v9.0, the the Self Tuning Memory Manager (STMM) was introducted. Users can use STMM instead of manually tuning several DB2 parameters. When the STMM is used, DB2 will assign the correct values to memory consumers based on the usage of the system and available resources. DB2 STMM can be used by setting the values of DB2 buffer pool to AUTOMATIC.

This chapter covers the following areas:
- DB2 buffer pool tuning
- Tuning DB2 and LDAP caches using the **idsperftune** tool
- Database maintenance using the **idsdbmaint** tool. This covers DB2 index organization, DB2 row compression, and tablespace conversion.
- Database maintenance using the **idsrunstats** tool.
- Optimization and organization (**reorgchk** and **reorg**)
- Other DB2 configuration parameters
- Backing up and restoring the database (**backup** and **restore**)
- Data row compression feature

For detailed information about DB2 commands, see the DB2 9.7 Information Center at the following Web site: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp.

Attention: Only users listed as database administrators can run the DB2 commands. Be sure the user ID running the DB2 commands is a user in the dbsysadm group (UNIX operating systems) or a member of the Administrator group (Windows operating systems.) This includes the DB2 instance owner and root.

If you have any trouble running the DB2 commands, check to ensure that the 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.) It is 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 instance owner as follows:

```
su - instance_owner
```

where *instance_owner* is the defined owner of the LDAP database.

To log on as the database administrator on a Windows 2000 operating system, run the following command:

```
runas /user:instance_owner db2cmd
```

where *instance_owner* is the defined owner of the LDAP database.

Tuning DB2 buffer pool

DB2 buffer pool tuning is one of the most significant types of DB2 performance tuning. A buffer pool is a data cache between LDAP and the physical DB2 database files for both tables and indexes. DB2 buffer pools are searched when entries and their attributes are not found in the entry cache. Typically, buffer pool tuning is needed to be done when the database is initially loaded and when the database size changes significantly. Disabling file system caching is recommended when buffer pools are used. Doing so will improve performance of utilities like bulkload, by removing a redundant level of caching.

There are several considerations to keep in mind related to DB2 buffer pools; for example:

- If there are no buffer pools, all database activity results in disk access.
- If the size of each buffer pool is too small, LDAP must wait for DB2 disk activity to satisfy DB2 SQL requests.
- If one or more buffer pools is too large, memory on the LDAP server might be wasted.
- If the total amount of space used by the LDAP caches and both buffer pools is larger than physical memory available on the server, operating system paging (disk activity) will occur.
- Most importantly, current versions of DB2 support automatic tuning of the buffer pools.

The Tivoli Directory Server performance tuning tool (idsperftune) will set DB2 configuration options so that automatic tuning of the buffer pools is done.

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

```
db2 connect to database_name
db2 select varchar(bpname,20) as bpname,npages,pagesize from  syscat.bufferpools
```

where *database_name* is the name of the database.

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.
```

To determine the current file system caching option for each of the tablespace, execute the following commands:

```
db2 get snapshot for tablespaces on ldapdb2 | egrep 'tablespace name|File system caching'
```

To turn off file system caching with DB2 version 8.2 or later and with operating systems and file system environments that support it, use the following command:

```
db2 connect to  ldapdb2
db2 alter  tablespace USERSPACE1  no file system  caching
db2 alter  tablespace LDAPSPACE  no file system  caching
db2 terminate
db2stop
db2start
```

To set the buffer pool sizes, use the following commands:

```
db2 alter bufferpool ibmdefaultbp size  <new size in 4096 byte pages>
db2 alter  bufferpool ldapbp   size <new size in 32768 byte pages>
db2 terminate
db2stop
db2start
```

If these commands are executed while the directory server is running, the db2stop command will fail with a message indicating there are still applications connected to the database. If this occurs, stop the directory server and then run the following commands:

```
db2stop
db2start
```

DB2 buffer pool analysis

To analyze the performance of DB2 buffer pool perform the following procedure.

1. Turn on buffer pool monitoring.
   ```
   db2 update database manager configuration using DFT_MON_BUFPOOL ON
   db2stop
   db2start
   ```
2. Connect to the database.
   ```
   db2 connect to  ldapdb2
   ```
   where, ldapdb2 is the database instance.
3. Start the workload to be analyzed.
4. Reset the monitor data.
   ```
   db2 reset  monitor all
   ```
5. Obtain the statistics with the workload in progress.
6. Take a snapshot of the buffer pool statistics and process the output. The following command takes a snapshot of the current buffer pool statistics:
   ```
   db2 get snapshot for bufferpools on ldapdb2
   ```
   The following command takes a snapshot and reports the read times:
   ```
   db2 get   snapshot for bufferpools on idsdb | awk '{
   if($1=="Bufferpool"&&$2=="name"){print  $0}
   if (index($0,"pool read   time")){print "\t"$0}
   }'
   ```
   The following command takes a snapshot and reports the number of logical and physical reads:
   ```
   db2 get   snapshot for bufferpools on idsdb | awk '{
   if($1=="Bufferpool"&&$2=="name"){print  $0}
   if (index($0,"cal reads")){print "\t"$0}
   }'  | grep  -v  temp
   ```
   The following command takes a snapshot and reports miss ratios:

```
db2 get   snapshot for bufferpools on idsdb | grep  -v  temporary | awk '{
if($1=="Bufferpool"&&$2=="name"){print  $0}
if (index($0,"logical  reads")){l=$NF;getline;p=$NF;
if (l==0){r=0}else{r=p/l};print "\tMiss ratio: "$3"  "r}
}'
```

7. Tune the buffer pool sizes such that the IBMDEFAULTBP has a very low read time, a low number of reads, and a low miss ratio, but do not exceed system physical memory size or reduce the LDAPBP size to less than 2075 pages of 32 KB.

   A higher miss ratio means there are a higher number of physical reads and a lower number of cache hits. Allocate the remaining physical memory to the LDAPBP as follows:

   With file system caching turned off
   LDAPBP size = (<total physical memory> - 1 GB (for IDS and DB2) - (IBMDEFAULTBP size) * 4096) / 32768

   With file system caching turned on
   LDAPBP size = (<total physical memory> - 1.75 GB (for IDS, DB2 and file system caching) - (IBMDEFAULTBP size) * 4096) / 32768

If any of the buffer pool sizes are set too high, DB2 will fail to start due to insufficient memory. If this occurs there might be a core dump file, but usually there are no error messages. On AIX systems, the system error log might report a memory allocation failure. To view this log, enter the following:

```
errpt –a  | more
```

If DB2 fails to start due to buffer pool sizes being too large, set the buffer pool sizes to lower values and restart DB2. Also, restoring a database that was backed up on a system with buffer pool sizes that are too large for the target system will cause the restoration to fail.

Notes:

1. If you have problems connecting to the database on Windows systems, check the DB2INSTANCE environment variable. By default this variable is set to `DB2`. However, to connect to the database, the environment variable must be set to the database instance name. For additional stability and performance enhancements, upgrade to the latest version of DB2.
2. In DB2 version 9.x, the self_tuning_mem database configuration parameter is automatically set to ON when you create a single-partition database and sets the following values to AUTOMATIC. By doing so, the following memory consumers can be enabled for self tuning:
   - Buffer pools (controlled by the ALTER BUFFERPOOL and CREATE BUFFERPOOL statements)
   - Package cache (controlled by the pckcachesz configuration parameter)
   - Locking memory (controlled by the locklist and maxlocks configuration parameters)
   - Sort memory (controlled by the sheapthres_shr and the sortheap configuration parameter)
   - Database shared memory (controlled by the database_memory configuration parameter)

   To limit DB2 buffer pools in using all the available memory, user must perform the following steps before enabling the STMM.
   - Allow the setting for database shared memory size configuration parameter, DATABASE_MEMORY, to use the default automatic values.

```
db2 ALTER BUFFERPOOL LDAPBP SIZE AUTOMATIC
db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE AUTOMATIC
```

- Run the instance under normal load and monitor the value of DATABASE_MEMORY to determine an optimum size for the setting.
- Set DATABASE_MEMORY to the determined size instead of automatic.

```
db2 ALTER BUFFERPOOL LDAPBP SIZE <Determined_Value>
db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE <Determined_Value>
```

This will optimize performance by stabilizing the setting to a fixed value.

Tuning DB2 transaction log size

The space required by DB2 transaction log is specified by the following DB2 parameters:

- LOGFILSIZ
- LOGPRIMARY
- LOGSECOND
- NEWLOGPATH

To view the DB2 parameters associated with DB2 transaction log and their values, run the following command:

```
db2 get database configuration for ldapdb2 | \
                 egrep 'LOGFILSIZ|LOGPRIMARY|LOGSECOND|NEWLOGPATH|Path to log  files'
 Log file size (4KB)                         (LOGFILSIZ) = 2000
 Number of primary log files                (LOGPRIMARY) = 8
 Number of secondary log files               (LOGSECOND) = 3
 Changed path to log files                  (NEWLOGPATH) =
```

Tivoli Directory Server uses transaction log disk space for storing uncommitted DB2 transactions from directory update operations. The transaction log parameters must be tuned to allow the transaction logs to grow to their maximum required size. The transaction log size is limited by the values of DB2 parameters LOGFILSIZ, LOGPRIMARY, and LOGSECOND, and also by the available disk space in the directory specified by the NEWLOGPATH DB2 parameter. If the transaction logs exceeds the limit due to the settings of the DB2 size parameter, then the transaction is backed out using the information in the transaction logs and the transaction fails. If the transaction logs exceed the limit due to a lack of available disk space, the database becomes corrupted and goes into an unusable state. If the database becomes corrupted in this way, it is possible to issue DB2 commands to recover the database. Alternatively, the database can be restored from a backup or reloaded. If the database becomes corrupted, often the recovery commands can be found in the sqllib/db2dump/db2diag.log file, which is located in the DB2 instance owner's home directory. By default the DB2 transaction log file size (LOGFILSIZ) is defined to be 2000 blocks of 4 KB in size or 8000 KB per log file. The number of primary logs files (LOGPRIMARY) is defined as 8 and the number of secondary log files (LOGSECOND) is 3. In order to increase the DB2 transaction log limits to allow for millions of users, it is necessary to increase the size of the transaction logs (LOGFILSIZ) and increase the number of secondary files (LOGSECOND). It is better to increase the number of secondary files rather than the number of primary files, because the secondary files periodically get deleted when not in use.

The transaction log requirements are small for a Tivoli Directory Server running with a normal workload. It is observed that runtime directory operations will increase the transaction log requirements for a short period of time.

- The ldapadd or ldapmodify commands use some amount of transaction log space as the number of multi-valued attributes added to a single LDAP entry in a single command grows. For example, when loading many members into a group.
- An ACL placed on a suffix object can result in that ACL propagating to every entry under that suffix. The Tivoli Directory Server performs ACL propagation as one single committed DB2 transaction.

Tuning database connections

The default number of connections that get created between Tivoli Directory Server and DB2 is 15, which suffices for most environment in which directory server is used. However based on the requirement, user can increase the database connection to unlimited (upper limit INT_MAX - 2147483647) by modifying the attribute ibm-slapdDbConnections. If a value of less than or equal to zero, or greater than INT_MAX is specified, then the default value of 15 will be considered.

Depending on the server load and the nature of connections, performance might improve will the increase in the number of back-end connections. The best way to make this decision is by determining the result of the monitor search and look for available_workers threads in the output. Database connection is basically the number of worker threads, so to increase the workers threads or to increase the backend connections, user must set the attribute ibm-slapdDbConnections under the DN entry cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration. To view the monitor search result, run the idsldapsearch command of the following format:

```
idsldapsearch -p <port> -D <adminDN> -w <adminPwd> -s base -b "cn=monitor" objectclass=* \
  | grep -i available_workers
```

The performance tuning tool (idsperftune)

The IBM Tivoli Directory sever provides a tool named **idsperftune** (the Performance Tuning tool) that enables administrators to achieve higher directory server performance by tuning directory caches, DB2 buffer pools, and DB2 parameters. The idsperftune tool must be run against a directory server instance with RDBM configured. If the tool is run against a proxy server instance, the tool will provide an appropriate error message and will exit.

The idsperftune tool runs based on the inputs it received from an administrator. If the inputs are not provided by the administrator, then the default values are considered. The inputs from the administrator are provided to the tool using the property file, perftune_input.conf. For information about configuring and running the Performance Tuning Tool using the Configuration Tool, see the *IBM Tivoli Directory Server Version 6.3 Installation and Configuration Guide*.

The **idsperftune** tool works in two modes: basic and advanced.

Basic tuning

The basic tuning mode of operation deals with the tuning of the following:

- LDAP caches: These include entry cache, filter cache, group member cache, and group member cache bypass limit.
- DB2 buffer pools: These include IBMDEFAULTBP and LDAPBP.

The basic tuning mode recommends optimum tuning values for LDAP caches and DB2 buffer pools and optionally updates the LDAP cache and DB2 buffer pool parameters to the recommended settings. These recommendations are based on the following inputs:

- Amount of free system memory (%) to be allotted to Tivoli Directory Server instance:

  This is the total memory that will be allocated to an instance and will be used as an input to the tool while tuning the size of entry cache, filter cache, and group member cache. If not specified, then the default value of 90% of system memory available at the time idsperftune is run will be taken.
- The number of entries and the average size of an entry that is in a directory server instance
  - Total number of entries that will reside in the directory:

    This value is used as an input to the tool to estimate the size that should be assigned to the cache.
  - Average size of entry (Bytes):

    This value represents the average size of an entry that is expected to reside in memory. The average size of an entry and the total number of entries is used by the **idsperftune** tool to calculate the total size of the directory. Based on this, the size that should be allotted to Entry and Filter cache is calculated.

  Note: The **idsperftune** tool provides a command line option to calculate the total number of entries and average size of entries that are present in the directory. For example, if an administrator provides the command line option "-s " then **idsperftune** will compute total number of entries and average size of entries and log the details in the perftune_input.conf file. If the administrator does not provide the command line option, then the total number of entries is set defaulted 10000. For further information about the **idsperftune** tool refer *IBM Tivoli Directory Server Version 6.3 Command Reference*.
- Update Frequency:

  The administrator must specify whether frequent updates, or only batch updates, are expected. If the administrator specifies that frequent updates are expected, then the filter cache is set to 0. Otherwise, it is set to 1 KB.
- Total number of groups to be cached:

  An administrator can tune this value by providing an estimate of the total number of groups whose members need to be cached. This should be the number of groups frequently used. If not specified, the default value of 25 is used.
- Average number of members in a group:

  An administrator can tune this value to set the total number of members within a group that will be cached. If not specified, the default value of 25000 is used.
- Server instance name:

  This value is taken from IDS_LDAP_INSTANCE environment variable. If this environment variable is not set then the server instance name is set to the name of the directory server instance that is present. However, if more then one instance is present and no instance name is provided by the administrator, then an appropriate error message is displayed.

The DB2 buffer pools IBMDEFAULTBP and LDAPBP are set to AUTOMATIC when the idsperftune tool is allowed to update the configuration settings (-u option). This enables the DB2 self tuning memory manager to dynamically adjust the sizes of the DB2 buffer pools. When the **idsperftune** tool is run, the size that is to be allotted to LDAP entry cache is calculated. If the idsperftune tool is run in basic mode by providing the –B and –u options, then DB2 buffer pools will be set to AUTOMATIC.

If the system memory available to the directory server is sufficient to cache at least 80% of directory entries then the LDAP entry cache size is set to the size required to cache 80% or more of total entries. To override the default requirement for at least 80% of entries that should reside in LDAP entry cache, run the idsperftune tool with the -E option and the target percentage of entries to be cached. For example, to cache a minimum of 50% of entries in the entry cache run the following command:

```
idsperftune -I <instance_name> -E 50
```

If the system memory allotted to the directory server is not enough to cache 80% of total entries present in the directory then the entry cache is set to a minimum value which is 1000.

SYS_MEM_AVL

If the variable SYS_MEM_AVL in the perftune statistics file is set to TRUE, then 80% of the directory entries will be cached. If the SYS_MEM_AVL variable is set to FALSE, a minimum amount of system memory is allotted to LDAP entry cache and the remaining is allotted to DB2 buffer pools.

Examples

Using the idsperftune command for basic tuning:

- To get basic tuning recommendations, run the idsperftune command with the following arguments:

  ```
  idsperftune –I <instance_name> -B
  ```

- To update the database with the suggested parameters during basic tuning, run the idsperftune command with the following arguments:

  ```
  idsperftune –I <instance_name> -B –u
  ```

  or

  ```
  idsperftune –I <instance_name> –u
  ```

- To update the admin input file with the total number of entries and average entry size, run the idsperftune tool with the following arguments:

  ```
  idsperftune -I <instance_name> -s
  ```

Advanced tuning

For this phase of tuning, the directory server should be deployed and populated with entries. Ideally, the directory server should be servicing client requests for some period of time but the idsperftune tool can be run against a directory server instance on which no operations have been performed to get recommendations for basic or advance tuning mode. In advanced tuning mode, setting the DB2 monitor switches to ON initiates data collection that enable better tuning of DB2. The DB2 monitor switches can be set to ON by running the idsperftune command with the -m option. If a user wants to perform tuning based on a typical workload, then the idsperftune command should be run with the -A and -m options, this will set the DB2 monitor switches to ON and then waits for some minutes before collecting the statistics. Tuning will be based on whatever workload ran prior to collecting the DB2 monitor information. A user also has option to set DB2 monitor switches to OFF by running the idsperftune command with the -o option. Administrators can run **idsperftune** with appropriate advanced tuning option to monitor different DB2 parameters. For further information about the **idsperftune** tool refer to *IBM Tivoli Directory Server Version 6.3 Command Reference*.

The DB2 parameters that can be monitored for tuning at run time are as follows:

- PCKCACHESZ: This parameter is allocated out of the database shared memory and is used for caching of sections for static and dynamic SQL and XQuery statements on a database.
- LOGFILSIZ: This parameter defines the size of each primary and secondary log file. The size of these log files limit the number of log records that can be written to them before they become full and a new log file is required.
- LOGBUFSZ: This parameter allows to specify the amount of the database heap (defined by the dbheap parameter) to use as a buffer for log records before writing these records to disk.
- SORTHEAP: This parameter defines the maximum number of private memory pages to be used for private sorts or the maximum number of shared memory pages to be used for shared sorts.
- MAXFILOP: This parameter specifies the maximum number of file handles that can be opened for each database agent.
- DBHEAP: This parameter specifies the maximum memory used by the database heap.
- CHNGPGS_THRESH: This parameter specifies the level (percentage) of changed pages at which the asynchronous page cleaners will be started, if they are not currently active.
- NEWLOGPATH: This parameter specifies the location where the log files are stored, a string value of up to 242 bytes.

The idsperftune tool checks if self tuning memory is enabled. If the idsperftune tool is run in advanced mode by providing the –A and –u options, then self tuning memory will be enabled if the tool has detected it to be in disabled state. Self tuning memory helps in memory configuration by setting values for memory configuration parameters automatically and sizing buffer pools. When enabled, the memory tuner dynamically distributes available memory resources among several memory consumers including the sort, package cache, lock list areas, and buffer pools.

The idsperftune tool checks if DB2 AUTOMATIC variables such as SELF_TUNING_MEM, AUTO_MAINT, AUTO_TBL_MAINT, and AUTO_RUNSTATS are set to ON. If the idsperftune tool is run in advanced mode by providing the –A and –u options, then these AUTOMATIC variables will be automatically set to ON, if the tool has detected the variables are in OFF state. AUTO_TBL_MAINT is the key parameter for other table maintenance related parameters (AUTO_RUNSTATS, AUTO_STATS_PROF, AUTO_PROF_UPD, and AUTO_REORG). Additionally, to enable AUTO_MAINT, the AUTO_TBL_MAINT parameter must be turned ON.

The idsperftune tool also checks for DB2 AUTOMATIC variables such as LOCKLIST, NUM_IOSERVERS, NUM_IOCLEANERS are set to AUTOMATIC. If the idsperftune tool is run in advanced mode by providing the –A and –u options, then these variables will be set to AUTOMATIC.

Table 1. DB2 9 parameters with their type and values

DB2 9 parameters	Type	Value in the stats file
NUM_IOCLEANERS	AUTOMATIC	AUTOMATIC / Numeric value
NUM_IOSERVERS	AUTOMATIC	AUTOMATIC / Numeric value
LOCKLIST	AUTOMATIC	AUTOMATIC / Numeric value
SELF_TUNING_MEM	AUTOMATIC	ON / OFF
AUTO_MAINT	AUTOMATIC	ON / OFF
AUTO_RUNSTATS	AUTOMATIC	ON / OFF
AUTO_TBL_MAINT	AUTOMATIC	ON / OFF
IBMDEFAULTBP	AUTOMATIC	AUTOMATIC
LDAPBP	AUTOMATIC	AUTOMATIC

To monitor DB2 parameters SORTHEAP, MAXFILOP, DBHEAP, CHNGPGS_THRESH, NUM_IOSERVERS, and NUM_IOCLEANERS, monitor switches BUFFERPOOL and SORTHEAP must be enabled. To enable the monitor switches, run the idsperftune tool with -m option. These switches allow DB2 to collect additional runtime data. However, enabling these monitor switches will have some negative impact on the performance of the directory server. If monitor switches BUFFERPOOL and SORTHEAP are not enabled, then the status of these parameters is displayed as “Not Collected” in the perftune_stat.log file. If monitor switches are enabled, then the suggested values will be updated for the parameters in the log file.

When the idsperftune tool updates directory server and DB2 configuration parameters it does keep some history of the previous values in the perftune_stat.log file. The initial values that existed before the first time that the tool made updates are recorded under the INITIAL TUNING PARAMETER VALUE section with the prefix "I_", for example, I_MAXFILOP. In addition, every time that the tool makes updates to any DB2 configuration settings, the previous values are stored under the OLD DB2 PARAMETER VALUE section with the prefix "O_", for example, O_LOGFILSIZ.

The idsperftune tool provides recommendations for DB2 parameters in the following format:

```
<DB2 parameters>=<Current Value>:<Recommendation>
```

The recommendation can be one of the following:

```
<Not Collected>|<OK>|<Increase>|<Decrease>
```

For example,

```
PCKCACHESZ=1533:Increase
```

This helps users to know the current value and the action to be taken.

Description of DB2 parameter status as mentioned above:

- **Not Collected**: The value of DB2 parameter is not monitored, this state will be observed for the DB2 parameters which need monitor switches to be turned on.
- **OK**: The value currently used for the DB2 parameter is optimal.

- **Increase**: The value of DB2 parameter must be increased to achieve optimal performance.
- **Decrease**: The value of DB2 parameter must be decreased to achieve optimal performance.

Examples

Using the idsperftune command for advanced tuning:

- To get advance tuning recommendations without setting the monitor switches ON, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -A
  ```
- To set monitor switches for DB2 parameters to ON, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -m
  ```
- To set monitor switches for DB2 parameters to OFF, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -o
  ```
- To update the database with the suggested DB2 parameters in advanced tuning without setting the monitor switches to ON, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -A -u
  ```
- To get advance tuning recommendations with the monitor switches set to ON, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -A -m
  ```
  The monitor switches will be set to OFF once the tool completes its operation.
- To update the database with the suggested DB2 parameters in advanced tuning with the monitor switches set to ON, run the idsperftune command with the following arguments:
  ```
  idsperftune -I <instance_name> -A -u -m
  ```
  The monitor switches will be set to OFF once the tool completes its operation.

Perftune input file (perftune_input.conf)

The property file, perftune_input.conf, contains a list of inputs in the form attribute-value pairs. If a user does not want to specify values for attributes, then the user should leave the values as "None" for the attributes.

The administrator must update the attribute values as per their Tivoli Directory Server environment requirements, and then run the tool by providing the property file as an input. Administrator must ensure that the attribute names are not altered in the property file. If the attribute names are altered, the idsperftune tool will display appropriate error messages and will exit. On AIX, Linux, and Solaris systems, the location of the property file, perftune_input.conf, is <instance-home>/idsslapd-<inst-name>/etc. On Windows system, the location of the property file, perftune_input.conf, is <instance-location>\idsslapd-<inst-name>\etc. The format of a sample perftune_input.conf file is as follows:

```
#-----------------------------------------------------
#   Admin Input
#-----------------------------------------------------
# Amount of system memory (%) to be allotted to TDS instance
TDS_SYS_MEM=90
# Total number of entries that will reside in the directory
TDS_TOTAL_ENTRY=10000
# Average size of entry (Bytes)
```

```
TDS_AVG_ENTRY_SZ=2560
# Update Frequency
#    1. Frequent updates expected, or
#    2. Only Batch Updates  expected
TDS_UPDATE_FREQ=1
#Total number of Groups to be cached
TDS_GROUP_CACHE=25
# Maximum number of members in a group that will be referenced frequently
TDS_GROUP_MEMBER=25000
#
#-------------------------------------------------------
# DB2 PARAMETER INPUT
#-------------------------------------------------------
# NEWLOGPATH allows you to specify a string of up to 242 bytes to change
# the location where the log files are stored. Eg, NEWLOGPATH="/newdevice"
NEWLOGPATH=None
# LOGFILSIZ defines the size of each primary and secondary log file. The size
# of these log files limits the number of log records that can be written to
# them before they become full and a new log file is required.
LOGFILSIZ=None
# DBHEAP determines the maximum memory used by the database heap.
DBHEAP=None
# PCKCACHESZ is allocated out of the database shared memory, and is used for
# caching of sections for static and dynamic SQL and XQuery statements on
# a database.
PCKCACHESZ =None
# LOGBUFSZ allows you to specify the amount of the database heap (defined
# by the dbheap parameter) to use as a buffer for log records before writing
# these records to disk.
LOGBUFSZ=None
# MAXFILOP specifies the maximum number of file handles that can be open
# for each database agent.
MAXFILOP=None
# CHNGPGS_THRESH specifies the level (percentage) of changed pages at which
# the asynchronous page cleaners will be started, if they are not currently active.
CHNGPGS_THRESH=None
# SORTHEAP defines the maximum number of private memory pages to be used
# for private sorts, or the maximum number of shared memory pages to be
# used for shared sorts.
SORTHEAP=None
```

Perftune statistics file (perftune_stat.log)

Information gathered during the basic tuning and advanced tuning phases are logged in the property file, perftune_stat.log. The log contains the information like, if a particular DB2 parameter value needs to be increased or decreased in order to get better performance out of the directory server. On AIX, Linux, and Solaris systems, the location of the property file, perftune_stat.log, is <instance-home>/idsslapd-<inst-name>/logs. On Windows system, the location of the property file, perftune_stat.log, is <instance-location>\idsslapd-<inst-name>\logs. The perftune_stat.log file is taken from a directory server instance that was running on a Linux system, which is loaded with entries from the /opt/ibm/ldap/V6.3/examples/sample.ldif file. The format of a sample perftune_stat.log file is as follows:

```
#------------------------------------------------------
# Perftune Basic tuning parameters
#------------------------------------------------------
#------------------------------------------------------
# Directory Cache
#------------------------------------------------------
TDS_ENTRY_CACHE=1000
TDS_FILTER_CACHE=0
TDS_GROUP_CACHE=25
TDS_GROUP_MEMBER=25000
#-----------------------------------------------------
```

```
# DB2 BUFFERPOOL (Number of pages)
#-----------------------------------------------------
IBMDEFAULTBP=AUTOMATIC
LDAPBP=AUTOMATIC
#-----------------------------------------------------------------
# System memory allotted to Directory Server Instance (Kilo Bytes)
#-----------------------------------------------------------------
SYSTEM_MEMORY=102417.12
#------------------------------------------------------
# Will be set to True if enough system memory is available to
# the directory instance to make directory caching effective
#------------------------------------------------------
SYS_MEM_AVL=FALSE
#------------------------------------------------------
# Perftune Advance tuning parameters
#------------------------------------------------------
# NEWLOGPATH allows you to specify a string of up to 242 bytes to
# change the location where the log files are stored.
NEWLOGPATH=None
#------------------------------------------------------------------
# DB2 PARAMETER STATUS
# <DB2 parameters>=<Current Value>:<Recommendation>
# Recommendation can be <Not Collected>/<OK>/<Increase>/<Decrease>
#------------------------------------------------------------------
# LOGFILSIZ defines the size of each primary and secondary log file.
# The size of these log files limits the number of log records that can
# be written to them before they become full and a new log file is required.
LOGFILSIZ=2000:OK
# PCKCACHESZ is allocated out of the database shared memory, and is used
# for caching of sections for static and dynamic SQL and XQuery statements
# on a database.
PCKCACHESZ=2299:Increase
# LOGBUFSZ allows you to specify the amount of the database heap
# (defined by the dbheap parameter) to use as a buffer for log records
# before writing these records to disk.
LOGBUFSZ=98:OK
# MAXFILOP specifies the maximum number of file handles that can be open
# for each database agent.
MAXFILOP=Not Collected
# CHNGPGS_THRESH specifies the level (percentage) of changed pages at
# which the asynchronous page cleaners
# will be started, if they are not currently active.
CHNGPGS_THRESH=Not Collected
# SORTHEAP defines the maximum number of private memory pages to be used for
# private sorts, or the maximum number of shared memory pages to be used for
# shared sorts.
SORTHEAP=Not Collected
#-----------------------------------------------------------
# DB2 parameters whose value will be automatically set by
# DB2 self tuning memory manager
# <DB2 parameters>=<Current Value>/<AUTOMATIC>
#-----------------------------------------------------------
# Indicates the amount of storage that is allocated to the lock list.
# There is one lock list per database and it contains the locks held by all
# applications concurrently connected to the database.
LOCKLIST=AUTOMATIC
# Number of I/O servers configuration parameter
NUM_IOSERVER=AUTOMATIC
# Number of asynchronous page cleaners configuration parameter
NUM_IOCLEANER=AUTOMATIC
#-------------------------------------------------------
# DB2 AUTOMATIC PARAMETERS
# <DB2 parameters>=<ON>/<OFF>
#-------------------------------------------------------
# SELF_TUNING_MEM determines whether the memory tuner will dynamically distribute
# available memory resources as required between memory consumers that
# are enabled for self tuning.
```

```
SELF_TUNING_MEM=ON
# AUTO_MAINT Automatic maintenance configuration parameter
AUTO_MAINT=ON
# This parameter is the parent of all table maintenance parameters
# (auto_runstats, auto_stats_prof, auto_prof_upd, and auto_reorg).
AUTO_TBL_MAINT=ON
# AUTO_RUNSTATS Automatic table maintenance configuration parameter
AUTO_RUNSTATS=ON
#--------------------------------------------------------
# OLD TDS CACHE PARAMETER ( Prior to last Update Operation )
#--------------------------------------------------------
O_TDS_ENTRY_CACHE=25000
O_TDS_FILTER_CACHE=25000
O_TDS_GROUP_CACHE=25
O_TDS_GROUP_MEMBER=25000
#--------------------------------------------------------
# OLD DB2 PARAMETER VALUE ( Prior to last Update Operation )
#--------------------------------------------------------
O_IBMDEFAULTBP=AUTOMATIC
O_LDAPBP=AUTOMATIC
O_PCKCACHESZ=2299
O_LOGBUFSZ=98
O_MAXFILOP=64
O_CHNGPGS_THRESH=80
O_SORTHEAP=355
O_DBHEAP=2333
O_NEWLOGPATH=None
O_LOGFILSIZ=2000
#--------------------------------------------------------
# INITIAL TUNING PARAMETER VALUE ( Prior to First Update Operation )
#--------------------------------------------------------
I_TDS_ENTRY_CACHE=25000
I_TDS_FILTER_CACHE=25000
I_TDS_GROUP_CACHE=25
I_TDS_GROUP_MEMBER=25000
I_IBMDEFAULTBP=AUTOMATIC
I_LDAPBP=AUTOMATIC
I_PCKCACHESZ=2299
I_LOGBUFSZ=98
I_MAXFILOP=64
I_CHNGPGS_THRESH=80
I_SORTHEAP=355
I_DBHEAP=2333
I_NEWLOGPATH=None
I_LOGFILSIZ=2000
```

The database maintenance tool (idsdbmaint)

The idsdbmaint tool is used for performing database maintenance activities such as DB2 index reorganization and DB2 row compression for a directory server instance. You must stop the directory server instance before running the idsdbmaint tool. This will ensure that the database remains in a consistent state after all the database maintenance activities are performed by the idsdbmaint tool.

Tablespaces

Tivoli Directory Server uses four DB2 tablespaces:

Tablespace 0: SYSCATSPACE
SYSCATSPACE is used to store a description of the database and its structure and contents. The disk requirements for this tablespace do not change with the size of the directory. The disk space requirements are covered by the default directory server disk requirements.

Tablespace 1: TEMPSPACE1

TEMPSPACE1 holds temporary data for sorting and collating DB2 results. The disk requirements for this tablespace grow at runtime if a complex search is performed on the directory server. The disk space requirements for this tablespace also grow with the usage of the bulkload utility.

The bulkload utility's disk requirements for this tablespace are a maximum of approximately 2 GB. The bulkload utility uses this maximum space when millions of entries are loaded into the directory server.

Tablespace 2: USERSPACE1
USERSPACE1 holds the portion of the database that contains the attribute tables and attribute table indexes for the directory server. These tables are used for optimizing searches on specific attributes.

Tablespace 3: LDAPSPACE

LDAPSPACE holds the portion of the database that contains the LDAP entry table and LDAP entry table indexes for the directory server. The LDAP entry table contains a few searchable attributes, such as Distinguished Name (DN), and a full, non-searchable definition of each LDAP entry.

The LDAP entry table is used to return the requested attributes from an LDAP search once the search has been narrowed down to a specific entry or set of entries.

In IBM Tivoli Directory Server v6.2 and later versions, users can select the type of tablespace to use with an option to choose either a system managed space (SMS) or database managed space (DMS) tablespace. When using DMS tablespaces, Tivoli Directory Server supports the use of raw devices. Along with file, a raw device can also be added to the containers in DB2 9.x. This presents an alternative to adding multiple physical disks to the containers for LDAP tablespaces (LDAPSPACE and USERSPACE1). To know more about tablespace, see *IBM Tivoli Directory Server Version 6.3 Administration Guide*.

A user can use the idsdbmaint tool to convert an SMS tablespace to a DMS tablespace and a DMS tablespace to an SMS tablespace. The tablespace conversion option is available only in the command line utility idsdbmaint tool and is not supported in the user interface of the Configuration tool, idsxcfg. The tablespaces LDAPSPACE and USERSPACE1 are considered by the idsdbmaint tool.

DB2 index reorganization

When DB2 index reorganization is performed on a table, the fragmented data is eliminated by reconstructing the index data. During the DB2 index reorganization the tool does the following:

- Queries DB2 sysibm.sysindexes on tables that belong to the directory server instance and then fetches all the tables for which indexes have been defined.
- Performs index reorganization on all the indexes.
- After index reorganization is performed on the table, statistics on the table are updated.

You can optimize the database by running the idsdbmaint command with the index reorganization option from the command prompt. For example:

```
idsdbmaint -I <instance_name> -i
```

DB2 row compression

DB2 row compression uses a static dictionary-based compression algorithm to compress data by row. This allows repeating patterns that span multiple column values within a row to be replaced with shorter symbol strings. The row compression on a table is performed by reconstructing the compression dictionary and compressing the information to eliminate fragmented data. Data compression reduces space required for the directory, reduces I/O and generally improves performance. The idsdbmaint tool performs the row compression in the following way:

- Queries DB2 syscat.tables and fetches all the tables that belong to the directory server instance.
- Inspects the table and fetches the row compression estimates for each table.
- If the compression estimate is more than thirty percent, then the tool does the following:
  - Alters the table to enable ROW COMPRESSION.
  - Runs the DB2 REORG command on the table and builds a new compression dictionary.
  - After running DB2 REORG on the table, all the statistics on the table are updated by running DB2 RUNSTATS on the table.
- Creates a new compression dictionary.

You can optimize the database by running the idsdbmaint command with the row compression option from the command prompt. For example:

```
idsdbmaint -I <instance_name> -r
```

Tablespace conversion

The idsdbmaint tool performs conversion of tablespaces type, from SMS to DMS and from DMS to SMS. The tablespaces LDAPSPACE and USERSPACE1 are considered by the idsdbmaint tool. When the idsdbmaint command is run with -t <ts_type>, it converts a tablespace type from SMS to DMS and from DMS to SMS respectively depending on the type tablespace that exists and the tablespace type to be converted. The valid values for the tablespace type are SMS and DMS. The idsdbmaint tool calculates the database size and determines the required disk space for the import and export of user's data. If the required disk space is not available then the tool displays an appropriate error message and exits.

The idsdbmaint tool performs tablespace conversion from an SMS tablespace to DMS tablespace in the following way:

- Exports all the data from the LDAPSPACE and USERSPACE1 tablespaces along with the table definitions.
- Drops the tablespace of type SMS.
- Creates the tablespace of type DMS.
- A REGULAR tablespace using a FILE container is used, which can perform auto-resize.
- Reconstructs all the tables within the tablespace and loads all the data into the database.

The idsdbmaint tool performs tablespace conversion from an DMS tablespace to SMS tablespace in the following way:

- Exports all the data from the LDAPSPACE and USERSPACE1 tablespaces along with the table definitions.

- Drops the tablespace of type DMS.
- Creates the tablespace of type SMS.
- A REGULAR tablespace using a PATH (directory) container is used, which can perform auto-resize.
- Reconstructs all the tables within the tablespace and loads all the data into the database.

Examples

Using the idsdbmaint command for tablespace conversion:

- To convert an SMS tablespace to a DMS tablespace and to store the exported data in a directory, mydata, run the following command:
```
idsdbmaint -I <instance_name> -t DMS -k
              <instance_location>/<instance_name>/mydata
```
- To specify a file container for LDAPSPACE tablespaces while converting from an SMS tablespace to a DMS tablespace and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t DMS -l
        /disk/32K_ldapspace_container/ldapspace -k /disk/mydata
```
- To specify a file container for USERSPACE1 tablespaces while converting from an SMS tablespace to a DMS tablespace and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t DMS -u
       /disk/container/userspace1 -k /disk/mydata
```
- To specify a file container for LDAPSPACE and USERSPACE1 tablespaces while converting from SMS to DMS and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t DMS
      -l /disk/32K_ldapspace_container/ldapspace
      -u /disk/container/userspace1 -k /disk/mydata
```
- To convert a DMS tablespace to an SMS tablespace and to store the exported data in a directory, mydata, run the following command:
```
idsdbmaint -I <instance_name> -t SMS -k
              <instance_location>/<instance_name>/mydata
```
- To specify a container path for LDAPSPACE tablespaces while converting from DMS to SMS and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t SMS
      -l /disk/32K_ldapspace_container/ -k /disk/mydata
```
- To specify a container path for USERSPACE1 tablespaces while converting from DMS to SMS and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t SMS
     -u /disk/userspace1_container/–k /disk/mydata
```
- To specify a file container for LDAPSPACE and USERSPACE1 tablespaces while converting from DMS to SMS and to store the exported data in a directory, run the idsdbmaint command with the following arguments:
```
idsdbmaint –I <instance_name> -t SMS
     -l /disk/32K_ldapspace_container/
     -u /disk/userspace1_container/ -k /disk/mydata
```

Note: The directory specified with the -k option must have read and write access for the DB2 instance owner.

Optimization and organization (idsrunstats, reorgchk and reorg)

DB2 uses a sophisticated set of algorithms to optimize the access to data stored in a database. These algorithms depend upon many factors, including the organization of the data in the database, and the distribution of that data in each table. Distribution of data is represented by a set of statistics maintained by the database manager.

In addition, IBM Tivoli Directory Server creates a number of indexes for tables in the database. These indexes are used to minimize the data accessed in order to locate a particular row in a table.

In a read-only environment, the distribution of the data changes very little. However, with updates and additions to the database, it is not uncommon for the distribution of the data to change significantly. Similarly, it is quite possible for data in tables to become ordered in an inefficient manner.

To remedy these situations, DB2 provides tools to help optimize the access to data by updating the statistics and to reorganize the data within the tables of the database.

Optimization

Optimizing the database updates statistics related to the data tables, which improves performance and query speed. Optimize the database periodically or after heavy database updates (for example, after importing database entries). The **Optimize database** task in the IBM Tivoli Directory Server Configuration Tool uses the **idsrunstats** command to update statistical information used by the query optimizer for all the LDAP tables.

Note: The **reorgchk** command also updates statistics. If you are planning to do a **reorgchk**, optimizing the database is unnecessary. See "Database organization (reorgchk and reorg)" on page 41 for more information about the **reorgchk** command.

To optimize the database using the Configuration Tool:

1. Start the Configuration Tool by typing **idsxcfg** on the command line.
2. Click **Optimize database** on the left side of the window.
3. On the Optimize database window, click **Optimize**.

After a message displays indicating the database was successfully optimized, you must restart the server for the changes to take effect.

The idsrunstats tool is used to update the DB2 system catalog statistics about the physical characteristics of tables and associated indexes in the database of a directory server instance. The physical characteristics of a table include number of records, number of pages, and average record length. To collect database statistics, the following flags are passed to the db2Runstats API, DB2RUNSTATS_ALL_COLUMNS|DB2RUNSTATS_ALL_INDEXES. The idsrunstats tool can be run against a directory server instance even when the instance is up and running.

To optimize the database using the command line, run the following command:

```
idsrunstats –I <instance_name>
```

where, <*instance_name*> is an optional parameter.

To know more about the usage of the idsrunstats or runstats command, see *IBM Tivoli Directory Server Version 6.3 Command Reference*.

Viewing DB2 system statistics settings

The db2look command can be used to see reports of all the system statistic settings in the database. Use the mimic option, -m, to produce a report that contains the DB2 commands that reproduce the current system statistic settings. Switch the user context as database instance owner before running the command.

```
db2look  -m -d  ldapdb2 -u ldapdb2 -o  output_file
```

where, *ldapdb2* is the database name, and *output_file* is the file name with location for storing the results.

Database organization (reorgchk and reorg)

Tuning the organization of the data in DB2 using the **reorgchk** and **reorg** commands might help to improve performance and might save disk space.

The **reorgchk** command updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables.

The **reorg** command may be used based on the recommendations from **reorgchk** to reorganize tablespaces to improve access performance and to reorganize indexes so that they are more efficiently clustered. The **reorgchk** and **reorg** commands can improve both search and update operation performance.

Performing a reorgchk

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

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

Where *ldapdb2* is the name of your database.

To generate a **reorgchk** output file (recommended if you plan to run the **reorg** command) add the name of the file to the end of the command, for example:

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

The following is a sample **reorgchk** report:

```
db2 => reorgchk current statistics on table all

Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
F3: 100 * NPAGES / FPAGES > 80

CREATOR   NAME                  CARD    OV    NP    FP    TSIZE  F1   F2 F3 REORG

--------------------------------------------------------------------------------

LDAPDB2   ACLPERM                  2     0     1     1      138   0    - 100 ---

LDAPDB2   ACLPROP                  2     0     1     1       40   0    - 100 ---

LDAPDB2   ALIASEDOBJECT            -     -     -     -        -   -    -   - ---
```

```
LDAPDB2   AUDIT                     1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITADD                  1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITBIND                 1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITDELETE               1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITEXTOPEVENT           1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITFAILEDOPONLY         1     0     1     1       18   0   - 100 ---

LDAPDB2   AUDITLOG                  1     0     1     1       77   0   - 100 ---

...

SYSIBM    SYSINDEXCOLUSE          480     0     6     6    22560   0 100 100 ---

SYSIBM    SYSINDEXES              216   114    14    28   162216  52 100  50 *-*

...

SYSIBM    SYSPLAN                  79     0     6     6    41554   0 100 100 ---

SYSIBM    SYSPLANAUTH             157     0     3     3     9106   0 100 100 ---

SYSIBM    SYSPLANDEP               35     0     1     2     5985   0 100  50 --*


--------------------------------------------------------------------------------


Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) / (NLEAF * INDEXPAGESIZE) > 50
F6: (100-PCTFREE) * (INDEXPAGESIZE-96) / (ISIZE+12) ** (NLEVELS-2) * (INDEXPAGES
IZE-96) / (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) < 100

CREATOR  NAME                 CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
--------------------------------------------------------------------------------

Table: LDAPDB2.ACLPERM
LDAPDB2  ACLPERM_INDEX           2     1     1     6      2  100    -    - ---
Table: LDAPDB2.ACLPROP
LDAPDB2  ACLPROP_INDEX           2     1     1     6      2  100    -    - ---
Table: LDAPDB2.ALIASEDOBJECT
LDAPDB2  ALIASEDOBJECT           -     -     -     -      -    -    -    - ---
LDAPDB2  ALIASEDOBJECTI          -     -     -     -      -    -    -    - ---
LDAPDB2  RALIASEDOBJECT          -     -     -     -      -    -    -    - ---
Table: LDAPDB2.AUDIT
LDAPDB2  AUDITI                  1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITADD
LDAPDB2  AUDITADDI               1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITBIND
LDAPDB2  AUDITBINDI              1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITDELETE
LDAPDB2  AUDITDELETEI            1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITEXTOPEVENT
...
Table: LDAPDB2.SN
LDAPDB2  RSN                 25012   148     2    14  25012   99   90    0 ---
LDAPDB2  SN                  25012   200     3    12  25012   99   61  119 --*
LDAPDB2  SNI                 25012    84     2     4  25012   99   87    1 ---
...
Table: LDAPDB2.TITLE
```

```
LDAPDB2  TITLEI                -     -     -     -      -    -    -    - ---
Table: LDAPDB2.UID
LDAPDB2  RUID              25013   243     3    17  25013    0   62   79 *--
LDAPDB2  UID               25013   273     3    17  25013  100   55   79 ---
LDAPDB2  UIDI              25013    84     2     4  25012  100   87    1 ---
Table: LDAPDB2.UNIQUEMEMBER
LDAPDB2  RUNIQUEMEMBER     10015   224     3    47  10015    1   60   44 *--
LDAPDB2  UNIQUEMEMBER      10015   284     3    47  10015  100   47   44 -*-
LDAPDB2  UNIQUEMEMBERI     10015    14     2     4      7  100   69    8 ---

...
Table: SYSIBM.SYSFUNCTIONS
SYSIBM   IBM127              141     1     1    13    141   65    -    - *--
SYSIBM   IBM25               141     2     2    34    141  100   72   60 ---
SYSIBM   IBM26               141     2     2    32    141   78   68   63 *--
SYSIBM   IBM27               141     1     1    23     68   80    -    - *--
SYSIBM   IBM28               141     1     1    12      2   99    -    - ---
SYSIBM   IBM29               141     1     1     4    141  100    -    - ---
SYSIBM   IBM30               141     3     2    59    141   78   76   38 *--
SYSIBM   IBM55               141     2     2    34    141   99   72   60 ---
...
-------------------------------------------------------------------------------


CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary
for indexes that are not in the same sequence as the base table. When multiple
indexes are defined on a table, one or more indexes may be flagged as needing
REORG.  Specify the most important index for REORG sequencing.
```

Using the statistics generated by **reorgchk**, run **reorg** to update database table organization. See "Performing a reorg."

Keep in mind that **reorgchk** needs to be run periodically. For example, **reorgchk** might need 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 63 for more information.

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

Because LDAP caches prepared DB2 statements, you must stop and restart **ibmslapd** for DB2 changes to take effect.

Performing a reorg

After you have generated organizational information about the database using **reorgchk**, the next step in reorganization is finding the tables and indexes that need reorganizing and attempting to reorganize them. Reorganizing a table can take a long time. The time it takes increases as the DB2 database size increases.

In general, reorganizing a table takes more time than updating statistics. Therefore, it is best to update statistics first and see if that sufficiently improves performance.

To reorganize database table information:

1. If you have not done so already, run **reorgchk**:
   ```
   db2 reorgchk update statistics on table all > reorgchk.out
   ```

The **reorgchk** update statistics report has two sections; the first section is the table information and the second section is the indexes. An asterisk in the last column indicates a need for reorganization.

2. 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.

   To reorganize a table to match the order of a particular index, use the following syntax:

   ```
   db2 reorg table table_name index index_name
   ```

   where,

   - *table_name* is the name of the table; for example, LDAPDB2.LDAP_ENTRY.
   - *index_name* is the name of the index; for example, LDAPDB2.SNI.

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

3. To reorganize the indexes with an asterisk in the last column:

   ```
   db2 reorg index index_name
   ```

Some guidelines for performing a reorganization are:

- 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, you can probably 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 the SYSIBM.SQL index if attempts to reorganize them seem to alternate between one or the other needing reorganization.
- When an attribute length is defined to be less than or equal to 240 bytes, the attribute table contains three columns: EID, attribute and reversed attribute columns. In this case, the forward index is created using the EID and attribute columns as index keys. For example, the attribute SN is defined to have the maximum length which is less than or equal to 240 bytes, so the attribute table contains the EID, SN and RSN columns and the following indexes are created for this attribute table:

  ```
  LDAPDB2.RSN <------  A reverse index whose defined index keys are the EID
   and RSN columns.
  LDAPDB2.SN  <------  A forward index whose defined index keys are the EID
   and SN columns.
  LDAPDB2.SNI <------  An update index whose defined index key is the EID column.
  ```

- Reorganize all the attribute tables 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.
- When an attribute length is defined to be greater than 240 bytes, the attribute table contains four columns: EID, attribute, truncated attribute and reversed truncated attribute columns. In this case, the forward index is created using the EID and truncated attribute columns as index keys. For example, the attribute CN is defined to have the maximum length which is greater than 240 bytes, so the attribute table contains the EID, CN, CN_T and RCN_T columns and the following indexes are created for this attribute table:

```
  LDAPDB2.RCN <------  A reverse index whose defined index keys are the EID
   and RCN_T columns.
  LDAPDB2.CN  <------  A forward index whose defined index keys are the EID
   and CN_T columns.
  LDAPDB2.CNI <------  An update index whose defined index key is the EID column.
```

The following is another example showing reverse, forward, and update indexes 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 indexes

DB2 indexes helps in improving the performance of search operations against Tivoli Directory Server on the attributes that are indexed. Indexing results in a considerable reduction in the amount of time it takes to locate requested data. DB2 indexes also improve the performance of directory operations such as start time, finding the subtree of a LDAP entry, and finding all children of an LDAP entry. For this reason, it can be very beneficial from a performance standpoint to index all relevant attributes used in searches. Tivoli Directory Server comes with set of attributes that are indexed by default.

The attributes that are indexed by default can be found in the directory schema files. An attribute with the keyword EQUALITY in the schema files indicates the attribute is indexed. Attributes can also be indexed issuing DB2 commands on the DB2 table that implement the attribute.

Use the following DB2 commands to verify that a particular index is defined. In the following example, the index being checked is for the attribute **seeAlso**:

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

where *database_name* is the name of your database.

If 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         SEEALSOI                       D                   1
LDAPDB2         SEEALSO                        D                   2
LDAPDB2         RSEEALSO                       D                   2

              3 record (s) selected.
```

To have IBM Tivoli Directory Server create an index for an attribute, do one of the following:

- To create an index using the Web Administration Tool:
  1. Expand **Schema management** in the navigation area, and click **Manage attributes**.
  2. Click **Edit attribute**.
  3. On the **IBM extensions** tab, select the **Equality** check box under **Indexing rules**.
- To create an index from the command line, issue the following command:

  ```
  ldapmodify -D cn=root -w root -i addindex.ldif
  ```

The addindex.ldif file should look like the following example for the "seeAlso" attribute:

```
dn: cn=schema
changetype: modify
replace: attributetypes
attributetypes: ( 2.5.4.34
 NAME 'seeAlso'
 DESC 'Identifies another directory server entry that may
  contain information related to this entry.'
 SUP 2.5.4.49
 EQUALITY 2.5.13.1
 USAGE userApplications )
-
replace: ibmattributetypes
ibmattributetypes: ( 2.5.4.34
 DBNAME( 'seeAlso'  'seeAlso' )
 ACCESS-CLASS normal
 LENGTH 1000
 EQUALITY )
```

It is important to run the runstats command on a table after an index has been created. The runstats command provides statistical information to the DB2 optimizer that aids the optimizer in making decisions about optimizing searches on that table.

To know more on indexing, see the section DB2 index reorganization.

DB2 SELECTIVITY

There are two alternative ways that Tivoli Directory Server influences the DB2 optimizer to make better choice about how to access data in the LDAP tables. These are controlled by environment variables, as follows:

LDAP_MAXCARD = YES | ONCE | NO

- If set to YES or by default, if neither environment variable is set, DB2's cardinality statistic for the LDAP_DESC table is artificially adjusted to prevent expensive scans of large subtree data. This setting of the cardinality is done at server startup and periodically thereafter.
- If set to ONCE, then the cardinality is set once during each server startup but not subsequently while the server is running.
- If set to NO, then the cardinality statistic is not changed.

IBMSLAPD_USE_SELECTIVITY = NO | YES

- If not set or set to NO, then selectivity is not used to influence DB2's access plans.
- If set to YES and LDAP_MAXCARD is not set to YES, then selectivity is used to influence DB2's decisions about accessing data during large subtree searches. The use of selectivity is explained further in the following paragraphs.

Note: If LDAP_MAXCARD is set to YES and IBMDSLAPD_USE_SELECTIVITY is set to YES, the server will log a message and SELECTIVITY will not be used.

You can improve the performance of subtree searches on search bases that are high in a directory tree by using SELECTIVITY in Structured Query Language (SQL). The inclusion of SELECTIVITY in SQL enables the DB2 optimizer in the formation of access plans to resolve the search request (identifying which tables to access first during searches). Determination of the entries that are high in the tree (having a large number of children) is based on DB2 statistics. If a subtree search is done using one of these entries as the search base, the SELECTIVITY clause will be added to the SQL query. This causes DB2 to use the search filter to narrow down the search results before reading the table that identifies the entries that are descendants of a base in a search.

To use SELECTIVITY, the DB2 registry for the database instance must have DB2_SELECTIVITY set to YES. This is in addition to the environment variables described above. This is done when creating a database instance or while migrating from previous versions.

Examples

To check the status of the DB2_SELECTIVITY for a directory server instance, myinst1:

```
su – myinst1
db2 connect to myinst1
db2set –all | grep –i selectivity
```

To set DB2_SELECTIVITY explicitly for the directory server instance, myinst1, issue the following command:

```
su – myinst1
db2 connect to myinst1
db2set DB2_SELECTIVITY=YES
```

Other DB2 configuration parameters

Performance benefits can come from setting other DB2 configuration parameters, such as DBHEAP and LOGFILSIZ. To set the DB2 configuration parameters use the following syntax:

```
db2 update database configuration for database name using \
parm name parm value
db2 force applications all
db2stop
db2start
```

where *database name* is the name of your database and where *parm name* is the parameter to change and *parm value* is the value it is to be assigned. The idsperftune tool also provides an interface using which the DB2 configuration parameters can be set. To update the DB2 configuration parameter with the values provided in perftune input file, perftune_input.conf, run the following command:

```
idsperftune -I <instance_name> -A -u update
```

To get DB2 configuration parameters and their values, you need to use the DB2 provided commands. The current setting of parameters can be obtained by issuing the following command:

```
db2 get database configuration for database name
```

where *database name* is the name of your database. For example, the following is the output showing the default settings after configuring the Tivoli Directory Server instance ldapdb2:

```
 db2 get database configuration for ldapdb2 | egrep  'HEAP|MAXLOCKS|MINCOMMIT'

 Percent. of lock lists per application       (MAXLOCKS) = AUTOMATIC(98)
 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(273)
 Sort list heap (4KB)                         (SORTHEAP) = AUTOMATIC(54)
 Database heap (4KB)                            (DBHEAP) = AUTOMATIC(2579)
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 85239
 SQL statement heap (4KB)                   (STMTHEAP) = AUTOMATIC(4096)
 Default application heap (4KB)           (APPLHEAPSZ) = AUTOMATIC(1280)
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = AUTOMATIC(4384)
 Group commit count                        (MINCOMMIT) = 1
```

These parameters do not required to be modified from their default settings. If you set MINCOMMIT to anything other than 1, you might get poor performance results. These parameters can be changed by using the commands of following format:

```
db2 update db cfg  for ldapdb2 using <parm_name> <parm_value>
db2 terminate
db2 force applications all
```

where *parm_name* is the name of the parameter shown in the output from the get database configuration command at the left hand side of the equals sign, and *parm_value* is the new value. Incorrect settings for some database parameters can cause database failures. If this is suspected, check the following files for DB2 error messages:

For Tivoli Directory Server 6.0 and later

- db2instance_owner_home_directory/idsslapd-instancename/logs/db2cli.log
- db2instance_owner_home_directory/sqllib/db2dump/db2diag.log

This command returns the settings of other DB2 configuration parameters as well. The following command also shows the DB2 configuration parameters for the entire database instance:

```
db2 get database manager configuration
```

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

Note: If applications are currently connected to the database, you must also do a **db2 force applications all** command prior to the db2stop.

For tuning DB2 buffer pools, users can use the DB2 utility, DB2 AUTOCONFIGURE. This utility calculates and provides recommended values for the DB2 buffer pools, database configuration, and database manager configuration parameters. For example, DB2 AUTOCONFIGURE with NONE as the argument displays the recommended changes in the configuration but does not apply them.

```
db2 AUTOCONFIGURE USING  MEM_PERCENT 60 WORKLOAD_TYPE simple  NUM_STMTS 500
ADMIN_PRIORITY performance  IS_POPULATED YES  NUM_LOCAL_APPS 20  NUM_REMOTE_APPS 20
ISOLATION RR  BP_RESIZEABLE YES APPLY NONE
```

The utility with DB AND DBM as the arguments display and apply the recommended changes to the buffer pool settings, database manager configuration, and the database configuration.

```
db2 AUTOCONFIGURE USING  MEM_PERCENT 60  WORKLOAD_TYPE simple  NUM_STMTS 500
ADMIN_PRIORITY performance  IS_POPULATED YES  NUM_LOCAL_APPS 20  NUM_REMOTE_APPS 20
ISOLATION RR  BP_RESIZEABLE YES APPLY DB AND DBM
```

To know more about the utility, see AUTOCONFIGURE command using the ADMIN_CMD.

For a list of DB2 parameters that affect performance, visit the DB2 9.7 Information Center at the following Web site: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp.

Note: If DB2 recognizes that a parameter is configured insufficiently, the problem is posted to the diagnostic log (db2diag.log). For example, if the DB2 buffer pools are too large, DB2 overrides the buffer pool settings and uses a minimal configuration. No notice of the change in buffer pool sizes is given except in the diagnostic log, so it is important to view the log if you are experiencing poor performance. The db2diag.log file is located in the sqllib/db2dump directory under the instance owner's home directory. For example, the ldapdb2 instance can find the db2diag.log file in the /home/ldapdb2/sqllib/db2dump directory.

Database backup and restore considerations

You can use the db2 backup and db2 restore commands that are provided by IBM DB2 for backing up and restoring db2 database associated with the Tivoli Directory Server. The advantage to using these commands is performance and flexibility for specifying the location of the database files. The db2 restore command can be used to distribute the database across multiple disks or to simply move the database to another directory. An important consideration when using db2 backup and db2 restore commands is the preservation of DB2 configuration parameters and system statistics optimizations in the backed-up database. The restored database has the same performance optimizations as the backed-up database. This is not the case with LDAP db2ldif, ldif2db, or bulkload.

However, when using the database backup and restore commands it is important to keep in mind that when 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, if you do not know whether a db2 runstats was performed before the database was backed up, tune the DB2 system statistics after the restore. The DB2 commands to perform backup and restore operations are as follows:

```
db2 force applications all
db2 backup  db dsrdbm01   to directory_or_device
db2 restore db dsrdbm01   from  directory_or_device replace existing
```

where, *dsrdbm01* is the name of Tivoli Directory Server instance, and *directory_or_device* is the name of a directory or device where the backup is stored.

When performing a db2 restore operation, user might see file permission error. The reason for the error and steps to prevent the error are as follows:

- The DB2 instance owner does not have permission to access the specified directory and file. One way to resolve this is to change directory and file ownership to the DB2 instance owner. For example, enter the following:

  ```
  chown dsrdbm01  file_or_device
  ```

- The backed-up database is distributed across multiple directories, and those directories do not exist on the target system of the restore.

  Distributing the database across multiple directories is accomplished with a redirected restore. To solve this problem, either create the same directories on the target system or perform a redirected restore to specify the proper directories on the new system. If creating the same directories, ensure that the owner of the directories is the DB2 instance owner, the dsrdbm01 user.

Chapter 4. AIX operating system tuning

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

- Enabling large files
- Setting MALLOCTYPE
- Setting other environment variables
- Viewing **ibmslapd** environment variables

Enabling large files

The standard file system on AIX has a 2 GB file size limit, regardless of the ulimit setting. 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 Enhanced Journaled File Systems or as Journaled File Systems (JFS2) with Large File Enabled. The file system containing the DB2 instance's home directory, and if **bulkload** is used the file system containing the **bulkload** temporary directory are file systems that can be created this way.

   Note: The default path is:

   ```
   <instance_home>/tmp
   ```

2. Set the soft file size limit for the root, ldap, and the DB2 instance owner 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. Each user must log off and log back in for the new soft file size limit to take effect. You must also restart DB2.

For additional information and file system options, see AIX documentation.

Setting MALLOCTYPE

Set the MALLOCTYPE environment variable as follows:

On all AIX 5.x versions

Set MALLOCTYPE as follows:

```
export MALLOCTYPE=buckets
```

Note: If you want to use MALLOCTYPE buckets, you must use ML03 (contains the fix for APAR IY50668) or higher. You can get this from IBM Support (www.ibm.com/support). If you are using MALLOCTYPE buckets, you must set ulimits for the LDAP instance to the following:

```
# ulimit -m unlimited
# ulimit -d unlimited
```

You can find more information about MALLOCTYPE in the AIX documentation.

Particularly for SMP systems, you can have the following MALLOC setting:

- MALLOCMULTIHEAP=1 (for SMP systems)

Setting other environment variables

You can enhance the performance of AIX system by setting the AIXTHREAD_SCOPE and NODISCLAIM environment as shown in the following commands. Check the AIX documentation to see if these settings might be right for your installation.

AIXTHREAD_SCOPE
To set AIXTHREAD_SCOPE, use the following command:

```
export AIXTHREAD_SCOPE=S
```

NODISCLAIM
To set NODISCLAIM, use the following command:

```
export NODISCLAIM=TRUE
```

SPINLOOPTIME
For SMP systems, use the following:

```
export SPINLOOPTIME=650
```

Viewing ibmslapd environment variables (AIX operating system only)

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

```
ps ewww PID | tr ' ' '\012' | grep = | sort
```

where *PID* is the **ibmslapd** process ID.

Example output:

```
ACLCACHE=YES
ACLCACHESIZE=25000
AIXTHREAD_SCOPE=S
AUTHSTATE=compat
A__z=!
CLASSPATH=/home/ldapdb2/sqllib/java/db2java.zip:/home/ldapdb2/sqllib/java/
 db2jcc.jar:/home/ldapdb2/sqllib/function:/home/ldapdb2/sqllib/java/
 db2jcc_license_cisuz.jar:/home/ldapdb2/sqllib/java/db2jcc_license_cu.jar:.
DB2CODEPAGE=1208
DB2INSTANCE=ldapdb2
HOME=/
IDS_LDAP_HOME=/opt/IBM/ldap/V6.3
LANG=en_US
LC__FASTMSG=true
LD_LIBRARY_PATH=/home/ldapdb2/sqllib/lib
LIBPATH=/opt/IBM/ldap/V6.3/lib64:/usr/lib:/home/ldapdb2/idsslapd-ldapdb2/
 db2instance/lib:/opt/IBM/ldap/V6.3/db2/lib64:/usr/lib:/lib:/home/ldapdb2/
 sqllib/lib:.
LOCPATH=/usr/lib/nls/loc
LOGIN=root
LOGNAME=root
MAIL=/usr/spool/mail/root
MAILMSG=[YOU
MALLOCTYPE=buckets
NLSPATH=/opt/IBM/ldap/V6.3/nls/msg/%L/%N:/opt/IBM/ldap/V6.3/nls/msg/%L/%N.cat:/
 usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
NODISCLAIM=TRUE
ODBCCONN=15
ODMDIR=/etc/objrepos
PATH=/opt/IBM/ldap/V6.3/sbin:/opt/IBM/ldap/V6.3:/usr/bin:/etc:/usr/sbin:/usr/
 ucb:/usr/bin/X11:/sbin:/usr/java14/jre/bin:/usr/java14/bin:/usr/java131/jre/
 bin:/usr/java131/bin:/home/ldapdb2/sqllib/bin:/home/ldapdb2/sqllib/adm:/
 home/ldapdb2/sqllib/misc
PWD=/home/ldapdb2/idsslapd-ldapdb2/workdir
RDBM_CACHE_BYPASS_LIMIT=100
RDBM_CACHE_SIZE=25000
RDBM_FCACHE_SIZE=25000
SHELL=/usr/bin/ksh
SSH_CLIENT=9.48.85.122
SSH_CONNECTION=9.48.85.122
SSH_TTY=/dev/pts/1
TERM=xterm
TISDIR=/opt/IBM/ldap/V6.3
TZ=CST6CDT
USER=root
VWSPATH=/home/ldapdb2/sqllib
_=/opt/IBM/ldap/V6.3/sbin/64/ibmslapd
instname=ldapdb2
location=/home/ldapdb2
```

Chapter 5. Hardware tuning

This chapter contains some suggestions for improving disk drive performance.

Disk speed improvements

With millions of entries in an 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
- Use raw devices for storing the tablespace data

Chapter 6. IBM Tivoli Directory Server features

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

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

Bulkload

The bulkload utility accepts many command line options introduced in previous releases for performance tuning. Many of these tuning options are deprecated. The default for the following options are optimal and should not be specified.

-A <yes|no>
Specifies whether to process the ACL information contained in the LDIF file. The default is **yes**. The **no** parameter loads the default acls.

-c | -C <yes|no>
Allows you to skip index recreation. For example, if you are running successive idsbulkloads and you want to skip recreation between loads, you can postpone index creation until the last idsbulkload. Issue the final idsbulkload with **-c yes**.

-e <yes|no>
Drop indexes before load.

Effects of using the -k option

The **-k** option enables users to bulkload their data in smaller chunks. It is especially useful for systems where memory is limited. What happens when utilizing this option is that the parsing and corresponding loading is done in smaller increments.

Note: Saving on memory by specifying small chunksize can result in the user experiencing longer bulkload times.

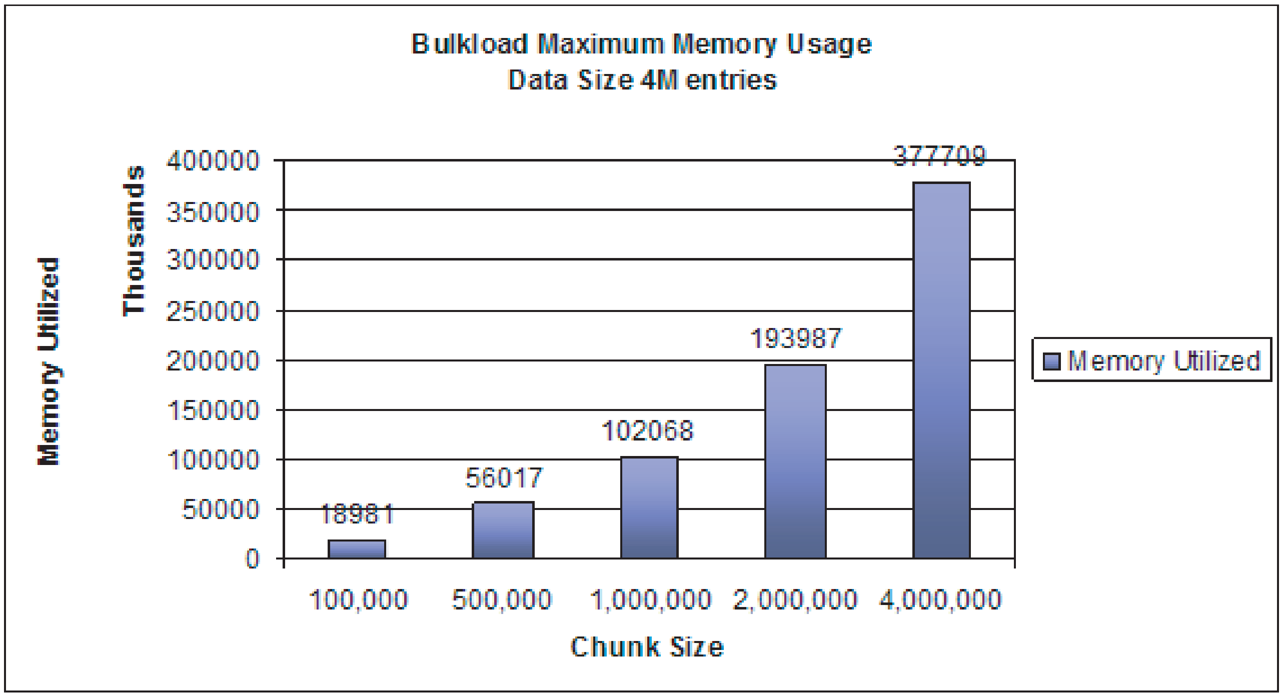


Figure 7. Effects of using the -k option

This graph illustrates the effects on memory usage. As the chunk size increases, the amount of memory utilized increases.

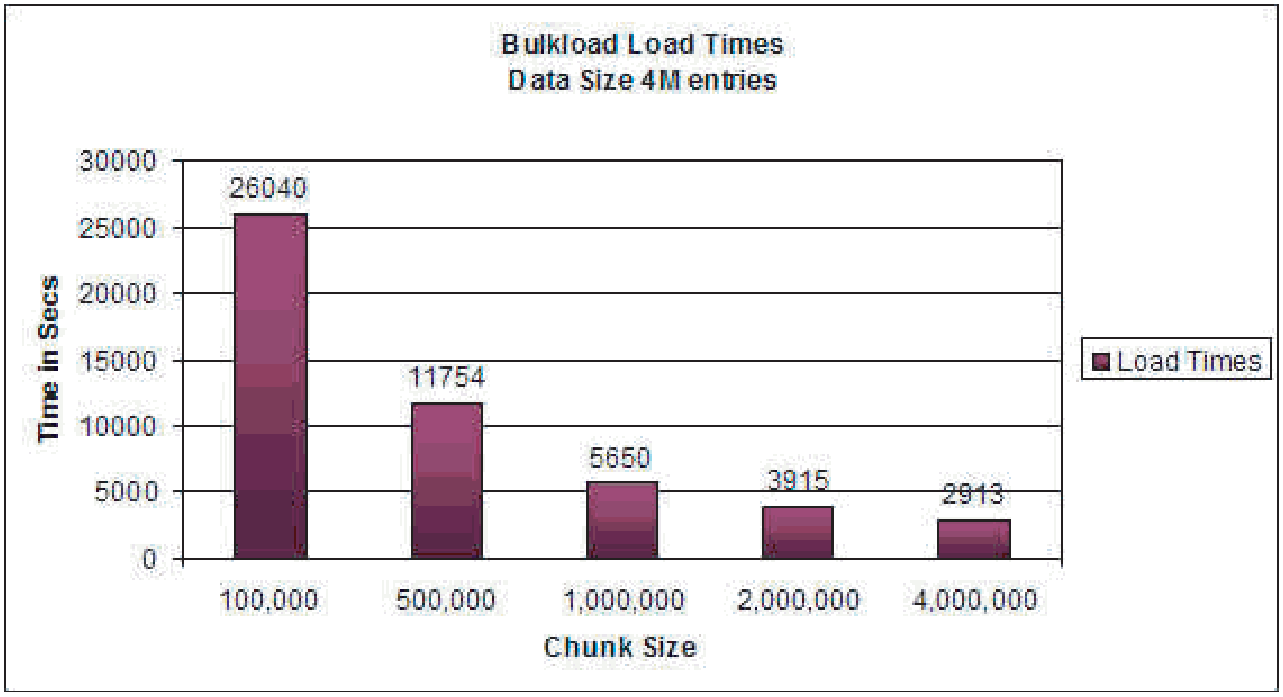


Figure 8. Effects of using the -k option

This graph illustrates that as the chunk size increases, the load time decreases. The recommendation is to use chunk sizes of one million entries at least.

Replication tuning

Replication is a technique used by directory servers to improve performance, availability, and reliability. The replication process synchronizes data stored in multiple directory servers.

Using multi-threaded (asynchronous) replication, administrators can replicate using multiple threads. These features were added to improve overall throughput of replication. For more information about asynchronous replication, see "Multi-threaded (asynchronous) replication" in the *IBM Tivoli Directory Server Version 6.3 Administration Guide*.

Anyone with a replication backlog might consider switching to multi-threaded (asynchronous) replication. Candidate environments can include the following:

- A high update rate
- No downlevel servers
- Common AES salt and synchronization if encryption is AES and passwords are updated often
- Small fanout (for example, 8 connections per agreement with 24 replicas might be too complicated depending on system configuration)
- Available servers and reliable network
- Real-time data consistency is not critical
- All replication schedules are immediate
- Multiprocessor machines

Multi-threaded (asynchronous) replication is difficult to administer if servers or networks are not reliable.

When errors occur, the errors are logged and can be replayed by the administrator, but the error logs must be monitored closely. The following is a search to show the replication backlog for all agreements supplied by one server:

```
ldapsearch -h supplier-host -D cn=admin -w ? -s sub -b <replication_context>
 objectclass=ibm-replicationagreement
 ibm-replicationpendingchangecount ibm-replicationstate
```

If the replication state is active, and the pending count is growing, there is a backlog that won't decrease unless the update rate decreases or the replication mode is changed from synchronous to asynchronous (multi-threaded).

Replication also adds to the workload on the master server where the updates are first applied. In addition to updating its copy of the directory data, the master server must send the changes to all replica servers. If your application or users do not depend on immediate replication, then careful scheduling of replication to avoid peak activity times will help minimize the impact to throughput on the master server. See "Creating replication schedules" in the *IBM Tivoli Directory Server Version 6.3 Administration Guide*.

The following are areas where tuning adjustment can be made to improve performance:

- Number of replication threads per supplier and consumer
- Replication context cache size
- Replication ready size limit

Number of replication threads

The number of replication threads (ibm-replicaconsumerconnections) attribute represents the number of connections used for each replication agreement. In testing, as the number of threads on both the supplier and consumer are increased, the transaction rate also increased. In the following graph, a transaction is defined as a queued replication record that is sent over, in this case an ldap_modify, to the supplier. The queued replication records (ldap_modify) are run with replication in a pending state. The replication state is then changed to **resume**, which starts the replication process.

Note: As the number of threads increases, so does the CPU usage on both supplier and consumer systems. Adjust this attribute as needed based on acceptable CPU usage and desired throughput.

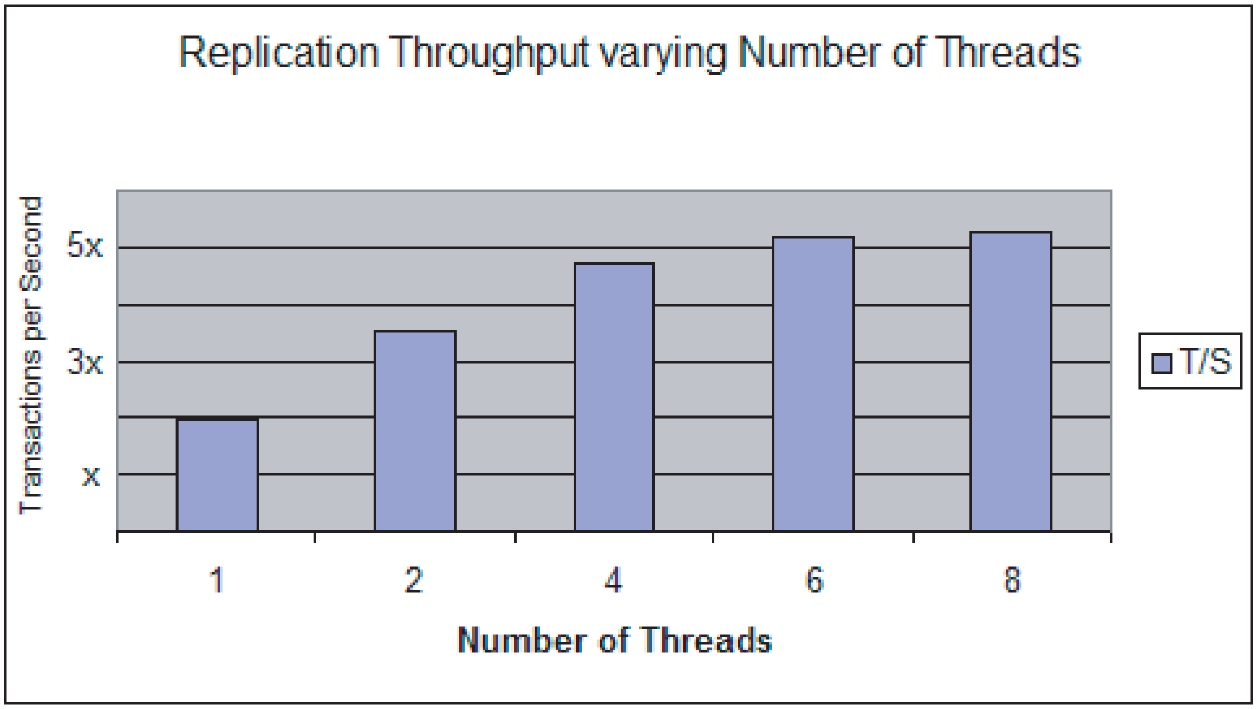


Figure 9. Number of replication threads

As the throughput increased, the CPU consumption on both the supplier and consumer increased. The CPU cost per transaction on the consumer increased slightly when adding threads, as there were more threads to manage.

Replication context cache size

The replication context cache size (ibm-slapdReplContextCacheSize) is an attribute that specifies, in bytes, the size in memory that is allocated to cache updates to be replicated. The default setting is 100,000 bytes. This attribute cannot be updated dynamically.

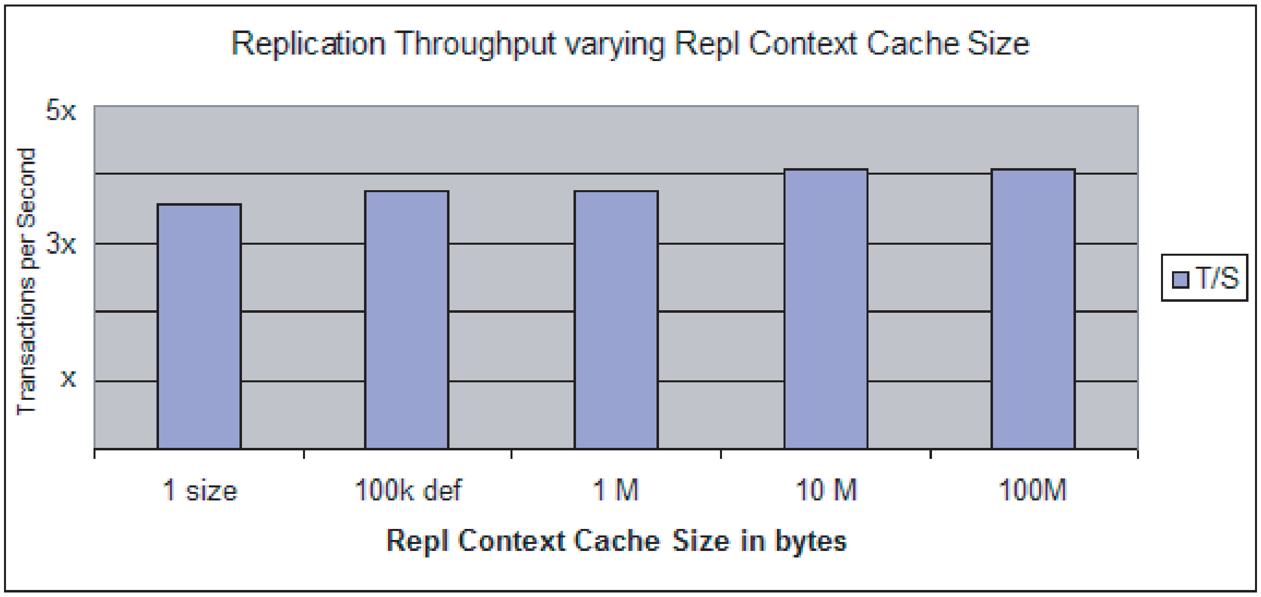


Figure 10. Replication context cache size

Replication ready size limit

The replication ready size limit (environment variable IBMSLAPD_REPL_READY_SIZE_LIMIT) controls the size of the queues of replication operations from the list of updates still to be replicated. The default size is 10. There is one queue per connection to a given replica. Related updates (for example, modifications or children of new entries) will be placed in the same queue. If the size of this queue exceeds the specified size limit, the main replication thread waits for the queue size to get below the limit again. This prevents the main replication thread from using too much CPU determining dependencies between the updates. In testing, the size of this queue was varied from 1 entry up to 200 entries. Although an increase in raw throughput was not evident, CPU savings were realized at certain settings of this parameter. The following chart displays throughput normalized to 100% CPU. Absolute throughput did not change in this test. A larger number in this graph means less CPU cost per transaction. In this graph a transaction is defined as a queued replication record (ldap_modify) that is sent to the supplier.

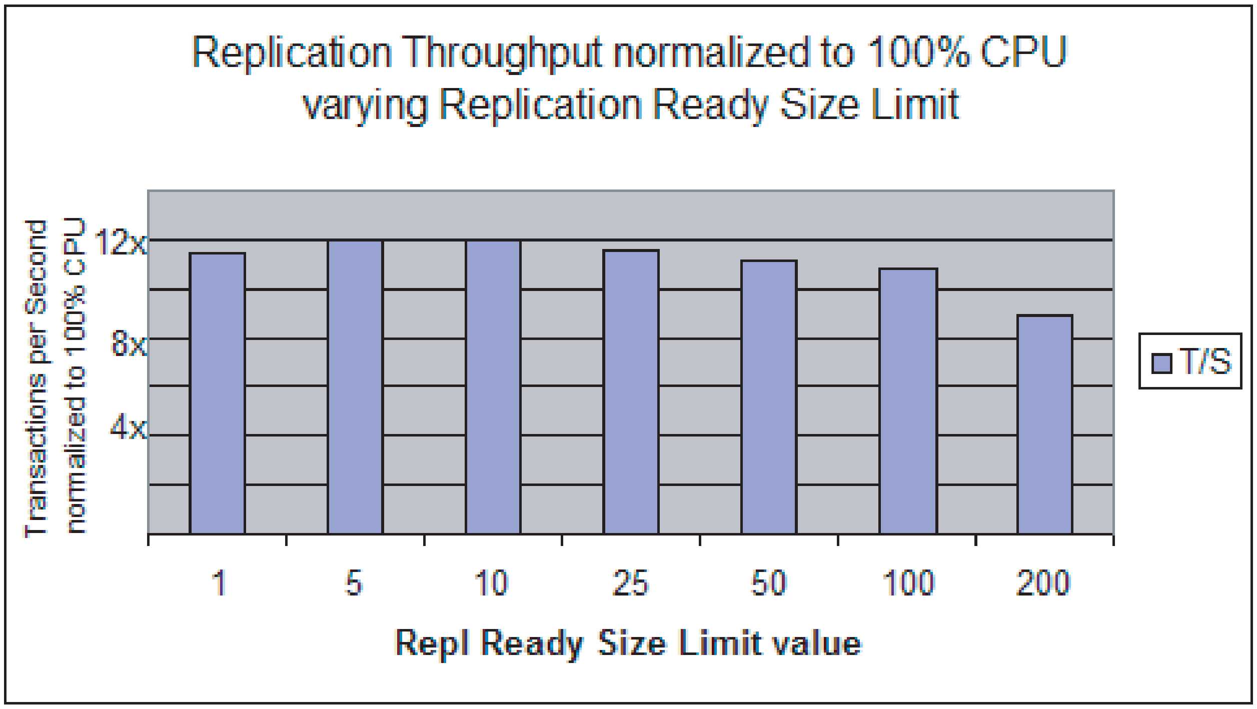


Figure 11. Replication ready size limit

Tuning Tivoli Directory Server audit log

The Tivoli Directory Server's audit logging feature significantly slows down Tivoli Directory Server performance, depending upon which audit logging features are turned on. If not required, it is advisable to turn off all audit logging features. The Tivoli Directory Server instance must be running for which a user want to disable the audit. To check the status of the audit logging feature, run the ldapsearch command of the following format:

```
idsldapsearch -p <port> -D <adminDN> -w <adminPwd> -s base \
   -b "cn=audit,cn=log management,cn=configuration" objectclass=* ibm-audit
cn=Audit, cn=Log Management, cn=Configuration
ibm-audit=false
```

where **ibm-audit=false** indicates that audit logging is off. If this value is true, to set the value to false for the ibm-audit attribute, run the ldapmodify command of the following format:

```
idsldapmodify -p <port> -D <adminDN> -w <adminPwd>
dn: cn=Audit, cn=Log Management, cn=Configuration
changetype: modify
replace: ibm-audit
ibm-audit: false
```

IBM Directory Proxy Server tuning

IBM Directory Proxy Server can be used in environments where the size of the data store exceeds the processing power and physical capacity of a single machine. Directory sizes greater than 40 M entries are candidates for a distributed directory environment. A proxy server gives customers the ability to distribute data across multiple backend servers.

Throughput performance of a proxy server can be affected by the size of the connection pool. This is a parameter configured on the IBM Directory Proxy server. For best results, the following guidelines to set an appropriate connection pool size must be followed:

- Configure more than one connection to the back-end server.
- Limit the connection pool size to the number of connections the operating system can support. The connection pool is a static pool of connections that the proxy server sets up during the proxy server startup. The operating system imposes a limit on the number of open file descriptors. The connection pool size should be less than this limit.
- Ensure that the connection pool size is less than the number of database connections configured with the back-end server, keeping a buffer for replication and changelog.

For better performance, all back-end servers and the proxy server should share the same stash files.

Monitoring performance

The **ldapsearch** command can be used to monitor performance, as shown in the following sections.

ldapsearch with "cn=monitor"

The following **ldapsearch** command uses "cn=monitor".

```
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:

cn=monitor

version=IBM Tivoli Directory, Version 6.3

total connections
: The total number of connections since the server was started.

current connections
: The number of active connections.

maxconnections
: The maximum number of active connections allowed.

writewaiters
: The number of threads sending data back to the client.

readwaiters
: The number of threads reading data from the client.

livethreads
: The number of worker threads being used by the server.

filter_cache_size
: The maximum number of filters allowed in the cache.

filter_cache_current
: The number of filters currently in the cache.

filter_cache_hit
: The number of filters retrieved from the cache rather than being resolved in DB2.

filter_cache_miss
: The number of filters that were not found in the cache that then needed to be resolved by DB2.

filter_cache_bypass_limit
: Search filters that return more entries than this limit are not cached.

entry_cache_size
: The maximum number of entries allowed in the cache.

entry_cache_current
: The number of entries currently in the cache.

entry_cache_hit
: The number of entries that were retrieved from the cache.

entry_cache_miss
: The number of entries that were not found in the cache that then needed to be retrieved from DB2.

acl_cache
: A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE).

acl_cache_size
: The maximum number of entries in the ACL cache.

currenttime
: The current time on the server. The current time is in the format:

```
year month day hour:minutes:seconds GMT
```

Note: If expressed in local time the format is

```
day month date hour:minutes:seconds timezone year
```

starttime
: The time the server was started. The start time is in the format:

```
year month day hour:minutes:seconds GMT
```

Note: If expressed in local time the format is

```
day month date hour:minutes:seconds timezone year
```

en_currentregs
: The current number of client registrations for event notification.

en_notificationssent
: The total number of event notifications sent to clients since the server was started.

The following attributes are for operation counts:

bindsrequested
: The number of bind operations requested since the server was started.

bindscompleted
: The number of bind operations completed since the server was started.

unbindsrequested
: The number of unbind operations requested since the server was started.

unbindscompleted
The number of unbind operations completed since the server was started.

addsrequested
The number of add operations requested since the server was started.

addscompleted
The number of add operations completed since the server was started.

deletesrequested
The number of delete operations requested since the server was started.

deletescompleted
The number of delete operations completed since the server was started.

modrdnsrequested
The number of modify RDN operations requested since the server was started.

modrdnscompleted
The number of modify RDN operations completed since the server was started.

modifiesrequested
The number of modify operations requested since the server was started.

modifiescompleted
The number of modify operations completed since the server was started.

comparesrequested
The number of compare operations requested since the server was started.

comparescompleted
The number of compare operations completed since the server was started.

abandonsrequested
The number of abandon operations requested since the server was started.

abandonscompleted
The number of abandon operations completed since the server was started.

extopsrequested
The number of extended operations requested since the server was started.

extopscompleted
The number of extended operations completed since the server was started.

unknownopsrequested
The number of unknown operations requested since the server was started.

unknownopscompleted
The number of unknown operations completed since the server was started. Unrecognized operations are rejected with a result message to the client including the LDAP_UNWILLING_TO_PERFORM result code.

opsinitiated
The number of initiated requests since the server was started.

opscompleted
The number of completed requests since the server was started.

entriessent
The number of entries sent by the server since the server was started.

searchesrequested
: The number of initiated searches since the server was started.

searchescompleted
: The number of completed searches since the server was started.

The following attributes are for server logging counts:

slapderrorlog_messages
: The number of server messages recorded since the server was started or since a reset was performed.

slapdclierrors_messages
: The number of DB2 error messages recorded since the server was started or since a reset was performed.

auditlog_messages
: The number of audit messages recorded since the server was started or since a reset was performed.

auditlog_failedop_messages
: The number of failed operation messages recorded since the server was started or since a reset was performed.

The following attributes are for connection type counts:

total_ssl_connections
: The total number of SSL connections since the server was started.

total_tls_connections
: The total number of TLS connections since the server was started.

The following attributes are for tracing:

trace_enabled
: The current trace value for the server. TRUE, if collecting trace data, FALSE, if not collecting trace data.

trace_message_level
: The current ldap_debug value for the server. The value is in hexadecimal form, for example:

```
0x0=0
0xffff=65535
```

trace_message_log
: The current LDAP_DEBUG_FILE environment variable setting for the server.

The following attributes are for denial of service prevention:

available_workers
: The number of worker threads available for work.

current_workqueue_size
: The current depth of the work queue.

largest_workqueue_size
: The largest size that the work queue has ever reached.

idle_connections_closed
: The number of idle connections closed by the Automatic Connection Cleaner.

auto_connection_cleaner_run
The number of times that the Automatic Connection Cleaner has run.

The following attribute is for alias dereference processing:

bypass_deref_aliases
The server runtime value that indicates if alias processing can be bypassed. It displays TRUE if no alias object exists in the directory, and FALSE if at least one alias object exists in the directory.

The following attributes are for the attribute cache:

cached_attribute_total_size
The amount of memory used by the directory attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

cached_attribute_configured_size
The maximum amount of memory, in kilobytes, that is enabled to be used by the directory attribute cache.

cached_attribute_hit
The number of times the attribute has been used in a filter that could be processed by the attribute cache. The value is reported as follows:

```
cached_attribute_hit=attrname:#####
```

cached_attribute_size
The amount of memory used for this attribute in the attribute cache. This value is reported in kilobytes as follows:

```
cached_attribute_size=attrname:######
```

cached_attribute_candidate_hit
A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the directory attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

```
cached_attribute_candidate_hit=attrname:#####
```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

Examples

The following sections show examples of using values returned by the **ldapsearch** command with "cn=monitor" to calculate the throughput of the server and the number of add operations completed on the server in a certain timeframe.

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

Suppose the values for the **opscompleted** 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 opscompleted (**t1**) and opscompleted (**t2**). The average throughput at the server during the interval between **t1** and **t2** can be calculated as:

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

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

Workload example: The monitor attributes can be used to characterize the workload, similar to the throughput example but split out by type of operation. For example, you can calculate the number of add operations that were completed in a certain amount of time.

Suppose the values for the **addscompleted** 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 addscompleted (**t1**) and addscompleted (**t2**). The number of add operations completed on the server during the interval between **t1** and **t2** can be calculated as:

```
(addscompleted(t2) - addscompleted(t1) /(t2 -t1)
```

Similar calculations can be done for other operations, such as **searchescompleted**, **bindscompleted**, **deletescompleted**, and **modifiescompleted**.

ldapsearch with "cn=workers,cn=monitor"

An administrator can run a search using "cn=workers,cn=monitor" to get information about what worker threads are doing and when they started doing it.

```
ldapsearch -D <adminDN> -w <adminpw> -b cn=workers,cn=monitor -s base objectclass=*
```

This information is most useful when a server is performing poorly or not functioning as expected. It should be used only when needed to give insight into what the server is currently doing or not doing.

The "cn=workers, cn=monitor" search returns detailed activity information only if auditing is turned on. If auditing is not on, "cn=workers, cn=monitor" returns only thread information for each of the workers.

Attention: The "cn=workers,cn=monitor" search suspends all server activity until it is completed. For this reason, a warning should be issued from any application before issuing this feature. The response time for this command will increase as the number of server connections and active workers increase.

For more information, see the *IBM Tivoli Directory Server Version 6.3 Administration Guide*.

ldapsearch with "cn=connections,cn=monitor"

An administrator can run a search using "cn=connections,cn=monitor" to get information about server connections:

```
ldapsearch -D<adminDN> -w <adminPW> -h <servername> -p <portnumber>
         -b cn=connections,cn=monitor -s base objectclass=*
```

This command returns information in the following format:

```
cn=connections,cn=monitor
connection=1632 : 9.41.21.31 : 2002-10-05 19:18:21 GMT  : 1 : 1 : CN=ADMIN : :
connection=1487 : 127.0.0.1 : 2002-10-05 19:17:01 GMT  : 1 : 1 : CN=ADMIN : :
```

Note: If appropriate, an SSL or a TLS indicator is added on each connection.

For more information, see the *IBM Tivoli Directory Server Version 6.3 Administration Guide*.

ldapsearch with "cn=changelog,cn=monitor"

You can run a search using "cn=changelog,cn=monitor" to obtain information about the changelog attribute cache. (See "When to configure Tivoli Directory Server change log" for information about the change log.) The command returns the following information:

cached_attribute_total_size
The amount of memory used by the changelog attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

cached_attribute_configured_size
The maximum amount of memory, in kilobytes, that is enabled to be used by the changelog attribute cache

cached_attribute_hit
The number of times the attribute has been used in a filter that could be processed by the changelog attribute cache. The value is reported as follows:

```
cached_attribute_hit=attrname:#####
```

cached_attribute_size
The amount of memory used for this attribute in the changelog attribute cache. This value is reported in kilobytes as follows:

```
cached_attribute_size=attrname:######
```

cached_attribute_candidate_hit
A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the changelog attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

```
cached_attribute_candidate_hit=attrname:#####
```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

When to configure Tivoli Directory Server change log

IBM Tivoli Directory Server has a function called **change log** that results in a significantly slows down 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's directory information tree (DIT). 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.

Chapter 7. Capacity Planning

There are several hardware decisions to be made before deploying IBM Tivoli Directory Server. Hardware resources to consider are:

- Hard disk
- Memory
- CPUs

IBM Tivoli Directory Server can perform differently for various hardware configurations, and it is important to understand which hardware configurations cause the directory server to perform best.

Several tuning measures were taken to find the best settings under which the server provides best results. Tuning factors tested were:

IBM Tivoli Directory Server tuning
LDAP Entry cache

DB2 tuning

- DB2 buffer pools
  - LDAP bufferpool
  - IBMDEFAULT bufferpool
- Optimization and organization (reorgchk and reorg)
- Other DB2 configuration parameters
- Backing up and restoring the database (backup and restore)
- Splitting of database

The capacity planning information in this chapter includes results observed from data loading (using the **bulkload** utility) and running specific benchmarks on a variety of AIX systems.

The results provided were obtained through the following methods: Two types of LDIF files were used. The first type of LDIF file contains small entries with a flat tree structure. The other type of LDIF file contains larger entry sizes as well as a deeper tree structure. LDIF files with 100,000 and 1 million entries were created. The first type of LDIF file was used only for search operations. The second type was used for both searches and updates. In each case, a directory server instance is created on the system. The LDIF file is loaded into the database and the recording are made. Operating system information and information related to the directory server instance is collected at intervals throughout the load and the performance test run.

Note: The statistics reported here are specific to the particular hardware setups used and were generated in a lab environment. These results might not be reproducible in other environments. The results reported here should be used only as guidelines.

The LDAP server performance was monitored by running various workloads on different hardware configurations. These workloads were run on a number of AIX systems.

Disk requirements

Because large amounts of data are stored in the directory server, it would be useful to have a formula that determines the capacity of the hard disk that is required to store the complete data. Also useful is a measure of how much CPU, memory, and time is required to load this large amount of data into the directory server from an LDIF file. The two most important factors here are:

- Time required to load the data into the directory server
- Space required to store the data on the hard disk

Bulkload time and space information

Two types of LDIF files were used to gather data statistics. The first is a flat structured LDIF file with small entries. The second is a deeper tree with larger entries. There are no access control lists (ACLs) or groups in either LDIF file.

Example small entry, approximately 415 bytes:

```
dn: cn=Joline Hickey, ou=Accounting, o=IBM.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
cn: Joline Hickey
sn: Hickey
description: Joline_Hickey
facsimileTelephoneNumber: +1 71 631-7308
l: Palo Alto
ou: Accounting
postalAddress: IBM.com$Accounting$Dept # 363$Room # 890
telephoneNumber: +1 408 995-7674
title: Supreme Accounting Mascot
userPassword: yekciHenil
seeAlso: cn=Joline
```

Example large entry, approximately 10,510 bytes:

```
dn: mdsListName=List219, mdsContainerName=container22, mdsUID=22, mdso=393, mdsc=C1,
 dc=IBM, dc=com
objectclass: MDSBlobList
mdsListName: List219
mdsHeadTitle: Listen Titel 9
mdsdata:: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXX
...
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
```

The attribute “mdsdata” contains binary data. Binary data is not stored in attribute tables and has a big impact on total disk space and on parsing and loading times.

Time Information

The following graphs and tables show results from running the **bulkload** utility with sizes of 100,000 entries and 1 million entries (for small and large entries). The **bulkload** was run in two passes, first the parse and then the load. The information was collected on an AIX 5.3 system with two 1656 MHz POWER5™ processors and 3792 MB of RAM. No tuning was done on the database settings before running the **bulkload** utility. For the smaller entry data, an index for the seeAlso attribute was added.

Note: The following results were obtained by using the **bulkload** utility with DB2 v9.x.

Times for bulkload for LDIF files with 100,000 entries

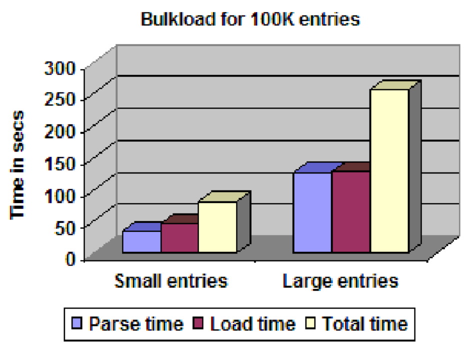


Table 2. Bulkload times for 100,000 entries

Time in seconds	Small entries	Large entries
Parse time	35	127
Load time	47	128
Total time	82	255

Times for bulkload for LDIF files with 1,000,000 entries

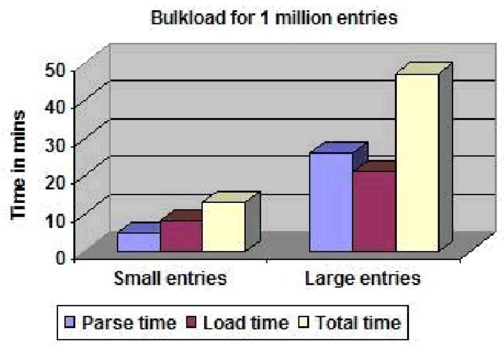


Table 3. Bulkload times for 1,000,000 entries

Time in minutes	Small entries	Large entries
Parse time	5	25
Load time	8	112
Total time	13	137

By comparing the **bulkload** times for the same LDIF file on two different systems, the time variations and impact of the hardware can be observed. The following graph and table show the results for two **bulkload** runs with the 1,000,000 small entries LDIF file. The two systems used were of the following specification:

System1
AIX 5.3 with 2 1656 MHz POWER5 processors and 3792 MB of RAM

System2
AIX 6.1 with 2 1499 MHz POWER5 processors and 7967 MB of RAM (for 1 logical partition (LPAR))

Times for bulkload for LDIF file with 1,000,000 small entries

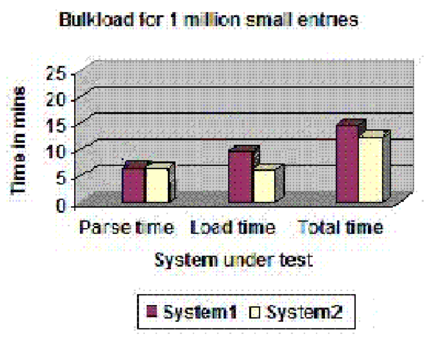


Table 4. Bulkload times for 1,000,000 small entries on two different systems

Time (minutes)	System1	System2
Parse time	5	5.12
Load time	8	5.41
Total time	13	10.53

The AIX 5.3 system performed the **bulkload** parse operation in same time as that of on AIX 6.1 system, but AIX 5.3 took more time for loading the same amount of data. The **bulkload** utility is single threaded, so improvements in time come from the CPU speed, disk speed of the hardware being used, and memory used.

In general, the parse time increases linearly as the number of entries in the LDIF file increases. For example, the parse time for 100,000 small entries was 35 seconds and the parse time for 1,000,000 small entries was roughly 10 times that, or 5 minutes. This is not true for the load times, however. Load time does not increase linearly with respect to the data set size.

During the parse phase, intermediate files are generated; these files are used in the load phase. If the input LDIF file is large, more data is written to the intermediate files. The time increase is mostly in disk I/O, and it can be seen in the results obtained for the different LDIF files. The average entry parse time per second is significantly greater for the larger entry size LDIF. For the 100,000 entry LDIF file with smaller sized entries, an average of 2800 entries were parsed per second. For the larger sized entries, however, only 780 entries were parsed per second.

Space information

On AIX 5.3 system, the space needed in three different filesystem directories was calculated for each of the four LDIF files that were created. The first was the space in the temporary directory used by the parse phase of the **bulkload** operation. This can be referred as the parse size. The second is the space needed in the actual directory where the database is stored. This can be referred as the database size. The third is the space needed to hold a backup of the database. This can be referred as the backup size. The charts displayed below show these statistics for both the small and large entry files, and for 100,000, 1,000,000 entries along with the size of the corresponding LDIF file.

Space used for bulkload for LDIF file with 100,000 entries

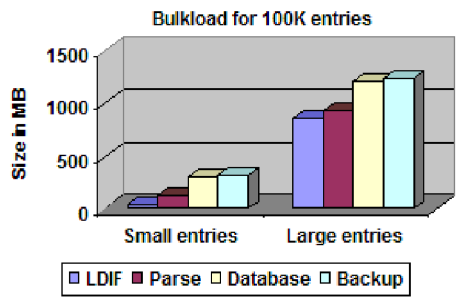


Table 5. Bulkload space for 100,000 entries

Size in MB	Small entries	Large entries
LDIF file size in MB	41	859
Parse space in MB	139	927
Database space in MB	304	1213
Backup space in MB	320	1233

Space used for bulkload for LDIF file with 1,000,000 entries

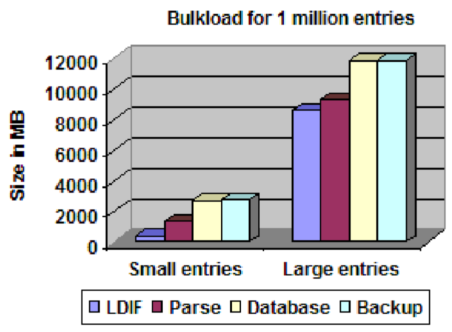


Table 6. Bulkload space for 1,000,000 entries

Size in MB	Small entries	Large entries
LDIF size in MB	407	8597
Parse space in MB	1405	9294
Database space in MB	2779	11864
Backup space in MB	2793	11866

From this data, you can generalize the amount of space needed per entry for the different types of entries. For the smaller entries about 3 KB of space per entry is needed for database storage. For the larger entries, the requirement increases to 12 KB per entry.

The space needed to back up the database is roughly equivalent to the space needed for the database itself. The space needed for the parse phase of the bulkload is about 3.5 times the size of the LDIF file for the smaller entries (or 1.5 KB per entry), and about 1.2 times the size of the LDIF file for the larger entries (or 9.5 KB per entry).

Memory requirements

The runtime memory requirements for the Tivoli Directory Server and the requisite product, IBM DB2, vary with number of users in the directory server, the size of the directory server caches, and the size of the DB2 buffer pools. When the LDAP server is tuned for performance gain, it is generally done by tuning the size of the LDAP caches and setting the DB2 buffer pools to AUTOMATIC. However, while doing this the memory capacity of the system must be considered. Allocating a very large amount for the caches will result in an increase in the overall memory requirement. Therefore, cache sizes must be allocated carefully. See Chapter 2, “IBM Tivoli Directory Server tuning,” on page 7 and Chapter 3, “Tuning DB2 and LDAP caches,” on page 23 for information.

CPU requirements

To ensure that the CPU is used at its optimum level, here are some guidelines:

- Ensuring that the required data is available in the caches results in low disk accesses, which are generally slower than memory accesses. This decreases the time that the CPU must wait for input/output to occur.
- Enabling simultaneous multithreading (SMT) on systems that are capable of hyperthreading increases the processing capability and the application throughput. See “Simultaneous multithreading” on page 79 for more information.
- Systems with fewer CPUs but greater CPU frequency work more efficiently than systems with more CPUs but lower CPU frequency. For example, a system with 2 processors with 1200 MHz speed performs better than a system with 4 processors with 600 MHz speed.

CPU scaling comparison for throughput (searches and updates)

The processor plays an important role in the overall performance of any application. There are two important factors for a processor that add to the overall performance:

- Number of processors
- Processor speed

Scaling of throughput for varying number of processors

To determine the effect of the number of processors on the overall performance of the directory server, all the hardware configurations were kept common, except for the number of processors. The workloads were run on 3 different AIX 5.3 systems with 1, 2, and 4 POWER® V processors (1499 MHz) and 6144 MB RAM.

On each system, the directory server was loaded with 100,000 entries. The same update and search workload was run on all the systems. The update workload consisted of modify and modrdn operations, while the search workload consisted of rootdse searches along with subtree searches. The entry cache size was set to the default value of 25 KB. The results for the search and update workloads with varied number of processors are shown in the following sections.

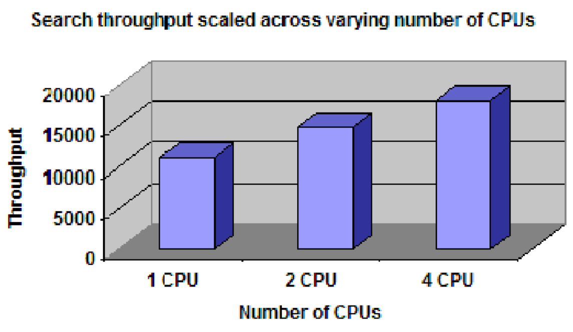
Search throughput: The following table shows the throughput figures for a directory server with 100,000 entries:

Table 7. Search throughput with 1, 2, and 4 CPUs

Number of CPUs	Throughput	CPU User time	CPU system time	CPU idle time	CPU wait time
1 CPU	11129.1	65	35	0	0
2 CPUs	14899.9	59	34	7	0
4 CPUs	18199.5	37	23	40	0

The results for a dual processor configuration are far better than the results for a computer with a single processor. Best results can be obtained using a multiprocessor computer.

The following graph shows the scaling of the search throughput across varying number of processors.

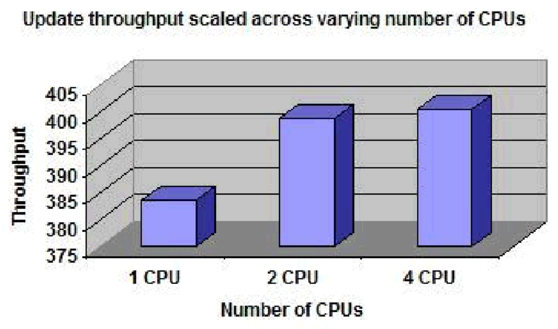


Update throughput: The following table shows the throughput figures for a directory server loaded with 100,000 entries:

Table 8. Update throughput for 1, 2, and 4 CPUs

Number of CPUs	Throughput	CPU User time	CPU system time	CPU idle time	CPU wait time
1 CPU	383.5	65	35	0	0
2 CPUs	398.7	59	34	7	0
3 CPUs	400.4	37	23	40	0

The following graph shows the scaling of the update throughput with varying numbers of processors.



The SMT option can be used to simulate the computer having more processors than are physically present. For more information about SMT, see “Simultaneous multithreading” on page 79.

Splitting the database across multiple disks

The tests showed that when the database is split across two different disks instead of residing on one disk, the throughput achieved is doubled. CPU utilization is one important factor that is responsible for this. The following tables show the effect on CPU utilization of splitting the database across two hard disks.

Effect on search throughput of splitting the database across two hard disks

Table 9. Search throughput with one and two hard disks

Database	Throughput	IB	LB	CPU %	Idle %	Wait %	AVM (MB)	Memory (KB)
Normal	260	95	74	23	68	8	191	389044
Split	416	95	74	46	37	16	179	378464

The update workload consists of search and modify operations on a directory with 1 million entries. The computer is a AIX 5.3 system with 2 POWER V processors (1499 MHz) and 6144 MB RAM. The CPU idle % is reduced to half when the database is split across two different disks. The results can be further enhanced by using a high speed hard disk drive to save I/O time as well.

Effect on update throughput of splitting the database across two hard disks

Table 10. Update throughput with one and two hard disks

Database	Throughput	IB	LB	CPU %	Idle %	Wait %	AVM (MB)	Memory (KB)
Normal	118	92	73	23	68	9	496	400736
Split	237	93	72	47	36	16	451	398960

For information about splitting the database across multiple disks, see the redpaper entitled "Performance Tuning for IBM Tivoli Directory Server."

Simultaneous multithreading

Simultaneous multithreading is a processor design that combines hardware multithreading with superscalar processor technology to allow multiple threads to issue instructions each cycle. Unlike other hardware multithreaded architectures in which only a single hardware context (or thread) is active on any given cycle, SMT permits all thread contexts to simultaneously compete for and share processor resources. Unlike conventional superscalar processors, which suffer from a lack of per-thread instruction-level parallelism, simultaneous multithreading uses multiple threads to compensate for low single-thread instruction-level parallelism. Simultaneous multithreading allows multiple threads to run different instructions in the same clock cycle, using the execution units that the first thread left spare. This is done without great changes to the basic processor architecture: the main additions needed are the ability to fetch instructions from multiple threads in a cycle, and a larger register file to hold data from multiple threads. The number of concurrent threads can be decided by the chip designers, but practical restrictions on chip complexity usually limit the number to 2, 4 or sometimes 8 concurrent threads.

The performance consequence is significantly higher instruction throughput and program speedups on a variety of workloads that include commercial databases, web servers and scientific applications in both multiprogrammed and parallel environments.

SMT on AIX FAQs

How would I know if my system is capable of using simultaneous multithreading)?
Your system is capable of SMT if it is a POWER5-based system running AIX 5L™ Version 5.3.

How would I know if SMT is enabled for my system?
If you run the **smtctl** command without any options, the response tells you if SMT is enabled or not.

Is SMT supported for the 32-bit kernel?
Yes, SMT is supported for both 32-bit and 64-bit kernel.

How do I enable or disable SMT?
You can enable or disable SMT by running the **smtctl** command. The following is the syntax:

```
smtctl [ -m off | on [ -w boot | now]]
```

The following options are available:

-m off
Sets SMT mode to disabled.

-m on
Sets SMT mode to enabled.

-w boot
Makes the SMT mode change effective on next and subsequent restarts if you run the **bosboot** command before the next system restart.

-w now
Makes the SMT mode change immediately but the change does not persist across restart.

If neither the **-w boot** or the **-w now** options are specified, the mode change is made immediately. It persists across subsequent restarts if you run the **bosboot** command before the next system restart.

Appendix A. Workload description

The performance tests are performed for base search and mixed job (a mixture of update, search, and compare operations). Each scenario consists of two phases, a warm-up phase and a run phase. During the warm-up phase, the searches primarily request entries that are not in the LDAP caches, most of these requests require interaction with the DB2 database. For all the measurements reported in this document, warm-up phase consisted of running all queries at least once; consequently, during the run phase all entries requested are potentially already in LDAP caches in memory if the caches are large enough to hold all of them. Thus the warm- up phase and the run phase comprise two distinctly different workloads.

During the run phase of base search job and mixed job, a number of client threads issue search requests to the IBM Tivoli Directory Server from predetermined scripts (clients). The scripts include a number of different kinds of operations, including base searches, update, and compare operations that return multiple entries per request. The client threads run through their scripts continuously for ten minutes. Throughput is measured on the server during each minute interval, and then each client starts over at the beginning of its script. Each ten minute interval is referred to as a run. The server is not restarted between runs.

Appendix B. Modifying TCP/IP settings

When the LDAP server is protected behind a firewall, socket connections might time out resulting in intermittent authentication failures. The socket connection failures are due to a mismatch between the firewall's connection timeout setting and the operating system's frequency of sending keep alive network packets to keep the connection alive. If socket connection failures occur, decrease the operating system network parameters that control the time between sending keep alive packets, sometimes called the keep alive interval.

The name of the parameters that control keep alive frequency vary with each operating system. Set the keep alive interval to be less than the firewall's connection timeout. If unsure of the firewall setting try two minutes.

Additionally, closed TCP/IP connections between the client and the LDAP server are cleaned at system-specified intervals. In environments where the connections are opened or closed at a high frequency, this can degrade LDAP server performance. To shorten the cleaning intervals, modify the registry keys.

Do the following to modify the TCP/IP settings on an AIX platform:

1. Use the following command
   ```
   no -o <attributename>=<value>
   ```
   to set the following attributes and values for performance tuning:

   tcp_keepidle
   Specifies the length of time to keep the connection active. This value is defined in 1/2 second units, and defaults to 14,400 (7200 seconds or 2 hours). The **tcp_keepidle** parameter is a runtime attribute. Reduce tcp_keepidle to be less than the firewall's connection timeout. If unsure, set tcp_keepidle to 240 (2 minutes).

   tcp_keepinit
   Sets the initial timeout value for a tcp connection. This value is defined in 1/2 second units, and defaults to 150 (75 seconds). It can be changed to any value with the **-o** option. The **tcp_keepinit** parameter is a runtime attribute.

   tcp_keepintvl
   Specifies the interval between packets sent to validate the connection. This value is defined in 1/2 second units, and defaults to 150 (75 seconds). The **tcp_keepintvl** parameter is a runtime attribute.

Do the following to modify the registry keys on a Windows platform:

1. Type the following at a command prompt to open Registry Editor:
   ```
   regedit
   ```
2. Go to HKey_Local_Machine\System\CurrentControlSet\Services\Tcpip\Parameters.
3. Add TcpTimedWaitDelay entry (if not already in the registry).
4. Set the DWORD value to **1e** for 30 seconds.
5. Add StrictTimeWaitSeqCheck entry (if not already in the registry).
6. Set DWORD Value to **1**

7. Reboot the machine.

Note: This applies to both client and server machines.

Appendix C. Platform configurations

The performance tuning examples in this guide use the following platform configurations:

-
  - System running SLAMD server
    - One 2x3.2 GHz, 2048 MB RAM, Intel® PRO/100 VE
    - RHEL AS 4
  - System running SLAMD clients
    - Five 2x3.2 GHz, 2048 MB RAM, Intel PRO/100 VE
    - RHEL 5
- Server
  - 4-Way 1.6 GHz, System p Model IBM 9118-575 with 1 or 4 processors active, 31744 MB RAM.
  - IBM 10/100 Ethernet Adapter.
  - AIX 5.3 ML 13
  - IBM Tivoli Directory Server Version 6.3
  - AIXTHREAD_SCOPE=S
  - MALLOCTYPE=buckets
  - NODISCLAIM=true (1way).
  - RDBM_CACHE_SIZE=460000 except where noted.
  - RDBM_FCACHE_SIZE=75000 except where noted.
  - RDBM_CACHE_BYPASS_LIMIT=100 except where noted.
  - Necessary Indexes created (for attribute seeAlso).
  - No ACLs were set. By default, anyone can search and compare. The directory administrator can update.
- DB2 v 9.5
  - maxlocks 100 sortheap 2500 dbheap 5000 ibmdefaultbp 20000 (4K pages) ldapbp 9800 (32K pages)
  - logfilsiz 2048, logprimary 6
- Miscellaneous
  - Caches were warmed up by running all scripts once.
  - Measurements were taken using 15 threads per client except where noted.
  - DB2 log files are not on the same disk as the containers.

Appendix D. Notices

This information was developed for products and services offered in the U.S.A. IBM may 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 described 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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 might 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 publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication 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.

Trademarks

Index

Printed in USA

SC27-2748-00