IBM Tivoli Directory Server

Performance Tuning Guide

Version 6.0

SC32-1677-00

IBM Tivoli Directory Server

Performance Tuning Guide

Version 6.0

SC32-1677-00

First Edition (April, 2005)

This edition applies to version 6, release 0, of IBM Tivoli Directory Server and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2005. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2005

Preface

The purpose of this document is to provide information about improving performance for IBM® Tivoli® Directory Server.

Information such as performance limitations, ways to increase performance, and examples are included.

Who should read this book

The target audience for this book includes:

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

Publications

Read the descriptions of the IBM Tivoli Directory Server library, the prerequisite publications, and the related publications to determine which publications you might find helpful. After you determine the publications you need, see "Accessing publications online" on page vi for information about accessing publications online.

IBM Tivoli Directory Server library

The publications in the IBM Tivoli Directory Server library are:

IBM Tivoli Directory Server Version 6.0 Release Notes
Contains information about the new features in the IBM Tivoli Directory Server Version 6.0 release.

IBM Tivoli Directory Server Version 6.0 Installation and Configuration Guide
Contains complete information for installing the IBM Tivoli Directory Server client, server, and Web Administration Tool. Includes information about migrating from a previous version of IBM Tivoli Directory Server or SecureWay® Directory.

IBM Tivoli Directory Server Version 6.0 Performance Tuning Guide
Contains information about tuning your server for better performance.

IBM Tivoli Directory Server Version 6.0 Administration Guide
Contains instructions for performing administrator tasks through the Web Administration Tool and the command line.

IBM Tivoli Directory Server Version 6.0 Plug-ins Reference
Contains information about writing server plug-ins.

IBM Tivoli Directory Server Version 6.0 C-Client SDK Programming Reference
Contains information about writing Lightweight Directory Access Protocol (LDAP) client applications.

IBM Tivoli Directory Server Version 6.0 Problem Determination Guide
Contains information about possible problems and corrective actions that can be tried before contacting Software Support.

© Copyright IBM Corp. 2005

IBM Tivoli Directory Server Version 6.0 Messages
Contains information about error messages that you might see.

Related publications

Information related to IBM Tivoli Directory Server is available in the following publications:

- IBM Tivoli Directory Server Version 6.0 uses the JNDI client from Sun Microsystems. For information about the JNDI client, refer to the *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.
- The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: http://publib.boulder.ibm.com/tividd/td/tdprodlist.html
- The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available from the **Glossary** link on the left side of the Tivoli Software Library Web page http://publib.boulder.ibm.com/tividd/td/tdprodlist.html

Accessing publications online

The publications for this product are available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format, or both in the Tivoli software library: http://publib.boulder.ibm.com/tividd/td/tdprodlist.html

To locate product publications in the library, click the first letter of the product name or scroll until you find the product name. Then, click the product name.

Product publications include release notes, installation guides, user's guides, administrator's guides, and developer's references.

Note: To ensure proper printing of PDF publications, select the **Fit to page** check box in the Adobe Acrobat Print window (which is available when you click **File** → **Print**).

Ordering publications

You can order many Tivoli publications online at the following Web site:

http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi

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, see the following Web site for a list of telephone numbers:

http://www.ibm.com/software/tivoli/order-lit/

Accessibility

Accessibility features help a user who has 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 also can use the keyboard instead of the mouse to operate all features of the graphical user interface.

Tivoli software training

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

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:

- Searching knowledge bases: You can search across a large collection of known problems and workarounds, Technotes, and other information.
- Obtaining fixes: You can locate the latest fixes that are already available for your product.
- Contacting IBM Software Support: If you still cannot solve your problem, and you need to work with someone from IBM, you can use a variety of ways to contact IBM Software Support.

For more information about these three ways of resolving problems, see Appendix D, “Support information,” on page 59.

Conventions used in this book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

Typeface conventions

The following typeface conventions are used in this reference:

Bold Lowercase commands or mixed case commands that are difficult to distinguish from surrounding text, keywords, parameters, options, names of Java classes, and objects are in **bold**.

Italic Variables, titles of publications, and special words or phrases that are emphasized are in *italic*.

<*Italic*>
Variables are set off with < > and are in <*italic*>.

`Monospace`
Code examples, command lines, screen output, file and directory names that are difficult to distinguish from surrounding text, system messages, text that the user must type, and values for arguments or command options are in `monospace`.

Operating system differences

This book uses the UNIX® convention for specifying environment variables and for directory notation. When you are using the Windows® command line, replace `$variable` with `%variable%` for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. 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 6.0 application components

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

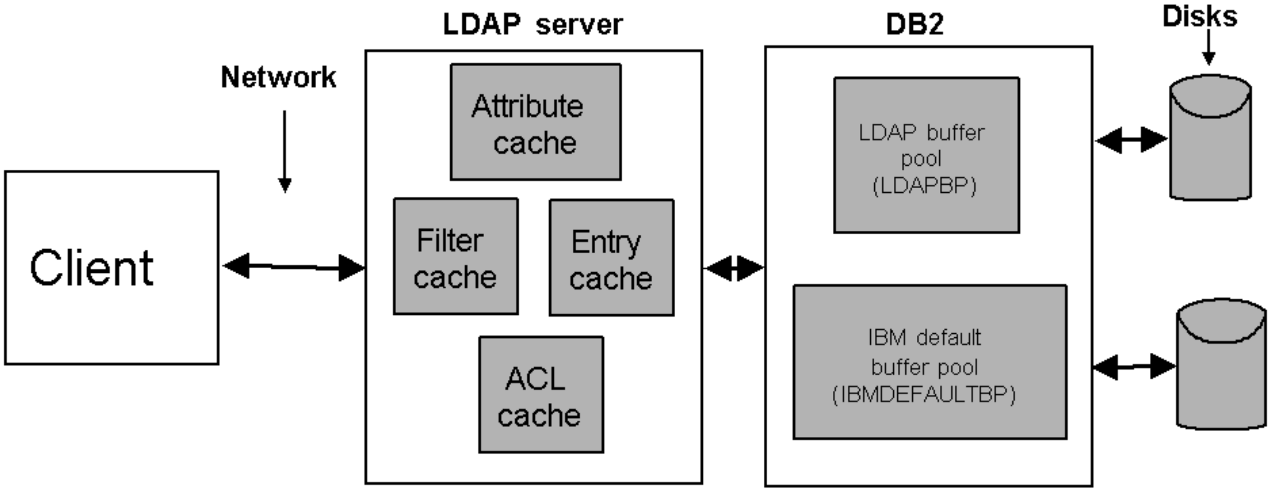

Figure 1. IBM Tivoli Directory Server 6.0

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.

© Copyright IBM Corp. 2005

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 DB2 buffer pools.

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 bufferpool 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
- LDAP filter cache
- Entry cache
- ACL cache

For more information on these caches, see "LDAP caches" on page 5.

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.

See "LDAPBP buffer pool size" on page 22 for more information.

IBMDEFAULTBP

DB2 system information, including system tables and other LDAP information, is cached in the IBMDEFAULTBP. You might need to adjust the IBMDEFAULTBP cache settings for better performance. See "IBMDEFAULTBP buffer pool size" on page 23 for more information.

Memory allocation between LDAP caches and buffer pools

The LDAP caches are generally more efficient as a means of caching LDAP searches; however, parts of the LDAP cache get invalidated on updates and must be reloaded before performance benefits return. Some experimentation between the two caching schemes is required to find the best memory allocation for your workload.

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.

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 5 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. Adjusting the size of the DB2 buffer pools can result in improved performance. See “DB2 buffer pool tuning” on page 22 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 (reorgchk and reorg)” on page 24 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 “Indexes” on page 30.

Attention: You should place the DB2 log on a physical disk drive separate from the data. For improved data-integrity, 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, “DB2 tuning and commands,” on page 21.

Generic LDAP application tips

The following are some tips that can help improve performance:

- Perform searches on indexed attributes only. See “Indexes” on page 30 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. A base-scoped search is a search done using the ldapsearch utility, where the scope of the search is defined as a base object.
- 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 the Proxy server in this release, 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

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, the attribute cache manager resolves the search operation 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 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 resolve complex filters only if they are conjunctive. In addition, the subfilters within the complex filters must be of the following types:

  – exact match filters

  – presence filters

© Copyright IBM Corp. 2005

  - conjunctive 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 44.

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 shows varying operations per second based on different filter cache sizes for one installation:

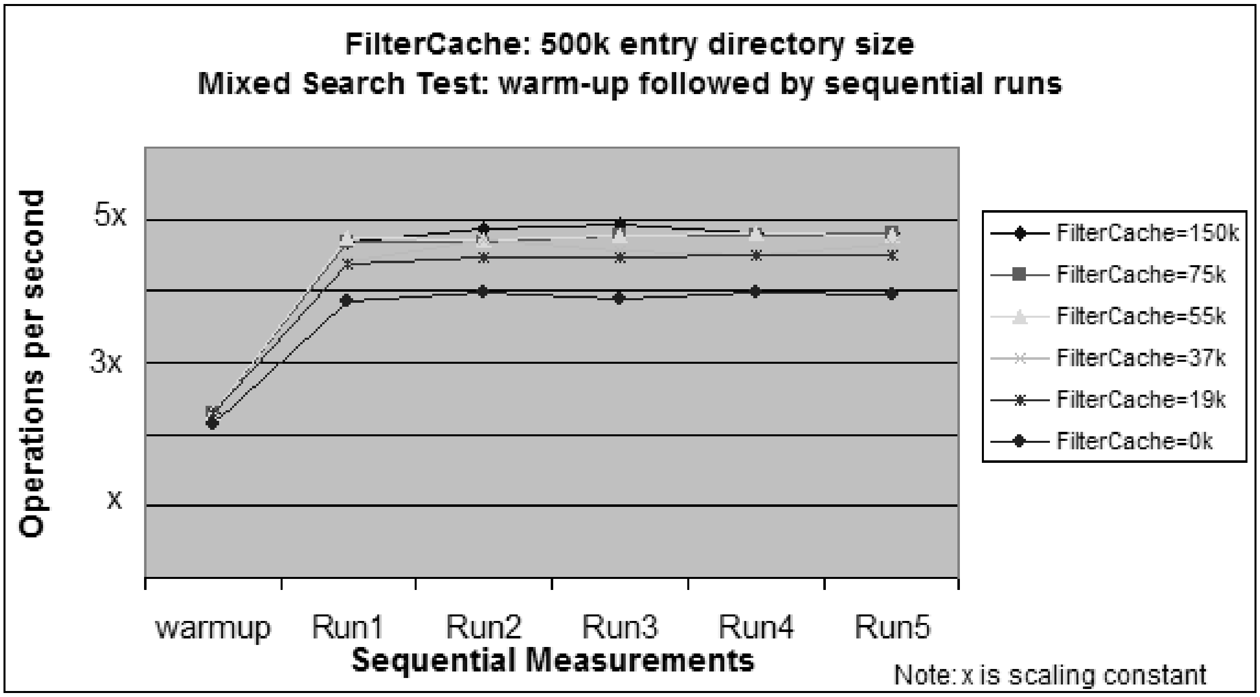

Figure 2. Varying the size of the filter cache

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 13 to set the filter cache size.

Filter cache size with updates

Figure 3 on page 10 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 and buffer pools. See “LDAP cache configuration variables” on page 13 for instructions on how to set configuration variables such as filter cache size.

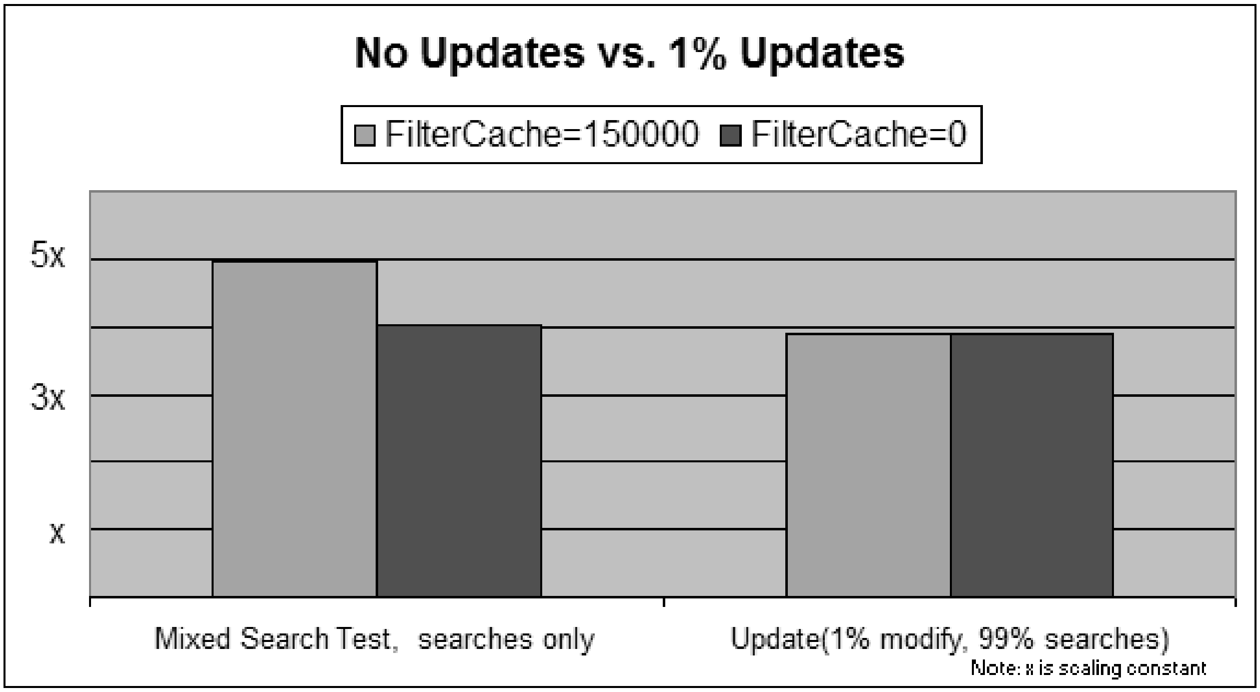

Figure 3. Effect of updates on the performance of the filter cache

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. To determine the best filter cache bypass limit for your workload, repeatedly run your workload with the filter cache bypass limits set to different values and measure the operations per second.

For example, Figure 4 on page 11 shows operations per second based on varying cache bypass limit sizes:

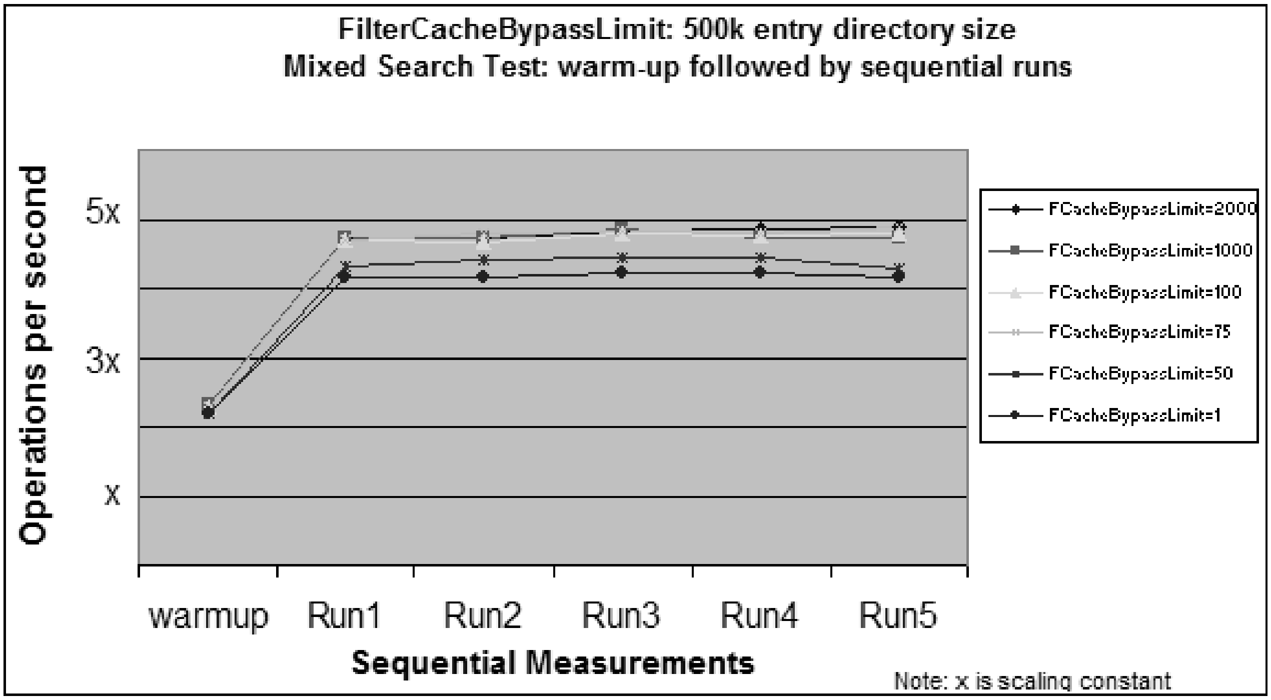

Figure 4. Varying the filter cache bypass limit

For the workload in Figure 4, setting the limit too low downgrades performance by preventing valuable filters from being cached. Setting the filter bypass limit to approximately 100 appears to be the best size for this workload. Setting it any larger benefits performance only slightly.

See “LDAP cache configuration variables” on page 13 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 5 on page 12 shows varying operations per second based on different entry cache sizes:

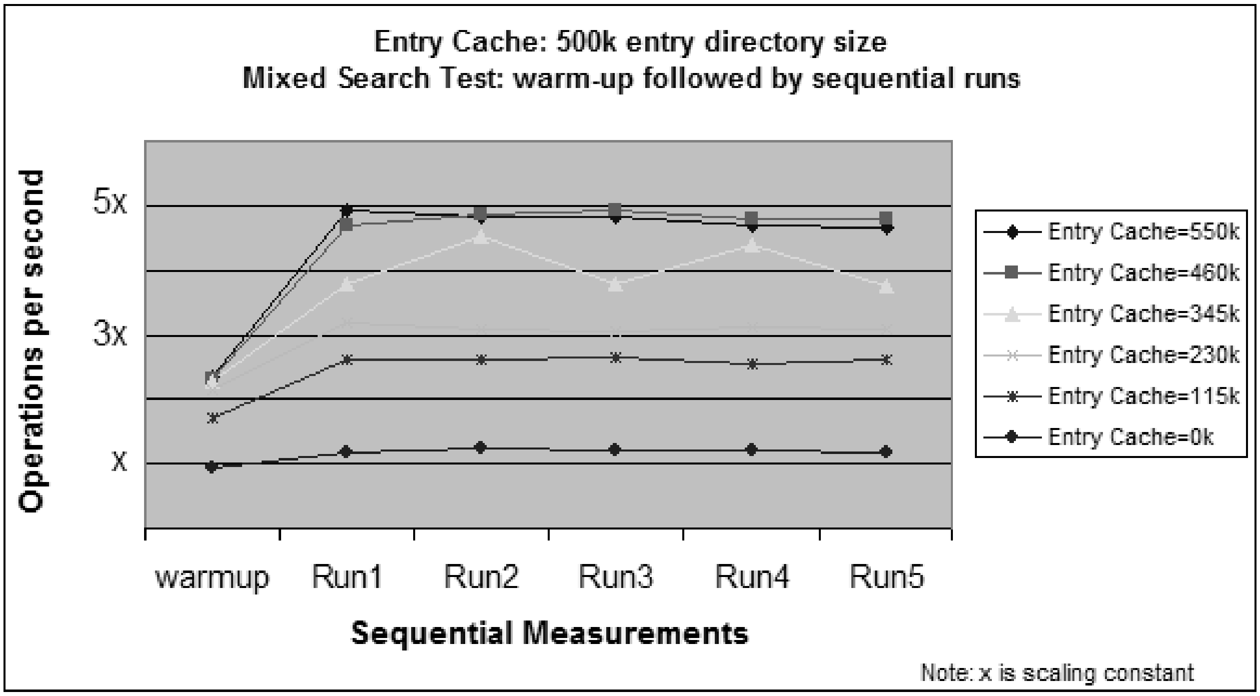

Figure 5. Varying the size of the entry cache

From the results in Figure 5, 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 13 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.

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.

Using the Web Administration Tool

To configure the attribute cache using the Web Administration Tool:

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

2. You can change the amount of memory available to the directory attribute cache by changing the **Directory cached attribute size (in kilobytes)** field. The default is 16,384 KB (16 MB).
3. You can change the amount of memory available to the changelog attribute cache by changing the **Changelog cached attribute size (in kilobytes)** field. The default is 16,384 KB (16 MB).

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

To add attributes to the attribute cache:

1. Select the attribute that you want to add as a cached attribute from the **Available attributes** menu. Only those attributes that can be cached 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 cn=directory and the cn=changelog containers.
2. Click either **Add to cn=directory** or **Add to cn=changelog**. The attribute is displayed in the appropriate list box. You can list the same attribute in both containers.

   Note: **Add to cn=changelog** is disabled if a changelog has not been configured.
3. Repeat this process for each attribute you want to cache.
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 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 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 6.0 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 11 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 9 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 10 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 6.0 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.

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 6 on page 17, it appears that performance degrades significantly as the database size grows.

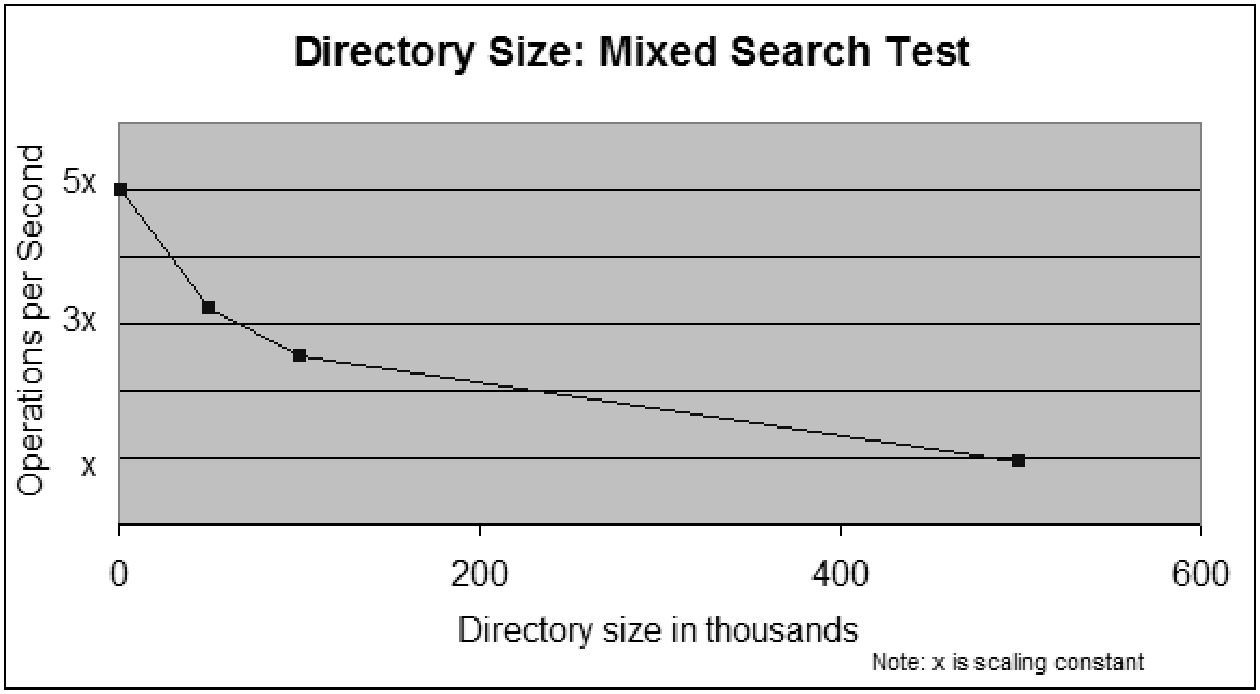

Figure 6. Operations per second

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 7 on page 18 shows, as the size of the directory grows, so does the number of entries returned in response to wildcard and exact-match search requests.

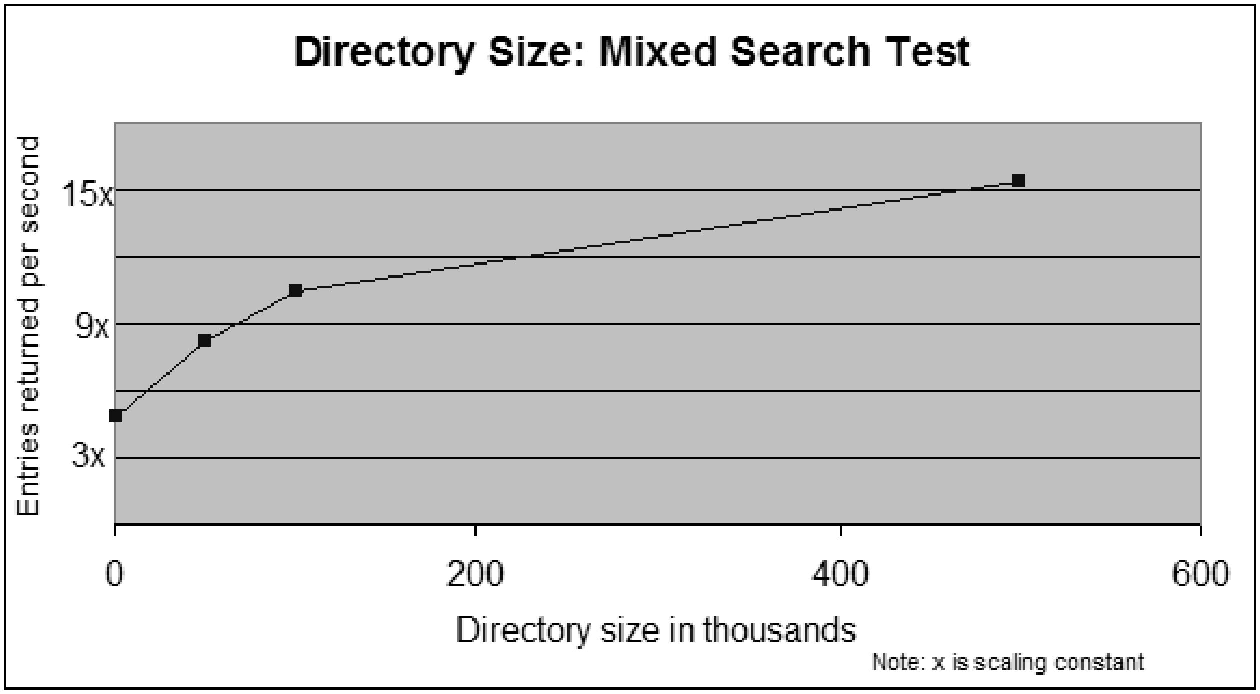

Figure 7. Entries returned per second

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 and DB2 buffer pools. You can determine the optimal sizes for your caches and buffer pools using the guidelines in “LDAP caches” on page 5 and “DB2 buffer pool tuning” on page 22.

objectclass table indexes on Linux

When migrating from an earlier release, dropping the index for the objectclass table and creating a new index with a unique flag can improve performance.

When the IBM Tivoli Directory Server 6.0 server is started for the first time, the index for the objectclass table is dropped (if the index exists) and a new index with a unique flag is created. If the index does not already exist, then the index for the objectclass table is created with the unique flag.

Note: This happens automatically on all platforms except Linux.

It is not possible to create the unique index for the objectclass table on Linux systems with data from previous versions of IBM Tivoli Directory Server. This is due to the fact that there are duplicate entries in the objectclass table for previous versions of IBM Tivoli Directory Server on Linux. The following is an example entry from an IBM Tivoli Directory Server 5.2 database on a Linux system:

EID	OBJECTCLASS	ROBJECTCLASS
5	CONTAINER	RENIATNOC
5	PWDPOLICY	YCILOPDWP

5	IBM-PWDPOLICYEXT	TXEYCILOPDWP-MBI
5	TOP	POT
5	TOP	POT
5	TOP	POT

Notice that there are three entries for the entry ID 5 and objectclass TOP.

To create the unique index, all duplicate rows must be eliminated. Do the following to manually delete duplicate rows in the objectclass table:

1. Export the distinct rows from the objectclass table to a file:
   ```
   db2 export to /tmp/db2.out of del select distinct "*" from objectclass
   ```
2. Remove all the data from the objectclass table:
   ```
   db2 delete from objectclass
   ```
3. Insert the distinct rows back into the objectclass table:
   ```
   db2 import from /tmp/db2.out of del insert into objectclass
   ```

You can also reload the data in the IBM Tivoli Directory Server 6.0 database, which eliminates the duplicate entries.

When duplicate rows in the objectclass table are completely removed, the objectclass index can be dropped and the unique index added.

Use the following to create the unique index:

```
create unique index LDAPDB2.OBJECTCLASS ON LDAPDB2.OBJECTCLASS
   ("EID" ASC,
   "OBJECTCLASS" ASC);
```

Note: The order of the key pairs is significant, for example, "EID" first, followed by "OBJECTCLASS".

Chapter 3. DB2 tuning and commands

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.

This chapter covers the following types of DB2 tuning:

- DB2 buffer pool tuning
- Optimization and organization (**reorgchk** and **reorg**)
- Other DB2 configuration parameters
- Backing up and restoring the database (**backup** and **restore**)

For detailed information about DB2 commands, see the DB2 documentation at the following Web site: http://www.ibm.com/software/data/db2/library/

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.

Note: 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.

© Copyright IBM Corp. 2005

For additional stability and performance enhancements, upgrade to the latest version of DB2.

DB2 buffer pool tuning

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. Buffer pool tuning typically needs to be done when the database is initially loaded and when the database size changes significantly.

There are several considerations to keep in mind when tuning the 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.

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

```
db2 connect to database_name
db2 "select bpname,npages,pagesize from sysibm.sysbufferpools"
```

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.
```

Buffer pool sizes

In IBM Tivoli Directory Server 6.0, the LDAP directory database (DB2) has two buffer pools: LDAPBP and IBMDEFAULTBP. The size of each buffer pool needs to be set separately, but the method for determining how big each should be is the same: Run your workload with the buffer pool sizes set to different values and measure the differences in operations per second.

Note: DB2 does not allow buffer pools to be set to zero.

LDAPBP buffer pool size

This buffer pool 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 to cache. It is possible that an entry that is not cached in the entry cache is located in LDAPBP.

To determine the best size for your LDAPBP buffer pool, run your workload with the LDAPBP buffer pool size set to different values and measure the differences in operations per second. For example, Figure 8 shows varying operations per second based on different LDAPBP buffer pool sizes:

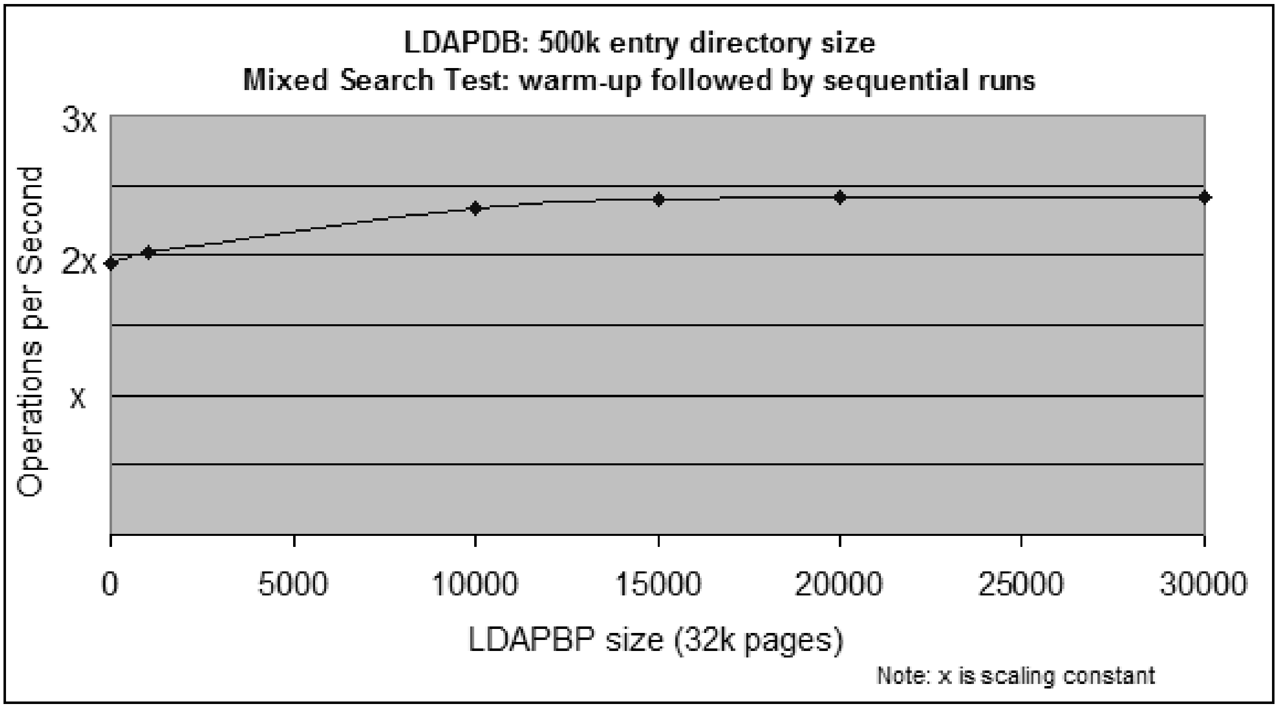

Figure 8. Varying the size of LDAPBP

For the workload in the above example, the best performance results from a LDAPBP size of approximately 15,000 32K pages. However, the performance gain of 15,000 over a size of 9,800 is slight. In a memory-constrained environment, setting the LDAPBP size to 9,800 saves approximately 166 MB of memory.

IBMDEFAULTBP buffer pool size

DB2 system information, including system tables and other information that is useful in resolving filters, is cached in the IBMDEFAULTBP buffer pool. You might need to adjust the IBMDEFAULTBP cache settings for better performance in the LDAPBP.

To determine the best size for your IBMDEFAULTBP buffer pool, run your workload with the buffer pool sizes set to different values and measure the differences in operations per second. For example, Figure 9 on page 24 shows varying operations per second based on different IBMDEFAULTBP buffer pool sizes:

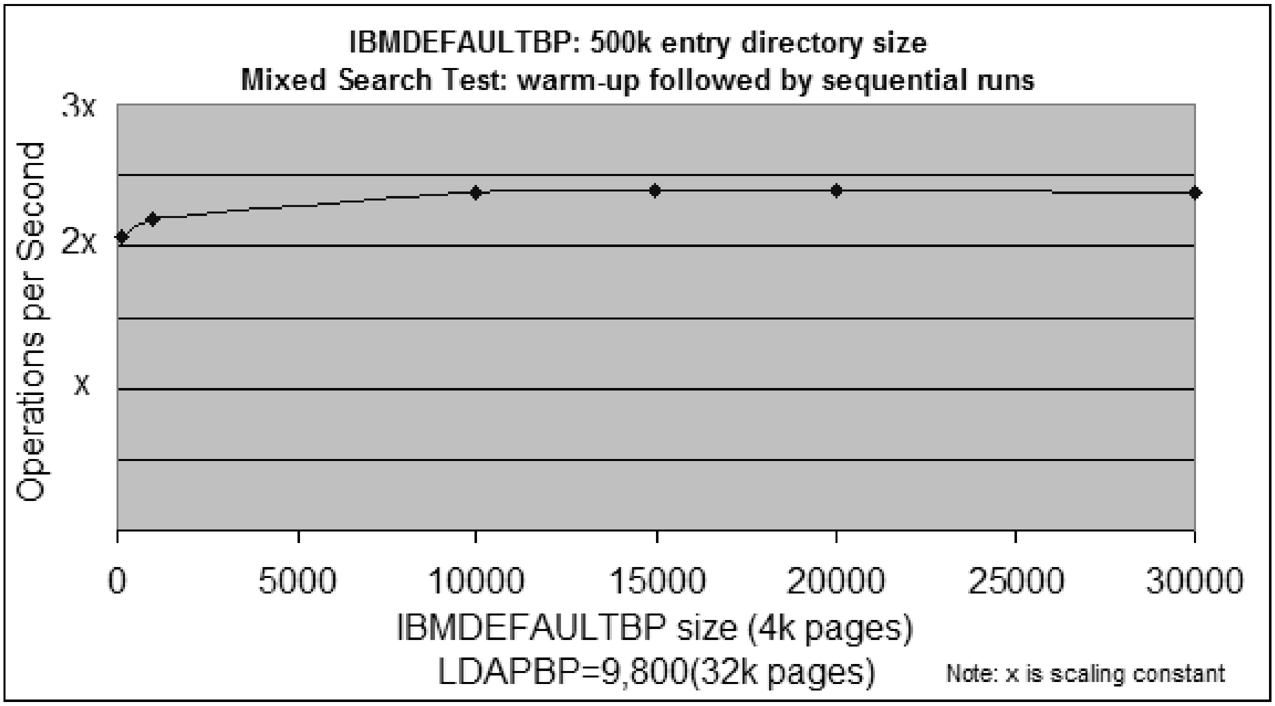

Figure 9. Varying the size of IBMDEFAULTBP

For the workload in the above example, setting the IBMDEFAULTBP large enough to hold the working set improves throughput approximately 20 percent over a small buffer pool size. There is little additional benefit to setting IBMDEFAULTBP larger than 20,000 4K pages.

Setting buffer pool sizes

Use the `alter bufferpool` command to set the IBMDEFAULTBP and LDAPBP buffer pool sizes. The following example shows the IBMDEFAULTBP and LDAPBP buffer pools being set:

```
db2 alter bufferpool ibmdefaultbp size 20000
db2 alter bufferpool ldapbp size 9800
db2 force applications all
db2stop
db2start
```

Note: The LDAP server (idsslapd) must be stopped while setting buffer pool sizes.

Optimization and organization (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 DB2 **runstats** 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)” 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.

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

```
runstats -I <instancename>
```

See "idsrunstats, runstats" in the *IBM Tivoli Directory Server Version 6.0 Administration Guide* for more information.

Run the following commands to update more db2 stats that might improve performance:

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

where *table_name* is the name of the table. You can use ALL for all tables.

Note: Use the runstats utility to update the database statistics. This utility preserves some LDAP-specific tuning done on statistics. If you use the DB2 RUNSTATS command, do the following to restore the LDAP-specific settings:

```
db2 "connect to database <ldap_db_name>"
db2 "update sysstat.tables set card=9E18 where tabname='LDAP_DESC'
  and card<>9E18"
db2 "terminate"
```

Database organization (reorgchk and reorg)

Tuning the organization of the data in DB2 using the **reorgchk** and **reorg** commands is important for optimal performance.

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, using the data generated by **reorgchk**, reorganizes table spaces to improve access performance and reorganizes indexes so that they are more efficiently clustered. The **reorgchk** and **reorg** commands can improve both search and update operation performance.

Note: Tuning organizes the data on disk in a sorted order. Sorting the data on disk is beneficial only when accesses occur in a sorted order, which is not typically the case. For this reason, organizing the table data on disk typically yields little change in performance.

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.
```

Note: After performing a reorgchk, do the following to restore the LDAP-specific settings:

```
db2 "connect to database <ldap_db_name>"
db2 "update sysstat.tables set card=9E18 where tabname='LDAP_DESC'
  and card<>9E18"
db2 "terminate"
```

See "idsrunstats, runstats" in the *IBM Tivoli Directory Server Version 6.0 Administration Guide* for more information.

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 44 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. This can take a long time. The time it takes to perform the reorganization process increases as the DB2 database size increases.

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

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.

   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 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, SYSIBM.SQL000414155358130.

4. Run **reorgchk** again. The output from **reorgchk** can then be used to determine whether the reorganization worked and whether it introduced other tables and indexes that need reorganizing.

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 attributes that you want to use in searches. In most cases you will want to reorganize to the forward index, but in cases with searches beginning with '*', reorganize to the reverse index.
- 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
```

Indexes

Indexing results in a considerable reduction in the amount of time it takes to locate requested data. For this reason, it can be very beneficial from a performance standpoint to index all attributes used in searches.

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

```
IndexSchema      Index Name                  Unique Rule          Number of Columns
 -------------  -------------------          ----------           -------------
LDAPDB2         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:

  ```
  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 )
```

Note: After adding an index, you should run reorgchk to update statistical information for the DB2 optimizer regarding Index statistics for the new index.

Other DB2 configuration parameters

Performance benefits can come from setting other DB2 configuration parameters, such as APPLHEAPZ and LOGFILSIZ. 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.

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
```

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.

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 a list of DB2 parameters that affect performance, visit the DB2 Web site: http://www.ibm.com/software/data/db2

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

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.

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 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 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.

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.

© Copyright IBM Corp. 2005

Setting other environment variables

You might experience better performance 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
```

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/V6.0
LANG=en_US
LC__FASTMSG=true
LD_LIBRARY_PATH=/home/ldapdb2/sqllib/lib
LIBPATH=/opt/IBM/ldap/V6.0/lib64:/usr/lib:/home/ldapdb2/idsslapd-ldapdb2/
 db2instance/lib:/opt/IBM/ldap/V6.0/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/V6.0/nls/msg/%L/%N:/opt/IBM/ldap/V6.0/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/V6.0/sbin:/opt/IBM/ldap/V6.0:/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/V6.0
TZ=CST6CDT
USER=root
VWSPATH=/home/ldapdb2/sqllib
_=/opt/IBM/ldap/V6.0/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 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

© Copyright IBM Corp. 2005

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.

© Copyright IBM Corp. 2005

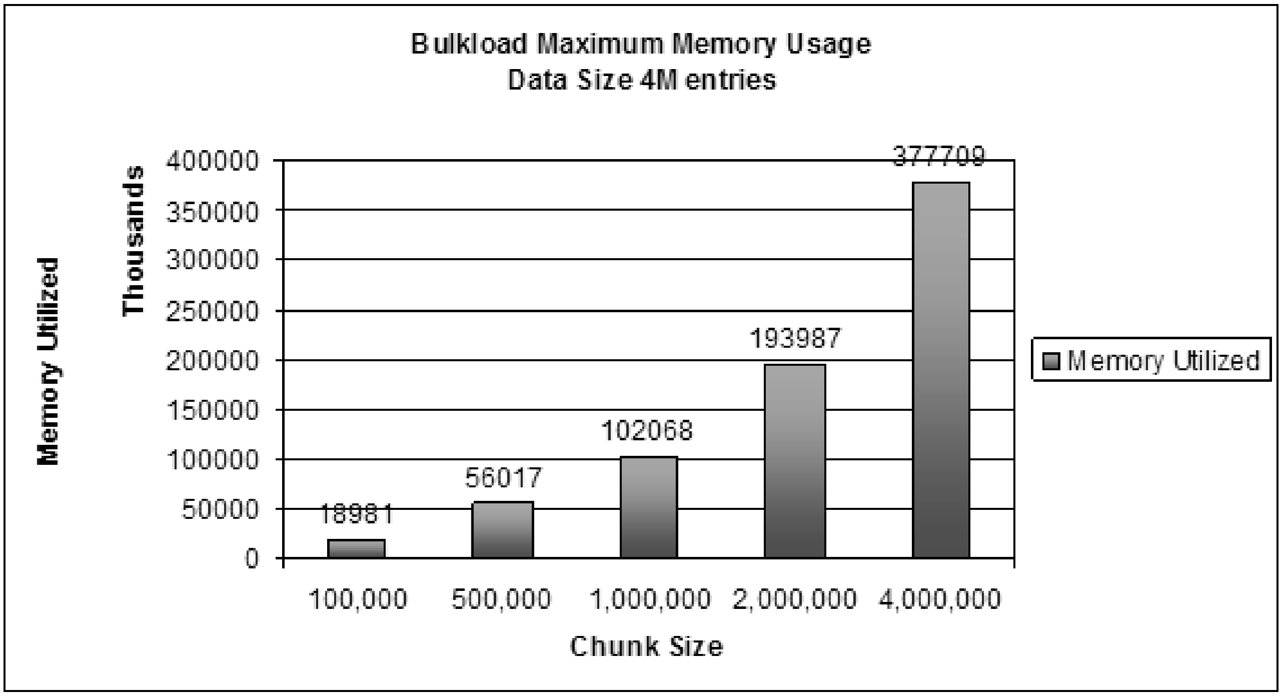

Figure 10. Effects of using the -k option

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

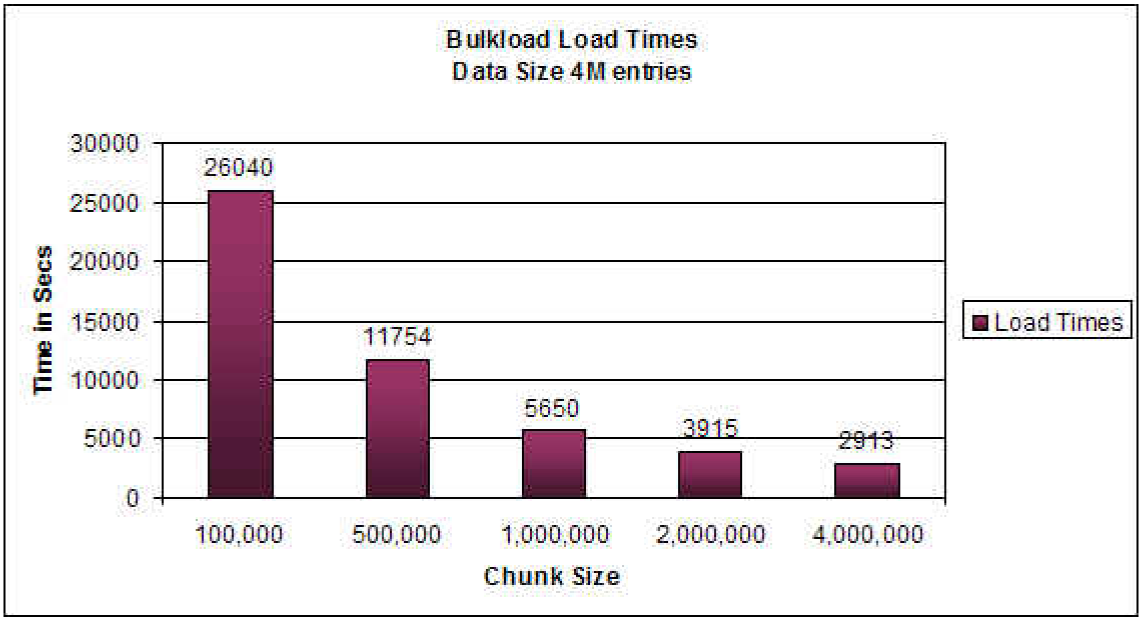

Figure 11. Effects of using the -k option

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 6.0 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
 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 6.0 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 executed 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.

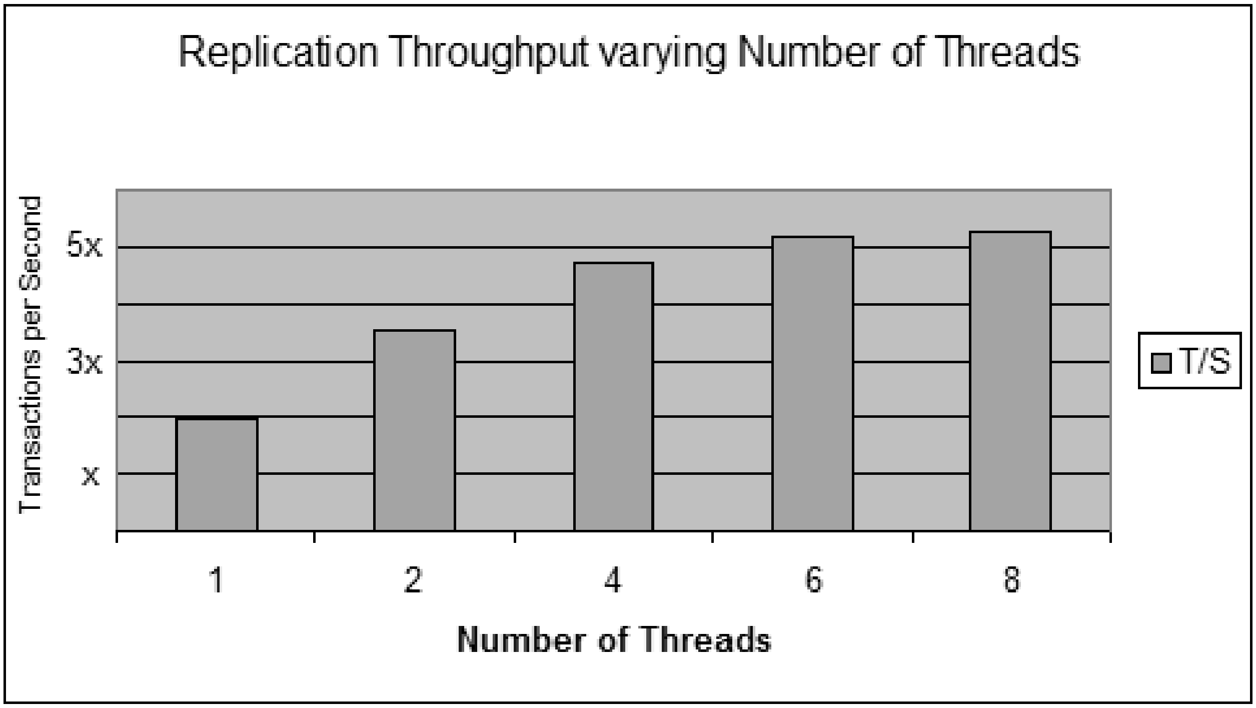

Figure 12. Number of replication threads

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.

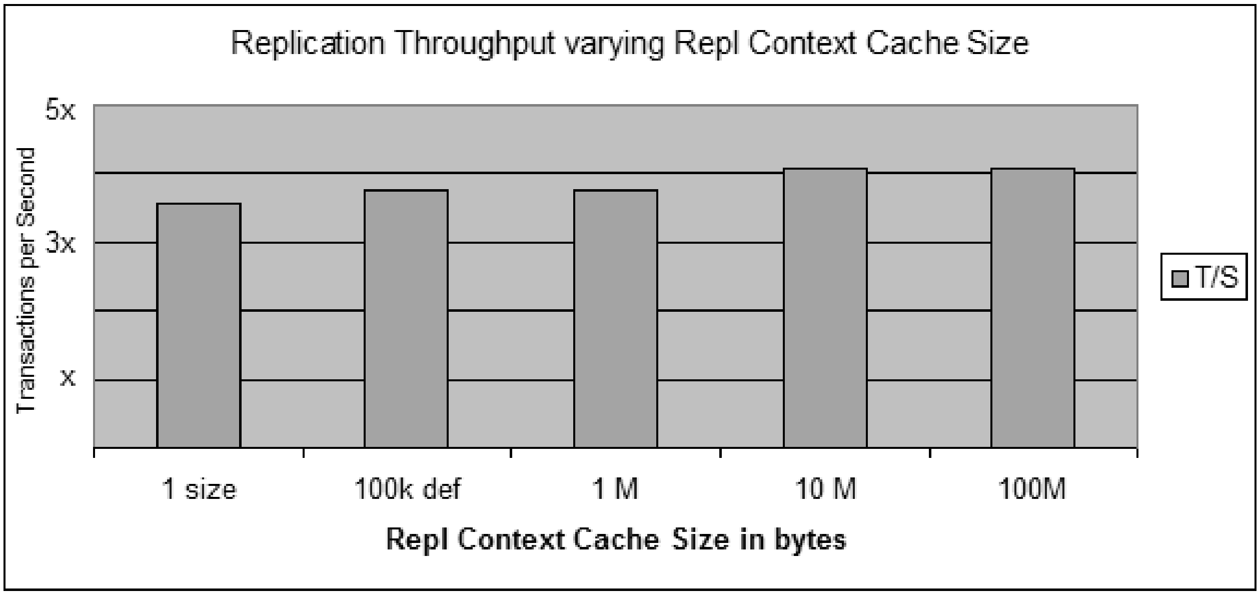

Figure 13. Replication context cache size

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.

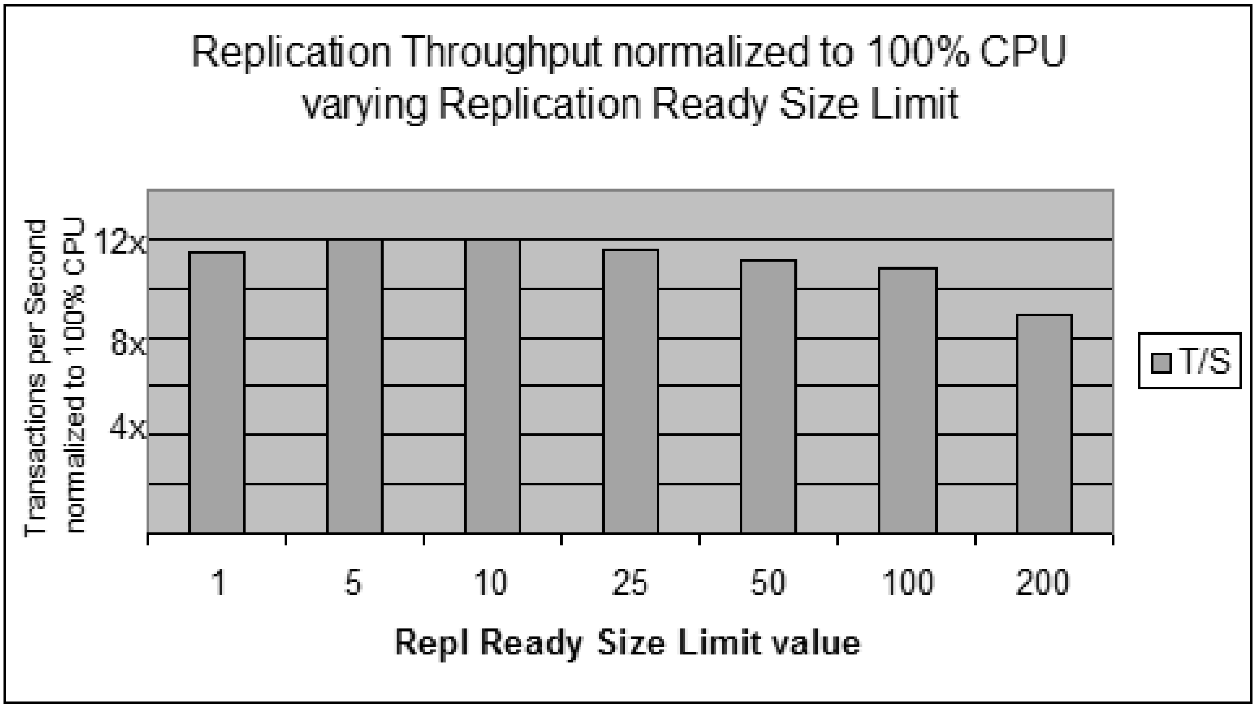

Figure 14. Replication ready size limit

Proxy server tuning

The proxy server is recommended for use 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. The proxy server gives customers the ability to distribute data across multiple backend servers.

Throughput performance of the proxy server can be affected by the size of the connection pool. This is a parameter configured on the Proxy server. It is recommended that you configure more than one connection to the backend server.

For better performance, all backend 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 6.0

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.

emergency_thread_running
The indicator of whether the emergency thread is running.

totaltimes_emergency_thread_run
The number of times the emergency thread has been activated.

lasttime_emergency_thread_run
The last time the emergency thread was activated.

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 6.0 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 6.0 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 the LDAP change log” on page 51 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 the LDAP change log

IBM Tivoli Directory Server 6.0 has a function called **change log** that results in a significantly slower LDAP update performance. The **change log** function should be configured only if needed.

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

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

Appendix A. Workload description

The tests used in this document contain a mixture of searches and binds, including wildcard searches, which return multiple entries.

Each scenario consists of two phases, a warmup phase and a run phase. During the warmup phase, the searches primarily request entries that are not in the LDAP caches; most of these requests require interaction with the DB2 backing store. For all the measurements reported in this document, warmup 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 warmup phase and the run phase comprise two distinctly different workloads.

During the run phase of the mixed search and bind test, a number of client threads issue search requests to the IBM Tivoli Directory Server from predetermined scripts. The scripts include a number of different kinds of searches, including wildcard and other searches that return multiple entries per request. The client threads run through their scripts continuously for three minutes. Throughput is measured on the server for each three-minute interval, and then each client starts over at the beginning of its script. Each three-minute interval is referred to as a run. The server is not restarted between runs.

© Copyright IBM Corp. 2005

Appendix B. Modifying TCP/IP settings

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 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.

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). **tcp_keepidle** is a runtime attribute.

   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. **tcp_keepinit** 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). **tcp_keepintvl** is a runtime attribute.

Note: This applies to both client and server machines.

© Copyright IBM Corp. 2005

Appendix C. Platform configurations

The examples in this guide use the following platform configurations:

- Clients
  - Four 1.8GHz, 512MB RAM, Intel PRO/100 VE
  - Windows 2000 Professional with SP2
- Server
  - 4-Way 450MHz, pSeries(TM) eServer(TM) Model 7026-B80 with 1 or 4 processors active, 4 GB RAM.
  - IBM 10/100 Ethernet Adapter.
  - AIX 5.2
  - IBM Tivoli Directory Server Version 6.0
  - 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 8.1.1.16
  - 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 50 client threads except where noted.
  - DB2 log files are not on the same disk as the containers.

© Copyright IBM Corp. 2005

Appendix D. Support information

This section describes the following options for obtaining support for IBM products:

- "Searching knowledge bases"
- "Obtaining fixes"
- "Contacting IBM Software Support" on page 60

Searching knowledge bases

If you have a problem with your IBM software, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the resolution to your problem is already documented.

Search the information center on your local system or network

IBM provides extensive documentation that can be installed on your local computer or on an intranet server. You can use the search function of this information center to query conceptual information, instructions for completing tasks, reference information, and support documents.

Search the Internet

If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem. To search multiple Internet resources for your product, expand the product folder in the navigation frame to the left and select **Web search**. From this topic, you can search a variety of resources including:

- IBM technotes
- IBM downloads
- IBM Redbooks™
- IBM developerWorks®
- Forums and newsgroups
- Google

Obtaining fixes

A product fix might be available to resolve your problem. You can determine what fixes are available for your IBM software product by checking the product support Web site:

1. Go to the IBM Software Support Web site (http://www.ibm.com/software/support).
2. Under **Products A - Z**, select your product name. This opens a product-specific support site.
3. Under **Self help**, follow the link to **All Updates**, where you will find a list of fixes, fix packs, and other service updates for your product. For tips on refining your search, click **Search tips**.
4. Click the name of a fix to read the description and optionally download the fix.

© Copyright IBM Corp. 2005

To receive weekly e-mail notifications about fixes and other news about IBM products, follow these steps:

1. From the support page for any IBM product, click **My support** in the upper-right corner of the page.
2. If you have already registered, skip to the next step. If you have not registered, click register in the upper-right corner of the support page to establish your user ID and password.
3. Sign in to **My support**.
4. On the My support page, click **Edit profiles** in the left navigation pane, and scroll to **Select Mail Preferences**. Select a product family and check the appropriate boxes for the type of information you want.
5. Click **Submit**.
6. For e-mail notification for other products, repeat Steps 4 and 5.

For more information about types of fixes, see the *Software Support Handbook* (http://techsupport.services.ibm.com/guides/handbook.html).

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus®, and Rational® products, as well as DB2 and WebSphere® products that run on Windows or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:
  - **Online**: Go to the Passport Advantage Web page (http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home) and click **How to Enroll**
  - **By phone**: For the phone number to call in your country, go to the IBM Software Support Web site (http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries®, pSeries™, and iSeries™ environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web page (http://www.ibm.com/servers/eserver/techsupport.html).

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the IBM Software Support Handbook on the Web (http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region for phone numbers of people who provide support for your location.

Follow the steps in this topic to contact IBM Software Support:

1. Determine the business impact of your problem.
2. Describe your problem and gather background information.

3. Submit your problem to IBM Software Support.

Determine the business impact of your problem

When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

Severity 1	**Critical** business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
Severity 2	**Significant** business impact: The program is usable but is severely limited.
Severity 3	**Some** business impact: The program is usable with less significant features (not critical to operations) unavailable.
Severity 4	**Minimal** business impact: The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented.

Describe your problem and gather background information

When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be re-created? If so, what steps led to the failure?
- Have any changes been made to the system? (For example, hardware, operating system, networking software, and so on.)
- Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.

Submit your problem to IBM Software Support

You can submit your problem in one of two ways:

- **Online**: Go to the ″Submit and track problems″ page on the IBM Software Support site (http://www.ibm.com/software/support/probsub.html). Enter your information into the appropriate problem submission tool.
- **By phone**: For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web (techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround for you to implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

For more information about problem resolution, see Searching knowledge bases and Obtaining fixes.

Appendix E. Notices

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

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

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

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

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

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

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

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

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

© Copyright IBM Corp. 2005

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

IBM Corporation
Department MU5A46
11301 Burnet Road
Austin, TX 78758
U.S.A.

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

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

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

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

AIX
Database 2
DB2
developerWorks
eServer
IBM
Lotus
Notes
Passport Advantage
pSeries
Rational
Redbooks
SecureWay
Tivoli
WebSphere
zSeries

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows is a registered trademarks of Microsoft Corporation

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

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

Index

© Copyright IBM Corp. 2005

Printed in USA

SC32-1677-00