IBM Security Directory Suite
8.0.1

Performance Tuning and Capacity Planning Guide

IBM

Note

Before using this information and the product it supports, read the general information under "Notices" on page 93.

Edition notice

Note: This edition applies to version 8.0.1.x of *IBM Security Directory Suite* (product number 5725-Y17) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

IBM® Security Directory Suite, previously known as IBM Security Directory Server or IBM Tivoli® Directory Server, is an IBM implementation of the Lightweight Directory Access Protocol.

IBM Security Directory Suite Performance Tuning and Capacity Planning Guide contains information about tuning the Directory Server for better performance.

Accessibility

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

For more information, see "Accessibility features for IBM Security Directory Suite" in the IBM Knowledge Center.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection, and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated, or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. Directory Server performance tuning and capacity planning

You must tune IBM Security Directory Suite Directory Server and the associated DB2® database to obtain optimum server performance.

IBM Security Directory Suite Directory Server is a Lightweight Directory Access Protocol (LDAP) directory that provides a layer on top of DB2. You can use the Directory Server to efficiently organize, manipulate, and retrieve data that are stored in the DB2 database. Tuning for optimal performance is primarily a matter of configuring the relationships between a Directory Server and DB2 according to the nature of your workload.

Based on the directory environment, each workload might be different. Therefore, instead of providing exact values for tuning settings, guidelines are provided, where appropriate, to determine the best settings for your system to obtain optimal performance.

Attention:

The examples performance test results that are provided are captured in a lab environment. The workload that is used for the test includes a mixture of searches and binds, including wildcard searches that return multiple entries. Your results might differ from the lab results shown in this document.

In the examples that are provided, the *instance_name* variable might be included. This variable is not applicable for the Directory Server virtual appliance stand-alone system, which comes with an embedded instance of DB2, where the instance name is `sdsinst1`. If you configured a remote DB2 for use with the Directory Server instance in virtual appliance, you might replace the *instance_name* variable with the instance name that is applicable in the commands.

Use the **tools** > **db2cmd** utility from virtual appliance command-line interface to run all DB2 commands on virtual appliance.

```
sdsva.example.com> tools db2cmd
```

The DB2 command-line processor is opened.

```
db2 =>
```

Directory Server application components

The Directory Server interacts with various application components to query and retrieve data. You must tune each application component for optimal performance.

The following figure illustrates how the Directory Server components interact with each other. Tuning these components results in improved performance.

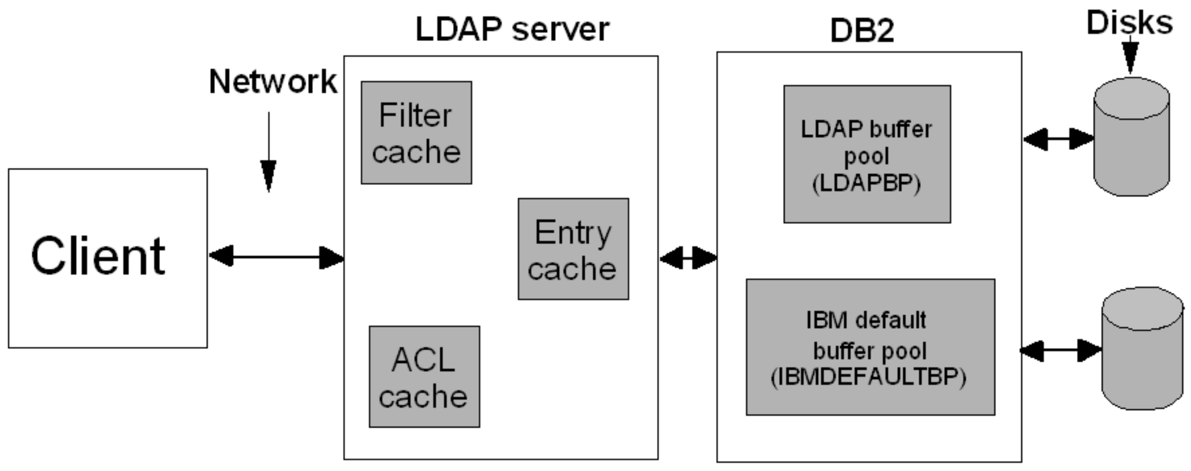

Figure 1. Directory Server components

The arrows in the figure represent the interaction path for a query that is issued from a client computer. The query follows a path from the Directory Server client to the server, to DB2, to the physical disks in search of the entries that meets the search filter criteria. The shorter the path to matching entries, the better overall performance you can expect from your system.

For example, if a query locates all the matching entries in a server, access to DB2 and the disks is not necessary. If the matching entries are not found in the server, the query continues on to DB2 and, if necessary, to the physical disks. From a performance standpoint, it is better to allocate a significant amount of memory to the server caches and setting the DB2 buffer pools to AUTOMATIC. Otherwise, the server might take considerable amount of time and resources to retrieve data from disk.

LDAP caches and DB2 buffer pools

You can use the LDAP caches and DB2 buffer pools to cache the previously accessed data. The cached data improves the performance of a Directory Server by reducing the disk access.

When the 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 in a cache or buffer pool.

The information type that is stored in each cache and buffer pool is different. To obtain better results, you must understand how and when each cache and buffer pool is accessed.

LDAP caches

You can load an LDAP cache with the LDAP data for retrieving the data faster.

Search operations attempt to use one or more caches to resolve the search filter and returning the matching entries. Most base-scoped searches can be resolved directly in memory. You can retrieve the base entry from the entry cache or the database buffer pool and compare the entry with the filter.

If a base-scoped search is not resolved directly in memory or the search is not base-scoped, the following options are checked.

- Use the attribute cache to resolve the filter in memory.
- Use the filter cache to retrieve the results of a previously run search operation.

If LDAP caches does not resolve the filter, the filter is resolved by using DB2. When entries are returned to the client, they are retrieved from memory by using the entry cache, if possible. If entries are not found in the entry cache, they are retrieved from DB2.

The following four LDAP caches are used in the Directory Server:

- LDAP attribute cache

Note: Attribute cache is deprecated. You must avoid use of attribute cache when you tune a Directory Server.

- LDAP filter cache
- Entry cache
- ACL cache

For more information about the LDAP caches, see "LDAP caches" on page 20.

DB2 buffer pools

A Directory Server uses the DB2 buffer pools to store cached data and to improve database performance.

A buffer pool is associated with a single database and can be used by more than one table space. Adequate buffer pool size is essential for good database performance because it reduces disk I/O, which uses a considerable amount of time.

The Directory Server uses 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 to determine which entries are cached. It is possible that an entry that is not cached in the entry cache is in LDAPBP. If the requested data is not found in the entry cache or LDAPBP, the query must access the physical disks.

IBMDEFAULTBP
When you create a database, a default buffer pool named IBMDEFAULTBP is created, which is shared by all table spaces. DB2 system information, including system tables and other LDAP information, is cached in the IBMDEFAULTBP.

Directory Server tuning overview

You can use the Directory Server performance statistics to decide when to tune a Directory Server.

You can store the frequently used LDAP data in the caches to tune an LDAP server to improve the performance of the server. It is important to remember that tuning an LDAP server alone is insufficient. You must also tune DB2 for optimal performance.

To get best results, it is advisable to tune a Directory Server instance immediately after the instance is configured. If a directory server is not tuned, it is likely to run poorly as the size of the instance grows. The deciding factors that you must consider for tuning a Directory Server instance.

- Poor or slow search and update response time.
- Considerable execution time for the **`bulkload`** command.

To improve poor performance, tune the Directory Server instance after you load data on the server. Later, update the DB2 system statistics after any large number of updates to the server.

Before you tune a Directory Server, you must take backup of the instance, database, and configuration files. For more information about Directory Server backup and restore, see the usage of **`idsdbback`** and **`idsdbrestore`** commands in the *Command Reference* section of the IBM Security Directory Suite documentation.

The fastest way to load and tune a directory server with millions of entries is to configure the server, tune the server, use the **`bulkload`** command to load the entries into the server, and finally update the DB2 system statistics.

The most significant performance tuning related to the IBM Security Directory Suite involves the LDAP caches. LDAP caches are fast storage buffers in memory that stores LDAP information such as queries, results, and user authentication for future use. While LDAP caches are useful mostly for applications that frequently retrieve repeated cached information, they improve performance by avoiding calls to the database. For information about how to tune the LDAP caches, see "LDAP caches" on page 20 .

DB2 tuning overview

You must identify the DB2 configuration parameters that you must tune to improve DB2 database performance.

DB2 serves as the data storage component of the Directory Server. You can improve the performance of Directory Server by tuning DB2 database that is associated with the instance.

You can use the following suggestions for tuning DB2:

- *DB2 buffer pools:* DB2 buffer pools are DB2 data caches. Each buffer pool is a data cache between the applications and the physical database files. Buffer pools are automatically tuned by default.For information about buffer pool, see "DB2 buffer pool " on page 31.
- *Optimization and organization:* After a directory server is loaded with data or after updates, it is important to update database statistics and organization for DB2 to run optimally. For more information, see "Optimization and organization of database " on page 49.
- *Indexes:* Indexes help locate data on disk in lesser time, providing a significant boost to performance. For information about how to create indexes, see "DB2 indexes" on page 56.

After the DB2 database is restored, you must retune the server unless the optimizations on a backed up DB2 database are known. When you run DB2 backup, the optimizations on the database are preserved in the backed up database. For example, DB2 system statistics and DB2 parameter settings are preserved in the backed up database. When you back up a Directory Server by using the **`idsdbback`** command, the server configuration is backed up along with the DB2 database. Users can also use the **`idsperftune`** command to improve the performance of the Directory Server. For more information about the **`idsperftune`** command, see "The performance tuning tool " on page 40.

Effect of multiple password policy on performance

When you configure multiple password policy to secure a Directory Server, there might be some performance degradation. You must consider the trade-off when you configure multiple password policies.

To evaluate effective password policy for an LDAP user, the directory server considers all the password policies that are associated with a user. A Directory Server evaluates the individual, group, and global password policies to determine effective password policy for a user. When you provide credentials to authenticate to a Directory Server, the effective password policy and group membership are resolved to apply password policies.

If you define group password policies, the performance of bind operations might get degraded.

Enforcing minimum ulimit values

You can set the minimum `ulimit` values that are based on the system resource availability for the optimal usage of resources by a Directory Server.

A Directory Server enforces minimum `ulimit` values such process data size, process virtual memory, and process file size that are considered important for running the server.To enforce minimum `ulimit`, a server checks if the `ulimit` values for the current processes are greater than or equal to the `ulimit` values specified in the configuration file. If the `ulimit` values for the current processes are lesser than the specified values, the server sets the `ulimit` values of the current processes to the specified values.

On Linux® operating systems, resource limits are defined for each user. When a process is started, the process inherits the resource limits set for the user context under which it was started. For example, if the **`idsslapd`** process, is started under the root user context, the **`idsslapd`** process inherits the resource limits of the root user. The inheritance occurs even if the process switches user contexts as the **`idsslapd`** process does. The **`idsslapd`** process switches the user context to the DB2 instance owner when need arises for the **`idsslapd`** process to take on the resource limits of the DB2 instance owner. In such case, the **`idsslapd`** process must be started while the DB2 instance owner is logged in.

The Directory Server uses the `ulimit` values that are set in the configuration file sets them for the current processes. To modify the existing `ulimit` values, you can modify the entries under the `cn=Ulimits, cn=Configuration` DN entry. For example:

```
dn: cn=Ulimits, cn=Configuration
cn: Ulimits
ibm-slapdUlimitDataSegment: 262144
ibm-slapdUlimitDescription: Prescribed minimum ulimit option values
ibm-slapdUlimitFileSize: 2097152
ibm-slapdUlimitNofile: 500
ibm-slapdUlimitStackSize: 10240
ibm-slapdUlimitVirtualMemory: 1048576
objectclass: top
objectclass: ibm-slapdConfigUlimit
objectclass: ibm-slapdConfigEntry
```

Note: An administrator can modify the minimum `ulimit` values by using **Web Administration Tool** or through command line.

Generic tips for using a Directory Server

You can optimize a Directory Server to improve the search and update performance of the server.

The following tips can help improve Directory Server performance.

- Run searches on indexed attributes only. For instructions for defining and verifying indexes for Directory Server, see "DB2 indexes" on page 56.
- Open a connection and reuse it for many operations if possible.
- Minimize the number of searches by retrieving multiple attribute values in a single search.
- Retrieve only the attributes that you need. Do not use ALL by default. For example, when you search for the groups a user belong to, ask for only the Distinguished Names (DNs) values, and not the entire group. Do not request the `member` or `uniquemember` attributes if possible.
- Minimize and batch updates, such as add, `modify`, `modrdn`, or `delete`, when possible.
- Use base-scoped searches whenever possible rather than one-level or subtree searches.
- You must not use 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 order. Leading character:

  ```
  sn=*term
  ```

  Trailing character:

  ```
  sn=term*
  ```

  Note: A filter such as `sn=*term*` is less efficient than the examples listed.
- When you use nested groups, keep the depth of nesting to 50 groups or less. Greater nesting depths can result in increased processing times when you run add or delete operation that involves 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 run search to retrieve all members, edit the returned list, then send the updated list as a modify-replace operation. This modify-replace scenario might not function correctly with large groups.
- For a Proxy Server, do not set the value in the **Connection pool size** field to less than 5.

Directory Server tuning

You must decide the tuning roadmap, tuning LDAP caches, and the effect of directory size on the performance of a Directory Server.

Directory Server tuning tasks

You can tune a Directory Server for general-purpose use or for a specific use for which you must configure the Directory Server.

To use the Directory Server in an LDAP environment, you must configure the server before you tune the server.

You must create an instance and configure the server with a database. If the Directory Server is loaded with entries, it is advisable to back up the Directory Server instance with database. You can prevent loss of data in case of failures during the tuning process. To know more about installing IBM Security Directory Suite and configuring an instance, see the Administering section of the IBM Security Directory Suite documentation.

Directory Server tuning roadmap

You can use the roadmap to identify the tasks that you must run to tune a Directory Server.

Procedure

1. You can use the **idsdbback** command to back up the Directory Server. It is advisable to back up the directory server before you do any major change. See "Directory Server backup methods" on page 7. To know more about the **idsdbback** command, see *Command Reference* section in the IBM Security Directory Suite documentation.
2. Expand the storage capabilities for your server. For more information, see Expanding storage area.
3. Start the Directory Server to complete the instance configuration. DB2 database tables for the Directory Server instance are not created until the first startup of the instance. See "Starting a Directory Server instance" on page 19.
4. Tune the resource limits of the operating system. See Tuning Linux or UNIX operating system resource limits.
5. Tune the DB2 database parameters. See Tuning DB2 and LDAP caches.
6. Create the `LDAP_DESC (AEID, DEID)` index. See Creating the LDAP_DESC(AEID, DEID) index.
7. Tune the Directory Server change log if it is configured. If the change log is configured, then the performance of the directory server might degrade. The performance of a Directory Server is better when the change log is not configured. See "Change log" on page 13.
8. Tune the Directory Server audit log. If the audit log is set, then the performance of the Directory Server might degrade. The performance is better when the audit log is not enabled. See "Audit log" on page 13.
9. Configure the suffixes on the Directory Server. See "Configuring the suffixes on a Directory Server" on page 14.
10. Create LDIF file with entries for configured suffixes. See Creating LDAP entries for configured suffixes.
11. Prepare the Directory Server instance for loading the entries. See "Commands to add user entries" on page 15.
12. Tune the access control lists (ACLs). See "Directory Server ACL tuning" on page 15.
13. Tune the DB2 indexes. See DB2 indexes.
14. Update the DB2 system statistics. See Updating the DB2 statistics.
15. Optionally, back up the Directory Server instance after every major updates.
16. Start the Directory Server instance.

Directory Server backup methods

You can use various commands to back up and restore a directory server instance. Based on your requirements, you must choose the appropriate command to back up and restore a Directory Server.

There are several ways to back up and restore a Directory Server instance, each with its own set of advantages and disadvantages.

idsdbback and idsdbrestore

The **idsdbback** and **idsdbrestore** commands are provided by IBM Security Directory Suite to back up a Directory Server instance. When you back up, both DB2 database and Directory Server configuration files are backed up. The advantages of using these commands are in backing up and restoring of the directory server configuration. To know more about these commands, see the *Command Reference* section of the IBM Security Directory Suite documentation.

The other backup options that are provided with IBM Security Directory Suite do not include the Directory Server configuration information. The directory server configuration information includes directory schema, configuration file, and key stash file. Although, it is possible to back up these files manually; the **idsdbback** and **idsdbrestore** commands make this task easier.

The disadvantage of using the **idsdbback** and **idsdbrestore** commands is less flexibility in how the underlying DB2 restore is done. For example, with the **idsdbrestore** command, the DB2 restore cannot be directed to distribute the database across multiple disks.

db2lif, ldif2db, and bulkload

The **db2ldif** and **ldif2db** commands are provided by the Directory Server to export data from a database to a Lightweight Directory Interchange Format (LDIF) file. You can also use the commands to import the data into the database from the LDIF file. The advantage to using these commands is the portability, standardization, and size factors that they provide.

The output LDIF file can be used to restore a Directory Server on a different operating system. Because the LDIF format is text-based, it is relatively easy to modify an LDIF file.

The disk space that is required for an LDIF file with entries is considerably less than the space required for a DB2 backup with the entries. The disk space requirement for the LDIF output from **db2ldif** is approximately six times less than the backed up database from **db2 backup** or **idsdbback**. The disadvantage with **db2ldif** and **ldif2db** is the time that is required to complete the process, which is more when compared to **db2 backup** and **db2 restore**.

For restoring the database, the **bulkload** command is many times faster than the **ldif2db** command, but is still much slower than **db2 restore**. It is because the **db2 backup** and **db2 restore** commands are essentially a disk copy, and they are the fastest alternative.

Unlike **db2 restore**, to restore a directory server by using **bulkload**, the Directory Server must be empty. It can be accomplished by restoring an empty database, or by unconfiguring, and then reconfiguring the Directory Server. It is not advisable to drop the database and re-create it using DB2 commands. Creating the database manually misses some important configuration steps that are part of **Configuration Tool**.

db2 backup and db2 restore

The **db2 backup** and **db2 restore** commands are provided by DB2. The advantage of using these commands is the performance and flexibility in specifying the location of the database files. The **db2 restore** command can be used to distribute the database across multiple disks or to move the database to another directory. For more information, see Distributing the database across multiple physical disks.

A disadvantage of the **db2 backup** and **db2 restore** commands are their complexity. Another disadvantage is the potential incompatibility in backing up and restoring across operating systems and across DB2 versions. For more information, see the appropriate DB2 backup and restore documentation.

An important consideration when you use the **db2 backup** and **db2 restore** commands is the preservation of DB2 configuration parameters and system statistics optimizations in the backed-up database. The restored database has the same performance optimizations as the backed-up database. When you use the LDAP commands **db2ldif**, **ldif2db**, or **bulkload**, the performance optimization might not be restored.

If you restore over an existing database, any performance optimization on that existing database is lost. Check all DB2 configuration parameters after you restore the Directory Server instance. If you do not know whether **db2 runstats** was run before the database was backed up, tune the DB2 system statistics after the restore (see Update the DB2 system statistics). Use the following DB2 commands to run backup and restore operations:

```
!db2 force applications all
!db2 backup db sdsinst1 to directory_or_device
!db2 restore db sdsinst1 from directory_or_device replace existing
```

Where, *directory_or_device* is the name of a directory or device where the backup is stored.

One of the most common errors that occurs during a restore operation is a file permission error. The error might occur due to the following reasons:

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

  ```
  chown sdsinst1 file_or_dev
  ```

- The backed-up database is distributed across multiple directories, and those directories do not exist on the target system of the restore. Distributing the database across multiple directories is accomplished with a redirected restore. To solve this problem, create the same directories on the target system or run a redirected restore to specify the directories on the new system. If you create the same directories, ensure that the owner of the directories is the DB2 instance owner typically, the sdsinst1 user. For more information about redirected restore, see Distributing the database across multiple physical disks.

Expansion of storage area

On a remote DB2, you can expand the disk space or distribute the data storage when a Directory Server database expands and reaches its disk space limits. If you do not handle the disk space requirement, the server might generate errors and exit.

Note: The steps outlined here are applicable only to a remote DB2 that is configured for use with the Directory Server instance in virtual appliance. These steps are not applicable for the IBM Security Directory Suite virtual appliance stand-alone system, which comes with an embedded instance of DB2.

You can use one of the following ways to expand storage.

- Use storage area network. It is beyond the scope of the document.
- Distribute the database across multiple physical disks.
- Use raw devices. For more information about raw devices, see the topics about Directory Server database and table spaces in Administering section of the IBM Security Directory Suite documentation.

Distributing the database across multiple physical disks

Distributing the database across multiple disks improves performance. If you use multiple disk drives, you can derive better performance due to concurrency.

Before you distribute the database across multiple physical disks, stop the Directory Server. For example:

```
ibmslapd -I instance_name -k
```

Setting up the file system permission and running the DB2 commands to distribute the database across multiple disks can be a complex task.

See the appropriate DB2 documentations to distribute the database across multiple disks.

Table spaces in IBM Security Directory Suite

When you create a directory database, it uses the **db2 create** database command with the database name specified on the directory configuration command.

Before you begin, switch context to the DB2 instance owner. These examples assume that the DB2 instance owner is `sdsinst1`.

- On Linux and UNIX operating systems, run the following command:

  ```
  su - sdsinst1
  ```

- On Windows operating systems, run the following command:

  ```
  !db2cmd
  set DB2INSTANCE=sdsinst1
  ```

You can view the table spaces by using the following DB2 commands. Run the command with the DB2 instance owner, typically the sdsinst1 user.

```
!db2 connect to sdsinst1
!db2 list tablespaces
```

The following examples show table space output for a Directory Server:

```
Tablespaces for Current Database
Tablespace ID       = 0
Name                       = SYSCATSPACE

Type                       = System managed space
Contents               = Any data
State                       = 0x0000
        Detailed explanation:
            Normal
Tablespace ID       = 1
Name                       = TEMPSPACE1
Type                       = System managed space
Contents               = Temporary data
State                       = 0x0000
        Detailed explanation:
            Normal
Tablespace ID    = 2
Name                       = USERSPACE1
Type                       = System managed space
Contents               = Any data
State                       = 0x0000
        Detailed explanation:
            Normal
Tablespace ID    = 3
Name                       = LDAPSPACE1
Type                       = System managed space
Contents             = Any data
State                       = 0x0000
        Detailed explanation:
            Normal
```

The Directory Server stores data in the user table space (USERSPACE1) and in the LDAP table space (LDAPSPACE). By default, there is only one container or directory for each of these table spaces. To view the details about the user table space, enter the following DB2 command:

```
!db2 list tablespace containers for 2
```

The output of the command is:

```
Tablespace Containers for Tablespace 2
Container ID     = 0
Name                       = /sdsinst1/NODE0000/SQL00001/SQLT0002.0
Type                       = Path
```

The container or directory that DB2 uses for table space 2 is `/sdsinst1/NODE0000/SQL00001/SQLT0002.0`. It contains LDAP attribute database tables.

Table space 3 contains the *ldap_entry* table. For more information about table spaces, see Table spaces.

Creating file systems and directories on the target disks

To distribute the DB2 database across multiple disk drives, you must create and format the file systems and directories on the physical disks.

Guidelines for distributing the DB2 database across multiple disk drives include:

- DB2 distributes the database equally across all directories. Therefore, you must make all the file systems, directories, or both, of the same size.
- Directories to use for the DB2 database must be empty. AIX® and Solaris systems create a `lost+found` directory at the root of any file system. Instead of deleting the `lost+found` directory, create a subdirectory at the root of each file system for distributing the database. For example, create a subdirectory, `sdsinst1_containers`, on each file system where the DB2 database is to be stored.
- You must create two more directories under the `sdsinst1_containers` directory. One directory to hold table space 2 and the other for table space 3. For example, these directories can be named `tblspc2` and `tblspc3`. Then, specify these directories on the set table space commands that are provided in Running a redirected restore of the database.
- The DB2 instance user must have write permission on the created directories. For AIX and Solaris systems, the following command sets the appropriate permissions:

```
chown sdsinst1 directory_name
```

Platform-specific guidelines include:

- For the AIX operating system, create an Enhanced Journaled Files System (JFS2) or create the file system with the Large File Enabled option. This option is one of the options on the **Add a Journaled File System smit** menu.
- For AIX and Solaris systems, set the file size limit to unlimited. You can also set file size limit to the size of the largest file system under which the DB2 database files are stored. On AIX systems, the `/etc/security/limits` file controls system limits and **-1** means unlimited. On Solaris systems, the **`ulimit`** command controls system limits.

Backing up the existing database

To back up the existing database, follow these steps:

1. Stop the Directory Server instance.
2. To close all DB2 connections, run the following commands:

   ```
   !db2 force applications all
   !db2 list applications
   ```

   The following message is shown when the commands are run. For example:

   ```
   SQL1611W No data was returned by Database System Monitor.
   ```

3. To initiate the backup process, run the following command:

   ```
   !db2 backup db sdsinst1 to [file system | tape device]
   ```

   If the database is backed up successfully, the following message is shown. For example:

   ```
   Backup successful. The timestamp for this backup image is : 20100420204056
   ```

Note: You must ensure that the backup process was successful before you remove the existing database. If the backup was not successful, the existing database is lost. You can verify whether the backup was successful by restoring to a separate system.

Running a redirected restore of the database

A DB2 redirected restore operation restores the specified database table space to multiple containers or directories. In the following example, assume that the following directories to hold table space 2 are created and are empty. The directories are set with the correct permissions for write access by the DB2 instance owner, typically the sdsinst1 user.

```
/disk1/sdsinst1_containers/tblspc2
/disk2/sdsinst1_containers/tblspc2
/disk3/sdsinst1_containers/tblspc2
/disk4/sdsinst1_containers/tblspc2
/disk5/sdsinst1_containers/tblspc2
```

In the following example, assume the following directories for table space 3 is created:

```
/disk1/sdsinst1_containers/tblspc3
/disk2/sdsinst1_containers/tblspc3
/disk3/sdsinst1_containers/tblspc3
/disk4/sdsinst1_containers/tblspc3
/disk5/sdsinst1_containers/tblspc3
```

Follow these steps for a redirected restore:

1. To start the DB2 restore process, run the command of the following format:

   ```
   !db2 restore db sdsinst1 from [location of backup] replace existing redirect
   ```

   The following message is shown when the command is run. For example:

   ```
   SQL2539W Warning! Restoring to an existing database that is the same as
   the backup image database. The database files will be deleted.

   SQL1277N Restore has detected that one or more tablespace containers are
   inAccessible, or has set their state to 'storage must be defined'.

   DB20000I The RESTORE DATABASE command completed successfully.
   ```

   This command prepares for the restore, but does not actually run the restore operation. These messages indicate that DB2 is prepared to receive the next commands, which define the location of the database files.

2. To define the containers for table space 2 and for table space 3, enter:

   ```
   !db2 "set tablespace containers for 2 using (path \
   '/disk1/sdsinst1_containers/tblspc2', \
   '/disk2/sdsinst1_containers/tblspc2', \
   '/disk3/sdsinst1_containers/tblspc2', \
   '/disk4/sdsinst1_containers/tblspc2', \
   '/disk5/sdsinst1_containers/tblspc2' )"
   ```

   ```
   !db2 "set tablespace containers for 3 using (path \
   '/disk1/sdsinst1_containers/tblspc3', \
   '/disk2/sdsinst1_containers/tblspc3', \
   '/disk3/sdsinst1_containers/tblspc3', \
   '/disk4/sdsinst1_containers/tblspc3', \
   '/disk5/sdsinst1_containers/tblspc3' )"
   ```

   Note: If many containers are defined, these commands can become too long to fit within the limits of a shell command. In such a case, you can put the command in a file and run within the current shell by using the dot notation. For example, assume that the commands are in a file named **`set_containers.sh`**. The following command runs it in the current shell: `. set_containers.sh`

   After completion of the DB2 set table space command, the following message is returned:

   ```
   DB20000I The SET TABLESPACE CONTAINERS command completed successfully.
   ```

   The command might also return the following message: `SQL0298N Bad container path. SQLSTATE=428B2`. This message indicates that one of the containers is not empty or the write permission is not set for the DB2 instance owner.

   Note: A newly created file system on AIX and Solaris contains a directory `lost+found`. You must create a directory at the same level as `lost+found` to hold the table space and then run the set table space command again. If you experience problems, see the DB2 documentation. The following files might also contain the required information to resolve the issue:

```
sdsinst1_home_dir /sqllib/Readme/en_US/Release.Notes
sdsinst1_home_dir /sqllib/db2dump/db2diag.log
```

Note the `db2diag.log` file contains some fairly low-level details that can be difficult to interpret.

3. Restore to new table space containers. This step takes considerable time to complete. The time varies depending on the size of the directory. To restore to the new table space containers, enter the following command:

```
!db2 restore db sdsinst1 continue
```

If problems occur with the redirected restore and you want to stop the restore process, run the following command:

```
!db2 restore db sdsinst1 abort
```

To know more about adding disk space to table spaces, see the Administering section of the IBM Security Directory Suite documentation.

Tuning Linux operating system resource limits

You can improve the performance of a Directory Server instance if you tune the operating system resource limits.

You can tune the resource limits of the Linux operating system that are necessary to support directories with millions of users.

Resource limits (`ulimit`)

The **`ulimit`** command controls the limits that can be set on system resource. You can modify the system resource, such as process data size, process virtual storage, and process file size.

Resource limits are defined for each user. When you start a process, the process inherits or takes the resource limits of the user context under which it was started.

For example, if you start the `idsslapd` process under the root user context, the `idsslapd` process takes the resource limits of the root user. This inheritance occurs even if the process switches user contexts as the `idsslapd` process does.

The `idsslapd` process switches the user context to the DB2 instance owner. For the `idsslapd` process to take the resource limits of the DB2 instance owner, you must start the `idsslapd` process as the DB2 instance owner.

The LDAP_DESC(AEID, DEID) index

You can create the `LDAP_DESC(AEID,DEID)` index to tune and improve the performance Directory Server for subtree operations.

When you create the `LDAP_DESC(AEID,DEID)` index, you must stop the Directory Server instance. You can run create index without stopping the Directory Server instance. If you use the server when you run indexing on the server, it might result in timeouts.

You must create the `(AEID,DEID)` index on the DB2 `LDAP_DESC` table. The following examples show how indexing can improve performance:

- Running a subtree search with an `objectclass=*` filter when 100000 entries exist in the Directory Server.
- Setting an ACL on a small subtree when millions of entries exist in the Directory Server. For example, running the **`ldapmodify`** command to set the ACL.

Propagating ACLs to the child entries of a large subtree is slow regardless of the existence of the index. Without the `LDAP_DESC(AEID,DEID)` index, all subtree searches with the `objectclass=*` filter are slow on a large Directory Server, even if the subtree that you are searching is small.

When you design an LDAP search, you must consider the order in which the subtree criterion is used in evaluating the response. The subtree criterion is used last. Subtree searches specifying a commonly used object class values on a small subtree of a large directory are slow even if the `LDAP_DESC(AEID,DEID)` exists. For best performance, you must not use searches that specify a commonly used object class value.

If the Directory Server has many existing entries, the index creation takes time. To check for the existence of the `LDAP_DESC(AEID,DEID)` index, run the following command:

```
!db2 connect to sdsinst1
!db2 describe indexes for table ldap_desc show detail
```

If the output does not include the `+DEID+AEID` keyword, then the index does not exist.

To create the `LDAP_DESC(AEID,DEID)` index, run the following command:

```
!db2 connect to sdsinst1
!db2 "create index LDAP_DESC_DEID on LDAP_DESC(AEID,DEID)"
```

You can remove the `LDAP_DESC(AEID,DEID)` index by the running the following command:

```
!db2 connect to sdsinst1
!db2 drop index IDSLDAP.LDAP_DESC_DEID
```

Change log

You can record all the directory updates in a change log DB2 database, which is different from the DB2 database associated an instance. When you configure the change log, it significantly slows down the update performance of a Directory Server.

To tune the Directory Server change log, you must start the directory server instance.

To determine the change log configuration status, you must search for the `CN=CHANGELOG` suffix. For example:

```
ldapsearch -h ldap_host -D cn=root -w admin_passwd -s base -b "" \
"objectclass=*" | grep "CN=CHANGELOG"
```

where, *admin_passwd* is the password for the primary administrator.

Audit log

You must determine whether you require auditing of operations that are run against a Directory Server. If you do not require to audit operations, you can disable the audit feature.

To tune audit log, you must start the Directory Server instance.

The Directory Server audit log feature significantly slows down the Directory Server performance, depending upon which audit log features are turned on. It is advisable to turn off all audit log features.

To check the status of the audit log feature, run the following **`ldapsearch`** command:

```
ldapsearch –D cn=root –w admin_passwd –s base –b "cn=Audit,cn=Log Management,cn=Configuration" \
"objectclass=*" ibm-audit
```

Where, *cn=root* is the Directory Server root administrator user, and *admin_passwd* is the password for the administrator.

The following output is returned if the audit log is not set:

```
cn=Audit, cn=Log Management, cn=Configuration
ibm-audit=false
```

where, `ibm-audit=false` indicates that the audit log feature is turned off.

If this value is true, run the following command to turn it off:

```
ldapmodify -D cn=root -w admin_passwd
dn: cn=Audit, cn=Log Management, cn=Configuration
```

```
changetype: modify
replace: ibm-audit
ibm-audit: false
```

Configuring the suffixes on a Directory Server

You must create and configure at least one suffix before you add an LDAP entry to a Directory Server instance.

Before you begin

You must ensure that the Directory Server instance is running in normal mode.

About this task

For optimum Directory Server authentication performance, you must define another suffix to hold all Directory Server user entries. You can define more than one suffix, and distribute the user entries among multiple suffixes. In such a hierarchy, the server might need more time to search and determine under which suffix a user entry exists during authentication. Instead of storing users in multiple suffixes, you must store users entries in multiple LDAP containers under the same suffix for better performance.

Procedure

1. Log in as the Directory Server instance owner.
2. To determine the existing suffixes, run the following command:

   ```
   ldapsearch -p port -L -D cn=root -w password -s base \
   -b "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, \
   cn=Configuration" "objectclass=*" ibm-slapdSuffix
   ```

   In addition to any user created suffixes, the output from the command returns the following suffixes:

   ```
   ibm-slapdSuffix: cn=localhost
   ibm-slapdSuffix: cn=ibmpolicies
   ibm-slapdSuffix: cn=Deleted Objects
   ```

3. Run the following command to create a suffix:

   ```
   ldapmodify -p port -D cn=root -w password
   dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
   changetype: modify
   add: ibm-slapdSuffix
   ibm-slapdSuffix: suffix_name
   ```

   where, *password* is the password for the primary administrator, and *suffix_name* is the LDAP suffix to configure.
4. Run one the following commands to effect the changes that are made to the Directory Server configuration file:

   - Restart the Directory Server instance and the administration server.

     ```
     ibmslapd -I instance -k
     ibmdiradm -I instance -k
     ibmslapd -I instance
     ibmdiradm -I instance
     ```

   - Run the **ldapexop** command to reread the attribute.

     ```
     ldapexop -p port -D cn=root -w password -op readconfig -scope single \
     "cn=Directory,cn=RDBM Backends,cn=IBM Directory, cn=Schemas,cn=Configuration" \
     ibm-slapdSuffix
     ```

LDAP entries for suffixes

You must create a suffix entry to represent as the parent entry in an LDAP directory information tree (DIT) hierarchy.

You can create LDAP entries for the suffixes by one of the following ways:

- Use the **ldapadd** command to create the LDAP entries for suffixes.
- Restore a directory database that contains the LDAP entries for suffixes.
- Load the Directory Server with an LDIF file that contains the LDAP entries for suffixes.

To create suffix entry for an `o=sample` suffix, run the **ldapadd** command. For example:

```
ldapadd -h host -p port -D cn=root -w admin_passwd
dn: o=sample
objectClass: organization
objectclass: top
o: sample
```

where, the variables indicate:

- *cn=root* is the primary administrator of an instance
- *admin_passwd* is the password credentials of the administrator
- *host* is the host name where the Directory Server instance is running
- *port* is the port number of the Directory Server instance

In the example, access control lists (ACLs) are not assigned to the suffix object. You can assign the appropriate ACLs based on your LDAP environment requirement.

Commands to add user entries

You must add user entries in a Directory Server if you plan to use the entries for retrieving information or for configuring LDAP authentication.

To add user entries to a Directory Server instance, you must prepare an LDIF file with user entries. You can use the **ldapadd**, **ldif2db**, or **bulkload** command to add entries to a directory server instance.

Before you load user entries to a Directory Server instance, you must create a directory tree on the Directory Server. The directory tree includes LDAP container entries for holding user and group information.

To add few entries, use the **ldapadd** command. Before you add entries by using the **ldapadd** command, you must ensure that the Directory Server instance is running. To add many entries, use the **ldif2db** or **bulkload** command. Before you add entries by using the **ldif2db** or **bulkload** command, you must ensure that the Directory Server instance is stopped.

Directory Server ACL tuning

You can assign specific access rights to the LDAP users by assigning the required access control list (ACL) to the entries in a Directory Server.

You must start the Directory Server instance before you tune the access control lists (ACLs).

If you set ACLs on the suffixes, you might observer performance delays and requirement for large transaction log spaces. The conditions are true, if:

- Millions of LDAP entries exist under the suffix.
- There is no previous ACL on the suffix.

To prevent performance delays, you must set ACLs on the suffixes when there are few or no user entries or LDAP entries under the suffix. You must set the ACLs on the suffixes before you add entries under the suffix. For more about assigning ACL on all suffixes, see the Administering section of the IBM Security Directory Suite documentation.

If you configure ACLs when millions of user entries exist under the suffix and there is no previous ACL on the suffix, performance delays might occur.

DB2 statistics

When you run update operations against a Directory Server, the DB2 statistics of the Directory Server database is affected. You must update the DB2 statistics of the Directory Server database after large updates to the directory server.

When you update the DB2 statistics of a database, you must stop the Directory Server to prevent any failures. You can update the DB2 statistics, while the Directory Server is running. However, application that is accessing the Directory Server might timeout.

The DB2 optimizer uses DB2 system statistics and the volatile table definitions to optimize DB2 queries. When LDAP entries are added to a Directory Server, the DB2 statistics on the affected tables can become out of date. The out-of-date system statistics can result in wrong choices by the DB2 optimizer, and can result in poor performance.

Not updating DB2 system statistics is one of the most common performance tuning mistakes that are made with DB2 databases. The first step in updating the DB2 system statistics is to run the DB2 **runstats** command on every table or run the DB2 **reorgchk** command. Later versions of DB2 provide method to tune automatic checks for out-of-date system statistics and update the out-of-date system statistic by using the **runstats** command.

Do not use the automatic **runstats** option with a Directory Server. Updating the DB2 system statistics for a Directory Server also involves overriding of the default DB2 system statistics by the **runstats** command.

After you update the DB2 system statistics, you must define all tables as volatile. You can also manually update the DB2 statistics by running the appropriate DB2 command.

Update system statistics with DB2 runstats or reorgchk

You can update the DB2 system statistics by running a DB2 **runstats** command on all tables in the database or by running the DB2 **reorgchk** command. You can obtain the list of tables in a database by running the following command:

```
!db2 connect to sdsinst1
!db2 list tables for all
```

It is only necessary to perform the DB2 runstats command on the tables having the DB2 instance schema. For example, in this document, the DB2 instance schema is sdsinst1. The DB2 runstats command is run as follows:

For DB2, Version 10.5.0.3:

```
!db2 runstats on table tablename with distribution and detailed indexes all
shrlevel change
```

Where, *tablename* is the name of the table on which to run the **runstats** command.

If you provide the `shrlevel change` and `allow write access` options, the database becomes accessible to the Directory Server instance. When the **runstats** commands are running, it can result in poor performance and timeouts in directory server operations.

Running the **runstats** command on all tables can be tedious if not automated with a script. The alternative is the DB2 **reorgchk** command. To update DB2 statistics, run the DB2 **reorgchk** command. For example:

```
!db2 connect to sdsinst1
!db2 reorgchk update statistics on table all
```

The **reogchk** command does some additional checking and reports on the organization of the data in the database. It is useful when an application requires to read the database sequentially. When you access a Directory Server for common use, the database is accessed randomly.

The main advantage of the DB2 **runstats** command is the ability to select which tables to tune. When you run the DB2 **runstats** command, you must selectively limit it to the tables that are less than 100,000 rows. In most cases, tables with 100,000 rows no longer require tuning. By limiting the **runstats** command for relatively small tables, the frequency to run **runstats** decreases. For example, if a million users are added to a Directory Server that already contain a million users, it is likely that the affected tables already have a million rows and making it unnecessary to do **runstats** on that table.

If the second million users introduce a new LDAP attribute that the first million did not have, it is necessary to run **runstats** on the tables for the new attributes. But it is not required for those tables that already have a million rows. In both these cases, the time to update the system statistics by using the **runstats** command is less than it would take to run **runstats** on all tables of the directory database.

Updating system statistics with idsrunstats

You can run the **idrunstats** tool that is provided with IBM Security Directory Suite, while the Directory Server is running. The **idsrunstats** command collects the distribution statistics on all the columns and indexes of LDAP_DESC and LDAP_ENTRY tables.

To know more about **idsrunstats**, see Optimization and organization (idsrunstats, reorgchk and reorg).

Improving disk utilization with DB2 row compression

If you installed a fully licensed version of DB2 Enterprise Server Edition, in addition to a DB2 Storage Optimization Feature license, you can use row-level compression.

You can improve the disk utilization and the overall performance with the DB2 row compression capability. You can select the rows that you want to compress by using the **idsdbmaint** tool that is provided with IBM Security Directory Suite. For more information, see DB2 row compression.

Defining DB2 tables as volatile

DB2, Version 10.5.0.3 supports the option to define tables as volatile. Defining the tables as volatile, enables the DB2 optimizer to use the indexes that are defined on the table. If you tune the DB2 optimizer, the DB2 optimizer makes the correct optimization choice when the statistics are out of date. For example, when a table suddenly grows in size before the system statistics is updated. Run the following command to define a table as volatile:

```
!db2 connect to sdsinst1
!db2 alter table tablename volatile
```

Where, *tablename* is the name of the table to be defined as volatile.

Override system statistics

In some cases, the DB2 optimizer makes a wrong choice in optimizing a query even if the system statistics are up to date. In such cases, it is necessary to override the DB2 system statistics to influence the DB2 optimizer to make the correct choice. A Directory Server can override some of the system statistic. For example, when the Directory Server is started as part of the **idsrunstats** command.

You can set the *LDAP_MAXCARD* environment variable to control system statistic overrides by a Directory Server. For more information about setting *LDAP_MAXCARD* and its behavior, see the http://www-01.ibm.com/support/docview.wss?uid=swg21316267 website.

LDAP_DESC

When you set the variable, it sets the cardinality column of the LDAP_DESC table in the system statistic tables to a value of 9E18, which is the scientific notation for a large number. A Directory Server can also force this override.

To tune manually, run the following command:

```
!db2 "update sysstat.tables set card = 9E18 where tabname = 'LDAP_DESC'"
```

When you override, it chooses the LDAP_DESC table last during an LDAP search. The LDAP_DESC table is used on subtree searches when the LDAP_DESC(AEID,DEID) index is defined.

If the `LDAP_DESC(AEID,DEID)` index is not defined, this override has no effect. This override allows small subtree searches to be fast if they are specified with an `objectclass=*` filter. The main purpose of this override is to prevent use of the `LDAP_DESC(AEID,DEID)` index with large subtrees. It ensures that the attributes on the filter are used first with an LDAP search, if attributes are specified.

LDAP_ENTRY

For one level searches, entry IDs are resolved using the `PEID` column of the `LDAP_ENTRY` table.

When you set the variable, it sets the cardinality column of the `LDAP_ENTRY` table in the system statistic tables to a value of 9E18, which is scientific notation for a large number. A Directory Server cannot force this override.

To tune manually, run the following command:

```
!db2 "update sysstat.tables set card = 9E18 where tabname = 'LDAP_ENTRY'"
```

In an LDAP search, the `LDAP_ENTRY` table `PEID` index is queried last because of the override. The `PEID` index is used when you run a one level search. It ensures that the attributes on the filter are used first with an LDAP search, if attributes are specified.

CN

When you set the variable, it sets the cardinality column of the CN table in the system statistic tables to a value of 9E10. A Directory Server cannot force this override.

To tune manually, run the following command:

```
!db2 "update sysstat.tables set card = 9E10 where tabname = 'CN'"
```

The override chooses the CN table before the subtree criteria (LDAP_DESC table) and the one level criteria (LDAP_ENTRY table).

This override also uses the OBJECTCLASS filter before the CN attribute with an LDAP search.

REPLCHANGE

When you set the variable, it sets the cardinality column of the `REPLCHANGE` table in the system statistic tables is to a value of 9E18, which is scientific notation for a large number. It also sets the `colcard` and `high2key` columns of the `REPLCHANGE` table with a `colname` of ID in the system statistic columns table to 9E18 and 2147483646. A Directory Server can also force this override.

To tune manually, run the following commands:

```
!db2 "update sysstat.tables set card = 9E18 where tabname = 'REPLCHANGE'"
!db2 "update sysstat.columns set colcard=9E18, high2key='2147483646'
where colname = 'ID' and tabname = 'REPLCHANGE'"
```

DB2 can use the index that is defined on the REPLCHANGE table when the override is set. The DB2 optimizer does not use indexes on an empty table. For DB2, Version 10.5.0.3, if you define the table as volatile then the override is redundant.

db2look command

The **`db2look`** command is useful for reporting all the system statistic settings of the database. Use the mimic option, **`-m`**, to generate a report that contains the DB2 command which produces the current system statistic settings. For example:

```
!db2look -m -d sdsinst1 -u sdsinst1 -o output_file
```

Where, *output_file* is the file location for storing the results.

Before you run the **`db2look`** command, switch the user context to the database instance owner.

Backing up a Directory Server instance

To restore a Directory Server from an unexpected failure, you must back up the Directory Server instance.

You can preserve the existing tuning configuration of a directory server instance if you back up the instance. For more information, see IBM Security Directory Suite backup methods.

Starting a Directory Server instance

To use a Directory Server instance, you must start the instance.

About this task

After you configure a Directory Server instance, you must start the instance to complete the database configuration. Database tables and indexes are not defined until you start the server for the first time.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. Determine whether a Directory Server instance is running.

   Run this command: **`ps -w | grep slapd`**

   If the output shows a `slapd` or `ibmslapd` process, the Directory Server is running.
3. Start the Directory Server instance.

   Run this command: **`ibmslapd -I`***instance*

   The *instance* variable is the name of the Directory Server instance to start.

Commands for tuning a Directory Server instance

You can use various commands to tune a Directory Server instance. Each command tunes a specific aspect of a Directory Server.

IBM Security Directory Suite provides the following commands to tune a Directory Server instance.

`idrunstats`

You can use this command to collect DB2 system statistics.

`idsperftune`

You can use this command to run a number of tuning activities that are based on user provided settings.

- Providing recommendations
- Updating parameters

This command can be used to run more advanced tuning of DB2 parameters.

`idsdbmaint`

You can use this command for the following operations:

- Index reorganization
- Row compression, see Improving disk utilization with DB2 row compression

The following developerWorks® documents can provide more information about analyzing DB2 and LDAP performance. For more information, see the following websites:

- Troubleshooting Directory Server performance, Part 1
- Troubleshooting Directory Server performance, Part 2

Directory Server performance statistics

You can use the operational statistics of a Directory Server to determine the server performance.

Monitor search

You can run the monitor search against a Directory Server to check the server performance statistics. You can use the directory server statistics to calculate server throughput.

To search for server status, run the **ldapsearch** command with the `cn=monitor` base:

```
ldapsearch -h ldap_host -p port -s base -b cn=monitor "objectclass=*"
```

The command returns several statistics. For a Directory Server, you can check the `opsinitiated` statistic for monitoring performance. It indicates the number of LDAP operations that were initiated after the server started.

The **ldapsearch** command itself accounts for three operations. The following formula can be used to determine the throughput for interval:

```
throughput for interval =
(opsinitiated(at stop time) - opsinitiated(at start time) - 3) /
(stop_time - start_time)
```

Concurrent updates

You can run concurrent updates by using multiple clients against a Directory Server to obtain better update performance.

The amount of performance improvement is limited to the speed of the system disk on the master, and in a replicated environment on the replica servers. In a replication environment, you might require to tune the number of replica threads (or consumer connections) in the master server configuration. Tuning of the number of replica threads is required if you observe an increased update rate on the master server.

Multi-threaded replication is supported, which improves the performance by supporting concurrent updates on replica servers.

LDAP caches

You can populate and tune LDAP caches to reduce disk I/O operations and to improve Directory Server performance.

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

An LDAP search command that accesses an LDAP cache can be faster than the one that requires a connection to DB2. The data retrieval from an LDAP cache is eve faster that retrieving the information that 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.

You must determine how to use LDAP caches, such as filter cache, ACL cache, and entry cache, and the optimum cache settings for your system. Cache sizes for the filter cache, ACL cache, and entry cache are measured in numbers of entries. Keep in mind that every workload is different, and some experimentation is required to find the best settings for your workload.

LDAP attribute cache (deprecated)

You can use LDAP attribute cache to resolve search filters faster if the attributes requested in the search filter are cached. (deprecated)

Note: Attribute cache is deprecated. You must avoid configuring attribute cache.

LDAP filter cache

The filter cache stores cached entry IDs that match a search filter that was previously resolved in DB2.

When the client issues a query for data, the query goes to the filter cache in the following conditions:

- The query is not a base-scoped search that can be resolved in memory.
- The filter cannot be resolved in memory by the attribute cache manager.

When a query arrives at the filter cache, one of the following actions occur:

- The IDs that match the filter settings that are used in the query are in the filter cache. If such cases, 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 such cases, the query must access DB2 in search of the data.

Filter cache size

You must determine the optimum filter cache size to resolve maximum filters in the memory.

To determine the size for the filter cache, run the workload with the cache set to different values and measure the differences in operations per second. For example, the figure shows varying number of operations per second based on different filter cache sizes.

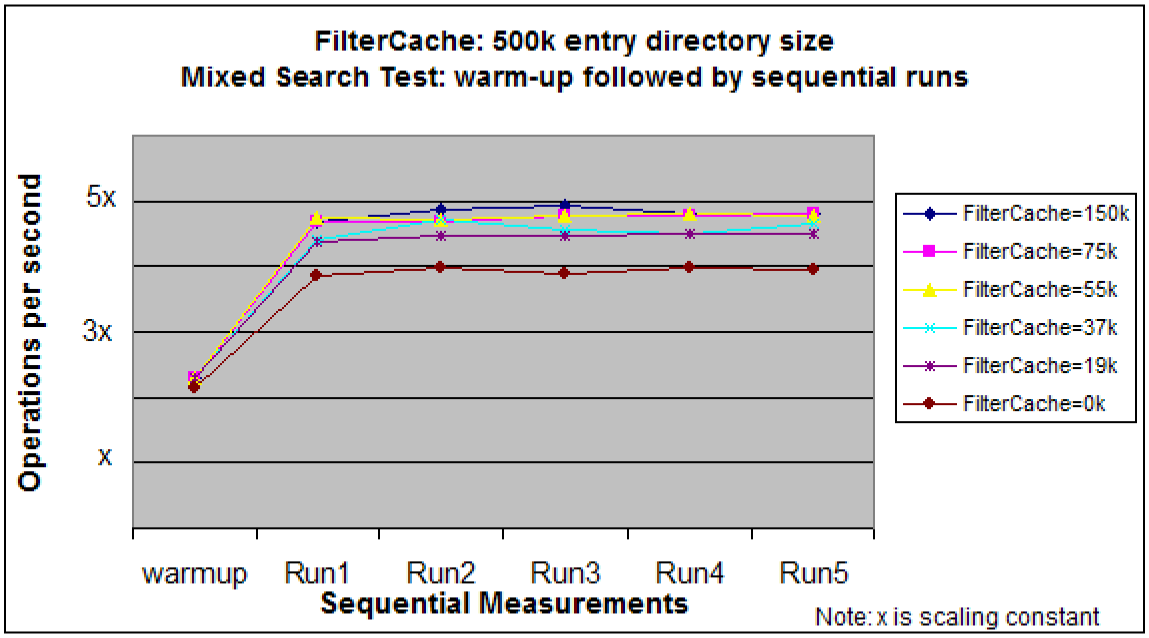

Figure 2. Varying the size of the filter cache

For the workload, a filter cache large enough to hold 55,000 entries results in the best performance. There is no benefit in configuring the filter cache larger than the value that result in best performance. For more information about setting the filter cache size, see “LDAP cache configuration variables” on page 25.

Filter cache size with updates

You must determine the filter cache size that is required to cache filters if your Directory Server environment handles frequent updates.

In a Directory Server environment, if a small fraction of the operations in the workload is updates then setting filter cache does not benefit. There is no performance benefit in allocating memory to the filter cache as shown in Figure 3 on page 22.

If the workload is similar in your environment, you must batch your updates to retain the performance advantage of a filter cache. Batch updates provide 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. For more information about setting the filter cache size, see "LDAP cache configuration variables" on page 25.

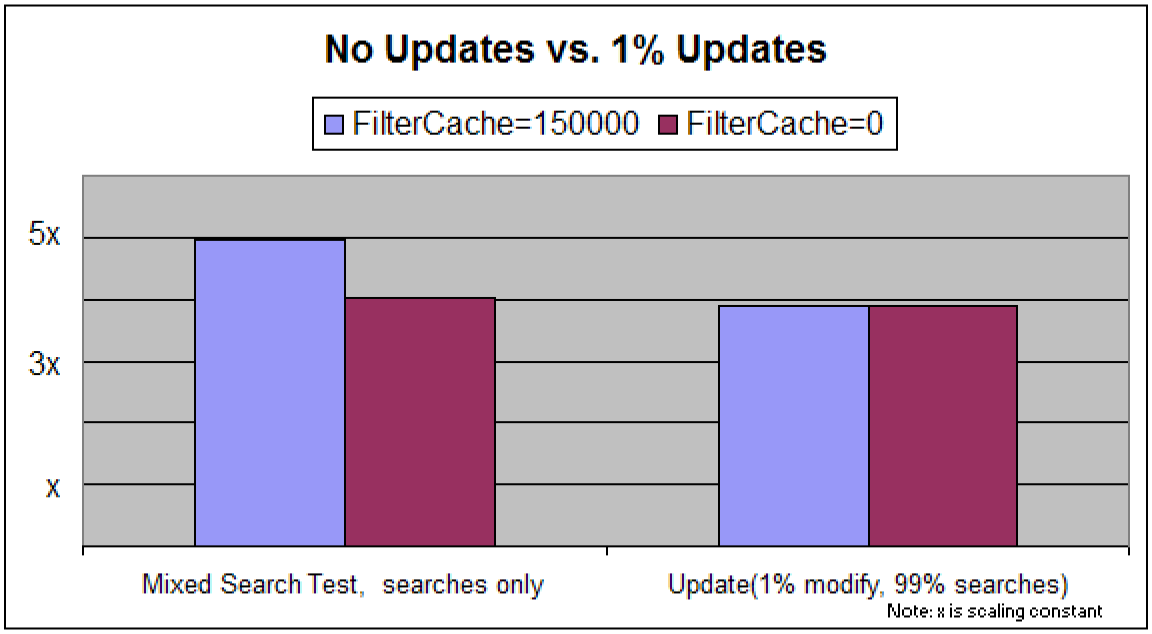

Figure 3. Effect of updates on the performance of the filter cache

Filter cache bypass limits

You can set the limit to prevent uncommon searches from overwriting the cache entries.

You can set the filter cache bypass limit to specify the maximum number of entries to add to the filter cache. For example, if the bypass limit is set to 1000, search filters that match more than 1000 entries are not added to the filter cache. For more information about setting the filter cache bypass limit, see "LDAP cache configuration variables" on page 25.

Entry cache

You can retrieve the entry IDs faster for the search filters that are resolved and the entry IDs are in the entry cache.

When you run queries to access entries, the entry IDs are cached in 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 for the entry IDs, the query is sent to DB2 to retrieve the matching entries.

Entry cache size

You must determine the optimum entry cache size to store and retrieve maximum entry IDs from the cache to improve the search performance.

To determine the required entry cache size, run the workload with the entry cache set to different sizes and measure the differences in operations per second. For example, Figure 4 on page 23 shows varying operations per second based on different entry cache sizes.

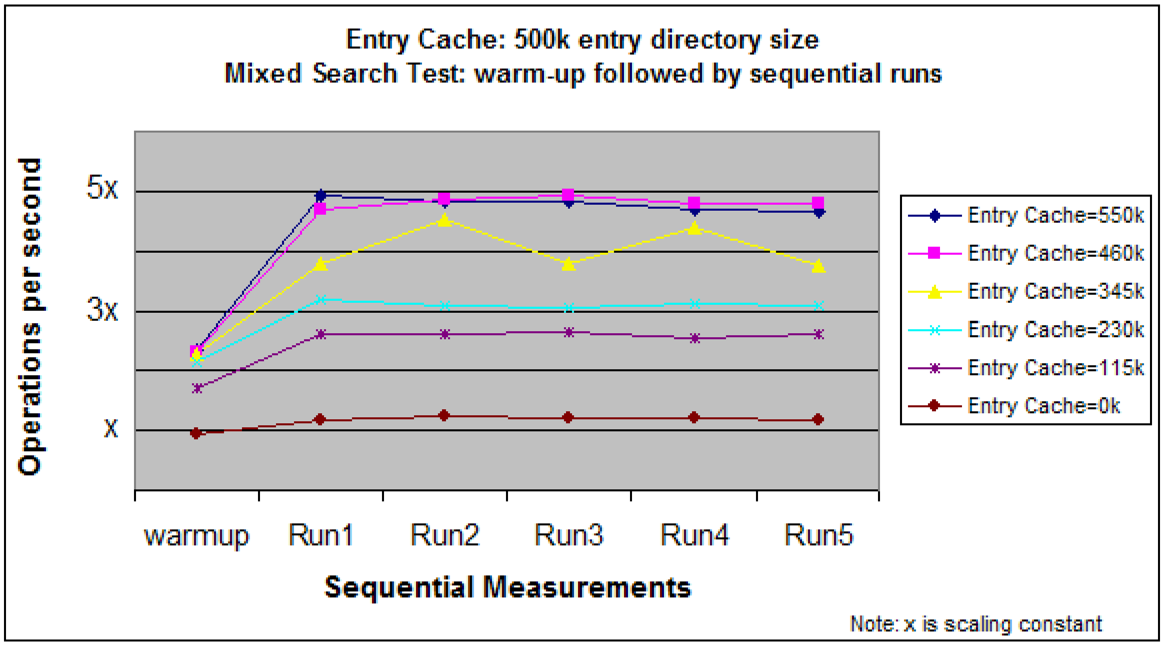

Figure 4. Varying the size of the entry cache

Based on the results in Figure 4 on page 23, an entry cache large enough to hold 460,000 entries results in the optimum performance. There is no benefit in configuring the entry cache larger than the value that result in best performance. Setting the entry cache at 460,000 results in four times as many operations per second than when the entry cache is set to zero. To find the best cache size for your workload, you must run your workload with different cache sizes. For more information about setting the cache size, see "LDAP cache configuration variables" on page 25.

Note: A test with entry cache size set to 345000 results in unpredictable performance. The nature of the test case, workload, and cache size set might affect performance.

Group members cache

You can use the group members cache to store `member` and `uniquemember` attributes and their values with entries.

The group members cache is an extension of the entry cache. The group entries are part of the group members cache if the entry structures have `members` and `uniquemembers`. Otherwise, they are part of the regular entry cache. Group member caching can be controlled with the following configuration attributes:

- `ibm-slapdGroupMembersCacheSize` defines the number of groups whose members are cached. The default value for the attribute is 25.
- `ibm-slapdGroupMembersCacheBypassLimit` defines the maximum number of members a group can contain to cache it in the group members cache. The default value of the attribute is 25000.

ACL cache

You can use the access control list (ACL) cache to store information about the access permissions for the recently queried entries.

The ACL cache stores information such as `entryOwner` and whether the permissions of the entry are explicit or inherited. Caching the ACL information in memory can quickly resolve the ACLs of a user who submitted a query. It also resolves whether the user is authorized to see all, some, or none of the results of the query.

Measuring cache entry size

Filter cache and entry cache size are measured in numbers of entries. To determine the number of entries to cache in the LDAP caches, you must find how large the entries in the caches are.

About this task

You can calculate the average size of an entry in a sample entry cache, but the average filter cache entry size can be calculated similarly.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. Take the following actions on the LDAP server:
   a) Set the filter cache size to 0.
   b) Set the entry cache size to a small value. For example, 200.
   c) Start the `ibmslapd` process of an instance.
3. Take the following actions from the client:
   a) Run your application.
   b) Run the following command to find the entry cache population, *population1*:

   ```
   ldapsearch -h server -p port -s base -b cn=monitor objectclass=*
   | grep    entry_cache_current
   ```
4. Take the following actions on the LDAP Server:
   a) Run the following command to find the memory that is used by the `ibmslapd` process, *ibmslapd1*:

   `ps -w | grep ibmslapd`

   b) Stop the `ibmslapd` process.
   c) Increase the size of the entry cache but keep it smaller than your working set.
   d) Start the `ibmslapd` process.
5. Run your application again and find the entry cache population, *population2*.

   ```
   ldapsearch -h server -p port -s base -b cn=monitor objectclass=*
   | grep    entry_cache_current
   ```
6. Find the memory that is used by the `ibmslapd` process, *ibmslapd2*.
7. Calculate the size of an entry cache entry by using the following formula:
   (*ibmslapd size2* - *ibmslapd size1*)/(*entry cache population2* - *entry cache population1*)

Results

For example, when you use the formula for a 500000 entry database, the following entry size is calculated:

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

LDAP cache configuration variables

You can use the LDAP cache configuration variables to assign values to the LDAP cache sizes, bypass limits, and other variables that affect Directory Server performance.

Attribute cache configuration

You can configure attribute cache for a directory database, change log database, or both to improve Directory Server performance during search operations.

The attribute cache size is measured by the amount of memory the attribute cache requires. You can configure the maximum amount of memory available to use for attribute caching. There is no benefit from configuring attribute caching for the change log database; unless you run frequent searches against change log.

Note: Attribute cache is deprecated. You must avoid configuring attribute cache.

Configuring attribute cache by using Web Administration Tool

You can configure attribute cache by using **Web Administration Tool** to improve Directory Server performance when search operations are run against the server.

About this task

You can configure attribute cache from **Web Administration Tool** or from the command prompt.

Procedure

1. Log in as the Directory Server administrator on a directory server from **Web Administration Tool**.
2. In the navigation pane, expand **Server administration** > **Manage cache properties**, and click **Attribute cache**.
3. In the Attribute cache page, take the following actions:
   a) To configure memory for the directory cache, assign an appropriate value in KB in the **Directory cached attribute size** field. The default is 16384 KB (16 MB).
   b) To configure memory for the change log cache, assign an appropriate value in KB in the **Changelog cached attribute size** field. The default is 16384 KB (16 MB).

   Note: If change log is not configured, you cannot configure memory size for the change log cache.
4. To set directory automatic attribute caching, take the following actions:
   a) Select the **Enable directory automatic attribute cache** check box.
   b) Enter the start time for directory automatic attribute caching in the **Start Time** field.
   c) From the Interval list, select the interval at which to run the directory automatic attribute caching.
5. To set change log automatic attribute caching, take the following actions:

   Note: You must not configure automatic attribute caching for change log unless frequent searches against the change log is required.

   a) Select the **Enable change log automatic attribute cache** check box.
   b) Enter the start time for change log automatic attribute caching in the **Start Time** field.
   c) From the Interval list, select the interval at which to run the change log automatic attribute caching.
6. To add specific attributes to cache, take the following actions:
   a) Select the attribute to cache from the **Available attributes** list.
      In the list, the attributes that are designated as cached attributes are listed. For example, `sn`. An attribute remains in the list of available attributes until it is placed in the **Database** or **Change log** containers. You can assign the same attribute in both containers.
   b) To place to the attribute in the Database container, click **Add to Database**.
   c) To place to the attribute in the Change log container, click **Add to Change log**.

d) Add the required attributes to cache by repeating the process.

   An attribute is removed from the **Available attributes** list when it is added to both the **Cached attributes under Database** and **Cached attributes under Change log** lists. If change log is not set, then **Add to Change log** is disabled. You cannot add the attribute to **Cached attributes under Change log** list.

e) To save the changes, click **Apply**.

   You can click **OK** to save the changes and to exit the page. You can click **Cancel** to exist the page without saving any changes.

Configuring attribute cache from the command line

You can configure the attribute cache from the command line to improve Directory Server performance when search operations are run against the server.

Procedure

1. Log in as the Directory Server instance owner on the system where the Directory Server instance is running.
2. To configure specific attributes for the directory database, run the following command:

   ```
   ldapmodify -p port -D adminDN -w adminPW -i filename
   ```

   where, *filename* contains the following entries:

   ```
   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
   ```

3. To configure automatic attribute caching for the directory database, run the following command:

   ```
   ldapmodify -D adminDN -p port  -w adminPW -i filename
   ```

   where, *filename* contains the following entries:

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

4. To configure specific attributes for the change log, run the following command:

   ```
   ldapmodify -D adminDN -p port  -w adminPW -i filename
   ```

   where, *filename* contains the following entries:

   ```
   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
   ```

5. Restart the Directory Server and Administration Server to effect the changes made.

   ```
   ibmslapd -I instance -k
   ibmdiradm -I instance -k
   ```

```
ibmslapd -I instance
ibmdiradm -I instance
```

ACL cache, entry cache, and filter cache configuration

You can set ACL cache, entry cache, and filter cache configuration variables with the values based on your requirement. Setting optimal values to the caches improves update and search performance of a Directory Server.

Configuring ACL cache, entry cache, and filter cache by using Web Administration Tool

You can configure ACL cache, entry cache, and filter cache by using **Web Administration Tool** to tune your Directory Server performance for search operations.

Procedure

1. Log in with the Directory Server administrator credentials by using **Web Administration Tool**.
2. In the navigation pane, expand **Server administration** > **Manage cache properties**.
3. To configure the ACL cache, click **ACL cache**.
   a) Click **Cache ACL information** to cache the ACL details of entries.
   b) In the Maximum number of elements in ACL cache field, type the number of elements to cache. The default is 25,000.
4. To configure the number of entries to cache in the entry cache, click **Entry cache**.
   a) In the Maximum number of elements in entry cache field, type the number of entries to cache. The default is 25,000.
5. To configure the number of elements to cache in the filter cache, click **Filter cache**.
   a) In the Maximum number of elements in search filter cache field, type the number of elements to cache. The default is 25,000.
   b) To specify the maximum number of entries to add to the filter cache that are matched by a search filter, click **Elements** and type a number. The default is 100.

      You can click **Unlimited** to cache the maximum number of entries that is supported by filter cache.
6. To save the changes, click **Apply**.

   You can click **OK** to save the changes and to exit the page. You can click **Cancel** to exist the page without saving any changes.

Related concepts

"Entry cache" on page 22
You can retrieve the entry IDs faster for the search filters that are resolved and the entry IDs are in the entry cache.

"LDAP filter cache" on page 21
The filter cache stores cached entry IDs that match a search filter that was previously resolved in DB2.

"Filter cache bypass limits" on page 22
You can set the limit to prevent uncommon searches from overwriting the cache entries.

Configuring ACL cache, entry cache, and filter cache from the command line

You can configure the ACL cache, entry cache, and filter cache from the command line. You must tune these caches to improve a Directory Server performance for search operations.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. To configure LDAP cache configuration variables, run the following command:

```
ldapmodify -p port -D adminDN -w adminPW -i filename
```

where, *filename* contains the following entries:

```
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
```

3. Restart the Directory Server and Administration Server to effect the changes made.

```
ibmslapd -I instance -k
ibmdiradm -I instance -k
ibmslapd -I instance
ibmdiradm -I instance
```

Additional attribute settings

You can improve the performance of a server by setting attributes that limit client activity and minimize the effect on server throughput and resource usage.

You can set the following attributes to limit various operations. The default values for the attributes are specified.

- `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`

The Directory Server response time for searches with the alias dereferencing can vary based on the options set. The time for searches to complete is greater if the dereferencing option is set to `always` or `searching` than when the dereferencing option set to `never`. You can use the `ibm-slapdDerefAliases` attribute under the `cn=Configuration` entry to override the dereference option that is specified in the client search requests. The supported values are:

- `never`
- `find`
- `search`
- `always`

If you set the value to `never`, the server does not attempt to dereference possible aliases, and the response time for searches improves.

SLAPD_OCHSELECT_USECS environment variable

Set the *SLAPD_OCHSELECT_USECS* environment variable to call `OCH select()` with a timeout value. The timeout value is of microsecond granularity.

If the *SLAPD_OCHSELECT_USECS* variable is not present or set to `0`, the `OCH select()` call continues with an indefinite timeout value. If *SLAPD_OCHSELECT_USECS* is set to a positive integer value, the `OCH select()` call continues with the set timeout value.

You can set the *SLAPD_OCHSELECT_USECS* environment variable or add the `ibm-slapdSetenv` attribute with the variable as value. You must add the attribute under the `cn=Front End, cn=Configuration` entry in the `ibmslapd.conf` file. For example, to set the value of *SLAPD_OCHSELECT_USECS* to `1000` microseconds, run the **`ldapmodify`** command. For example:

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

You must restart the Directory Server to effect the changes made. When you restart the server, *SLAPD_OCHSELECT_USECS* is set with a value of 1000 microseconds (or 1 millisecond).

Directory size

You must consider the directory size and potential growth in directory size when you configure a Directory Server.

It is important when you run your workload that you consider several measurements. For example, the number of operations per second as shown in Figure 5 on page 29. The number of operations the server resolves reduces as the database size grows.

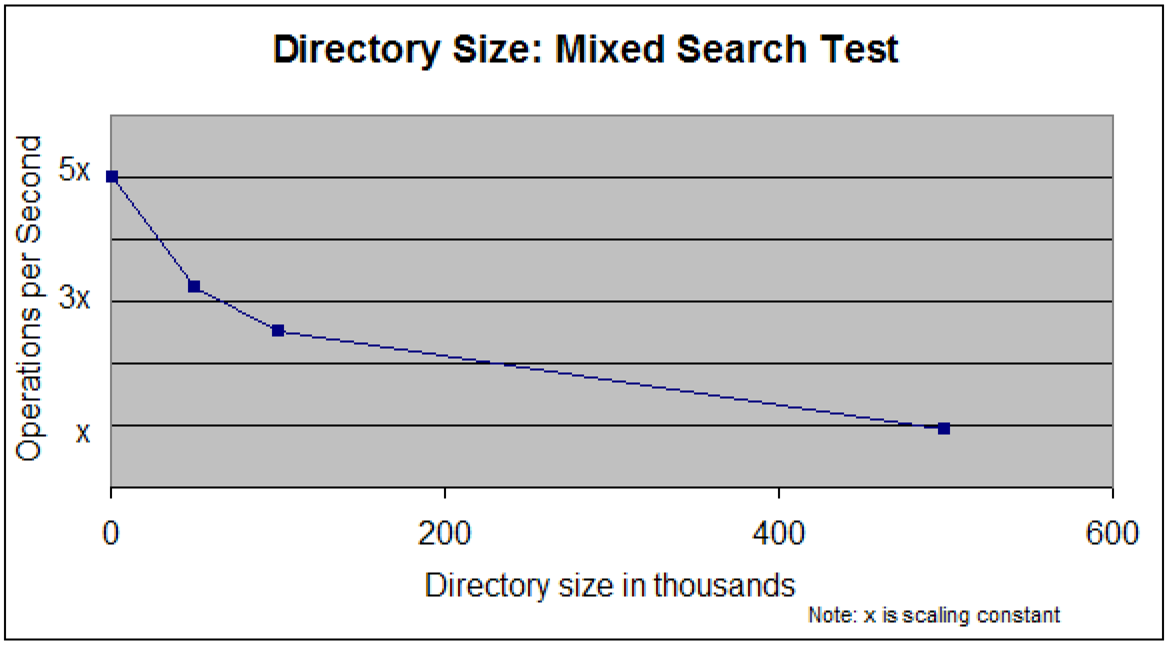

Figure 5. Operations per second

However, the benchmark tool test includes a large fraction of wildcard searches and exact-match searches, such as `sn=Smith`. It returns all entries where the `sn` value is `Smith`. The wildcard and exact-match searches return multiple entries in response to a single search request. The Figure 6 on page 30 figure indicates as the size of the directory grows, the number of entries that are returned for wildcard and exact-match search grows.

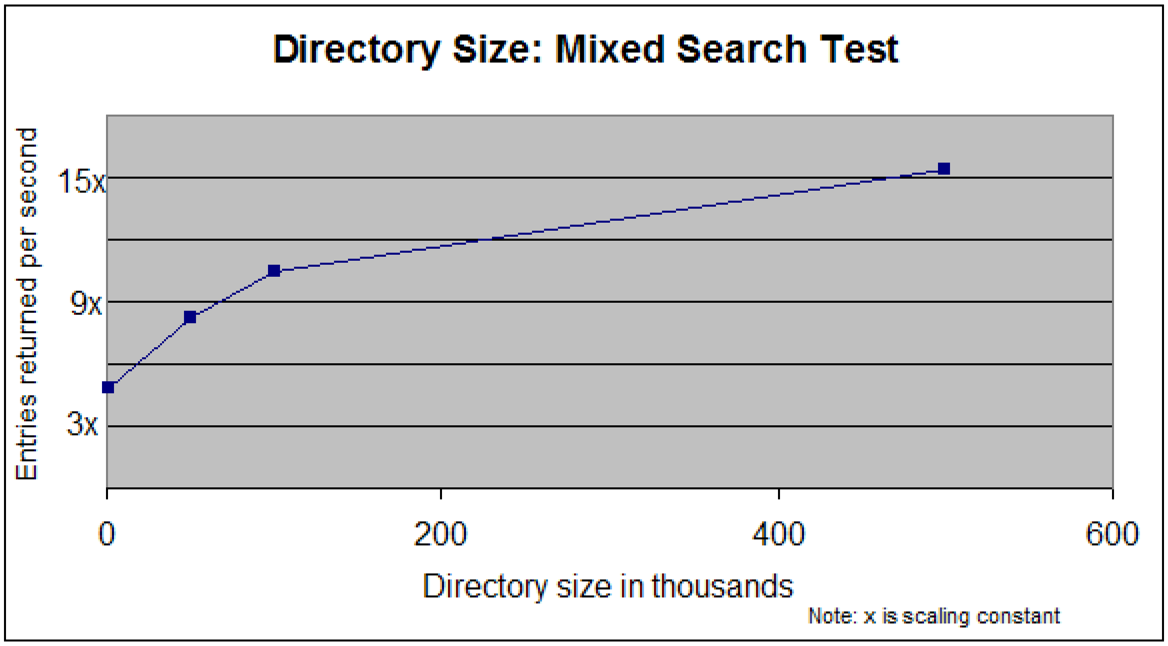

Figure 6. Entries returned per second

The number of entries that are returned per second is a better measure of throughput than operations per second. It is because, each operation requires more time as the size of the database grows.

Note: As your directory grows, it might become necessary to readjust the sizes of the LDAP caches. You can determine the optimal sizes for your caches and buffer pools by using the guidelines in "LDAP caches" on page 20 and "DB2 buffer pool " on page 31. The DB2 buffer pool is tuned automatically.

Tuning DB2 and LDAP caches

You can tune the DB2 database that is configured with a Directory Server instance to reduce the search and update time.

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

Many DB2 configuration parameters affect either the memory (buffer pools) or disk resources. Since disk access is usually much slower than memory access, the key database performance tuning objective is to decrease the amount of disk activity. In DB2, Version 10.5.0.3, you can use Self Tuning Memory Manager (STMM) instead of manually tuning several DB2 parameters. When you use the STMM, DB2 assigns the correct values to memory consumers based on the usage of the system and available resources. You can use the DB2 STMM by setting the values of DB2 buffer pool to `AUTOMATIC`.

When you tune DB2, you must consider the following options:

- Tuning DB2 buffer pool
- Tuning DB2 and LDAP caches by using the **`idsperftune`** tool
- Database maintenance by using the **`idsdbmaint`** tool. You can use the tool for DB2 index organization, DB2 row compression, and table space conversion.
- Database maintenance by using the **`idsrunstats`** tool.
- Optimization and organization (**`reorgchk`** and **`reorg`**)
- Other DB2 configuration parameters

- Backup and restore the database
- Data row compression

For more information about DB2 commands, see the DB2 documentation at http://www-01.ibm.com/support/knowledgecenter/SSEPGG/welcome.

DB2 buffer pool

You can use the DB2 buffer pools to cache entries and their attributes. If the entries are in cache, the search time reduces when querying for the cached data.

Tuning DB2 buffer pool 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. If entries and their attributes are not found in the entry cache, the server searches the DB2 buffer pools for the values. You must tune the buffer pool when the database is initially loaded and when the database size changes significantly. Disabling file system caching is advisable when buffer pools are used. It improves the performance of utilities like **bulkload**, by removing a redundant level of caching.

The Directory Server uses two buffer pools, one for the USERSPACE1 table space and the other for the LDAPSPACE table space. The buffer pool for USERSPACE1 is named IBMDEFAULTBP and the buffer pool for the LDAPSPACE table space is named LDAPBP. For more information about USERSPACE1 and LDAPSPACE, see Table spaces.

There are several points that you must consider are related to DB2 buffer pools. For example:

- If there are no buffer pools, all database activity results in disk access.
- If the size of each buffer pool is too small, LDAP must wait for DB2 disk activity to satisfy DB2 SQL requests.
- If one or more buffer pools are too large, memory on the LDAP server might be wasted.
- If the space that is used by the LDAP caches and the buffer pools is larger than the memory available on a server, operating system paging might occur. Operating system paging might result in increased disk activity.
- Tuning buffer pools do not improve performance of a directory server significantly, it the server contains tens of millions of users. It is not possible or practical to cache a large enough percentage of the database to greatly improve performance. With directories up to millions of users, it is possible to cache a large enough percentage of the database to make an improvement to performance.
- The DB2 buffer pools sizes must be low enough to prevent operating system paging, but high enough to provide maximum benefit.
- Most importantly, current versions of DB2 support automatic tuning of the buffer pools.

The **idsperftune** performance tuning tool sets the DB2 configuration options so that the buffer pools automatically tuned.

Note: Use the **tools** > **db2cmd** utility from virtual appliance command-line interface to run all DB2 commands on virtual appliance.

```
sdsva.example.com> tools db2cmd
```

The DB2 command-line processor is opened.

```
db2 =>
```

To retrieve the current DB2 buffer pool sizes, run the following commands at the DB2 command prompt:

```
!db2 connect to sdsinst1
!db2 select varchar(bpname,20) as bpname,npages,pagesize fromsyscat.bufferpools
```

The following example output shows the default settings:

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

2 record(s) selected.
```

To determine the current file system caching option for each of the table space, run the following commands:

```
!db2 get snapshot for tablespaces on sdsinst1 | egrep ’tablespace name|File system caching’
```

To turn off file system caching with DB2, Version 10.5.0.3 in operating systems and file system environments that support it, run the following command:

```
!db2 connect to sdsinst1
!db2 altertablespace USERSPACE1no file systemcaching
!db2 altertablespace LDAPSPACEno file systemcaching
!db2 terminate
!db2stop
!db2start
```

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

```
!db2 alter bufferpool ibmdefaultbp size new size in 4096 byte pages
!db2 alterbufferpool ldapbpsize new size in 32768 byte pages
!db2 terminate
!db2stop
!db2start
```

If these commands are run while the Directory Server is running, the **db2stop** command fails. An error message is generated indicating there are applications that are connected to the database. If an error is generated, stop the Directory Server and then run the following commands:

```
!db2stop
!db2start
```

To assign optimum memory size for the DB2 buffer pools, you must determine the values. For more information, see “Analyzing DB2 buffer pool performance” on page 33.

If any of the buffer pool sizes are set too high, DB2 fails to start because of insufficient memory. If DB2 fails, the DB2 might generate a core dump file, usually there are no error messages.

If DB2 fails to start because of large buffer pool size, set the buffer pool size to lower values and restart DB2. If you restore a database to a target system from a source system with large buffer pool sizes, the restore operation might fail.

You must also consider upgrading DB2 to latest fix pack level for stability and performance enhancements.

In DB2, Version 10.5.0.3 the `self_tuning_mem` database configuration parameter is automatically set to ON when you create a single-partition database. The value for the parameter is set to AUTOMATIC. The following memory consumers are enabled to tune automatically if you set the value to AUTOMATIC:

- Buffer pools (controlled by the `ALTER BUFFERPOOL` and `CREATE BUFFERPOOL` statements)
- Package cache (controlled by the **pckcachesz** configuration parameter)
- Locking memory (controlled by the **locklist** and **maxlocks** configuration parameters)
- Sort memory (controlled by the **sheapthres_shr** and the **sortheap** configuration parameter)
- Database shared memory (controlled by the **database_memory** configuration parameter)

You must ensure that the database memory is optimally used when the Self Tuning Memory Manager (STMM) is set. You can determine the optimal value for the **DATABASE_MEMORY** parameter. For more information, see “Determining the DATABASE_MEMORY parameter value” on page 34.

You can limit DB2 buffer pools from using all the available memory. To limit the use of memory, consider the following settings before you enable the STMM:

- To use the default automatic values, set the database shared memory size configuration parameter, **DATABASE_MEMORY**:

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

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

  ```
  !db2 ALTER BUFFERPOOL LDAPBP SIZE Determined_Value
  !db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE Determined_Value
  ```

  The `Determined_Value` value optimizes the performance of database.

Analyzing DB2 buffer pool performance

You must analyze the server performance for varying size of DB2 buffer pool to determine the optimum DB2 buffer pool size for your server.

Procedure

1. Run the following command to turn on buffer pool monitoring:

   ```
   !db2 update database manager configuration using DFT_MON_BUFPOOL ON
   !db2stop
   !db2start
   ```

2. Run the following command to connect to the database:

   ```
   !db2 connect to sdsinst1
   ```

   where, `sdsinst1` is the database instance.
3. Start the workload to be analyzed.
4. Reset the monitor data.

   ```
   !db2 resetmonitor all
   ```

5. Obtain the statistics with the workload in progress.
6. Take a snapshot of the buffer pool statistics and process the output. Run the following command to take a snapshot of the current buffer pool statistics:

   ```
   !db2 get snapshot for bufferpools on sdsinst1
   ```

7. Run the following command to take a snapshot and report the read times:

   ```
   !db2 get snapshot for bufferpools on idsdb | awk ’{
   if($1=="Bufferpool" && $2=="name"){print  $0}
   if (index($0,"pool read time")){print "\t"$0}
   }’
   ```

8. Run the following command to take a snapshot and report the number of logical and physical reads:

   ```
   !db2 get snapshot for bufferpools on idsdb | awk ’{
   if($1=="Bufferpool" && $2=="name"){print$0}
   if (index($0,"cal reads")){print "\t"$0}
   }’ | grep-vtemp
   ```

9. Run the following command to take a snapshot and report miss ratios:

   ```
   !db2 get snapshot for bufferpools on idsdb | grep -v temporary |
   awk ’{ if($1=="Bufferpool" && $2=="name"){print$0}
   if (index($0,"logical  reads")){l=$NF;getline;p=$NF;
   ```

```
if (l==0){r=0}else{r=p/l};print "\tMiss ratio: "$3""r}
}'
```

10. Tune the buffer pool sizes such that the IBMDEFAULTBP buffer pool meets the following conditions:
    - A low read time.
    - A low number of reads.
    - A low miss ratio.
    - Must not exceed system physical memory size.
    - Must not reduce the LDAPBP size to less than 2075 pages of 32 KB.

    A higher miss ratio indicates that there are a higher number of physical reads and a lower number of cache hits.
11. Allocate the remaining physical memory to the LDAPBP with the following criteria:

    With file system cache turned off

    ```
    LDAPBP size = ( total physical memory - 1 GB(for DS and DB2)
    - (IBMDEFAULTBP size) * 4096 ) / 32768
    ```

    With file system cache turned on

    ```
    LDAPBP size = ( total physical memory - 1.75GB (for DS,
     DB2, and file system caching) - (IBMDEFAULTBP size) * 4096 ) / 32768
    ```

Determining the DATABASE_MEMORY parameter value

To use database memory efficiently when the Self Tuning Memory Manager (STMM) is set, you must determine the optimal value of **DATABASE_MEMORY** parameter.

Procedure

1. Log in with the database instance owner credentials.
2. Stop the Directory Server instance.
3. Verify that the **DATABASE_MEMORY** parameter is set to AUTOMATIC.
4. On Linux systems, stop and start DB2.

   DB2 adjusts the size of **DATABASE_MEMORY** when DB2 starts.
5. Run the system under normal loads to determine the optimal size of **DATABASE_MEMORY** for the workload.
6. To view the value that is set for **DATABASE_MEMORY** by DB2, run the following command:

   ```
   !db2 get db cfg show detail
   ```
7. To optimize performance, change the **DATABASE_MEMORY** parameter value with the value determined by DB2 instead of AUTOMATIC.
8. On Linux systems, stop and start DB2.
9. Run the system under normal load again.

Results

On Linux systems, DB2 adjusts the size of **DATABASE_MEMORY** when DB2 starts.

DB2 transaction log size

You must tune the transaction log size for the logs to grow to their maximum allowed size.

The Directory Server uses transaction log disk space for storing uncommitted DB2 transactions from directory update operations.

You can set the following DB2 parameters to specify the space that is required by DB2 transaction log:

- `LOGFILSIZ`
- `LOGPRIMARY`
- `LOGSECOND`
- `NEWLOGPATH`

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

```
!db2 get database configuration for sdsinst1 | \
egrep 'LOGFILSIZ|LOGPRIMARY|LOGSECOND|NEWLOGPATH|Path to logfiles'
 Log file size (4KB)(LOGFILSIZ) = 2000
 Number of primary log files(LOGPRIMARY) = 8
 Number of secondary log files(LOGSECOND) = 3
 Changed path to log files(NEWLOGPATH) =
```

The transaction log size is limited by the values of DB2 parameters `LOGFILSIZ`, `LOGPRIMARY`, and `LOGSECOND`. The log size is also affected by the disk space in the directory that is specified by the `NEWLOGPATH` DB2 parameter.

If the transaction log size exceeds the limit that is set, the transaction is backed out by using the information in the logs. If the transaction logs exceed the limit because of the lack of disk space, the database becomes corrupted and unusable. If the database becomes corrupted, it is possible to use the DB2 commands to recover the database. The database can also be restored from a backup or reloaded. If the database becomes corrupted, you can use the recovery commands in the `sqllib/db2dump/db2diag.log` file. The file is in the DB2 instance owner home directory.

By default, the DB2 transaction log file size (`LOGFILSIZ`) is defined to 2000 blocks of 4 KB or 8000 KB per log file. The number of primary logs files (`LOGPRIMARY`) is defined to 8 and the number of secondary log files (`LOGSECOND`) is 3. To increase the DB2 transaction log limits for millions of users, it is necessary to increase the size of the transaction logs (`LOGFILSIZ`) and increase the number of secondary files (`LOGSECOND`). You must increase the number of secondary files rather than the number of primary files. It is because the secondary files periodically get deleted when not in use.

The transaction log requirements are small for a Directory Server with a normal workload. It is observed that runtime directory operations increases the transaction log requirements for a short time.

- The **`ldapadd`** or **`ldapmodify`** commands use some amount of transaction log space. It is because log space is required when you use a command to add a number of multivalued attributes to a single LDAP entry. For example, when you load many members into a group.
- An ACL placed on a suffix object can result in the propagation of the ACL to every entry under that suffix. The Directory Server runs ACL propagation as one single committed DB2 transaction.

You can set the transaction log by running the following DB2 commands:

```
!db2 update database configuration for LOGFILSIZE using 10000
!db2 update database configuration for LOGPRIMARY using 2
!db2 terminate
!db2 force applications all
!db2 connect to sdsinst1
!db2 get database configuration for sdsinst1 | \
 egrep 'LOGFILSIZ|LOGPRIMARY|LOGSECOND|NEWLOGPATH|Path to log files'
```

You can set the `LOGFILSIZ` and `NEWLOGPATH` parameters by using the **`idsperftune`** tool.

DB2 query optimization for aliased object subtree and one level searches

To retrieve one or subtree level search results faster when using aliased objects, the SQL queries are optimized to use literal values of parent or ancestor respectively.

When Directory server is using aliased objects as part of DIT, then SQL corresponding to single level and subtree scope searches uses parameter markers to bind the values of parent and ancestor eid values. Based on the DIT layout and size, these SQL may have some performance impact. To improve performance, use the environment variable **`IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS`**, which is a configurable option. Use the variable to change the SQL to either use parameter markers or literal values.

1. When **IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS** environment variable is not set (default) or set to FALSE or set to a value other than TRUE:

   Aliased object related SQL use parameter markers.
2. When **IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS** environment variable is set to TRUE:

   Aliased objects related SQL use literal values of PEID and AEID values in Single Level and Subtree scope searches respectively.

You can set the **IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS** environment variable by using one of the following approaches:

Using the LDAP client utility

1. Run **ldapmodify** command by using the `ldif` file:

   `idsldapmodify -p <port> -D <adminDN> -w <adminPW> -i configupdate.ldif`

   where `configupdate.ldif` contains the following lines:

   ```
   dn: cn=Front End, cn=configuration
   changetype: modify
   add: ibm-slapdSetEnv
   ibm-slapdSetEnv: IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS=TRUE
   ```
2. Restart the directory server instance from the IBM Security Directory Server virtual appliance command line interface or local management interface.

Using the command prompt

1. Use the **idsenvvars** command to specify the environment variable and value.
   - To add the environment variable, use the following command:

     `idsenvvars -a IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS -v TRUE`
   - To modify the environment variable value, use the following command:

     `idsenvvars -m IBMSLAPD_USE_LITERAL_FOR_ALIASEDOBJECTS -v TRUE`
2. Restart the directory server instance from the virtual appliance command line interface or local management interface.

Directory Server and DB2 database

To use IBM Security Directory Suite Directory Server as a repository in your organization for identity and access management, you must configure a DB2 database with the Directory Server. A Directory Server stores a representation of the directory in a DB2 database.

To implement a directory representation, Directory Server uses database tables. You can use commands to list the database tables that are associated with a Directory Server. It is not necessary to access a Directory Server with DB2 commands, this information might be useful to database administrators.

The Directory Server tables can be grouped into the following categories:

- LDAP entry table
- Subtree tables
- Attribute tables
- ACL tables
- Replication tables

In the examples, the `sdsinst1` DB2 database name is used. To view the table that is associated with the database, you must use the database instance owner credentials. For your environment, substitute the database instance owner and database name as per your configuration. You must switch the user context to the DB2 instance owner to run the commands.

To connect to the database, run the following command:

```
!db2 connect to sdsinst1
```

LDAP entry table

The LDAP entry category consists of a single table, the LDAP_ENTRY table.

The LDAP_ENTRY table contains the LDIF definition of each LDAP entry. One of the columns in the table is the EID (Entry ID) column. All other tables of the database use the EID column to identify the LDAP entry that is referenced from the LDAP_ENTRY table. Directory Server uses the LDAP_ENTRY table in the following ways:

- To retrieve the requested attribute values for a **ldapsearch** command.
- To evaluate the one level scope on a **ldapsearch** command.

The one level scope is evaluated by using the EID (Parent EID) column of the LDAP_ENTRY table. To include indexes on the distinguished name (DN), the LDAP_ENTRY table includes a DN_TRUNC column and a full non-searchable DN column.

To describe the LDAP_ENTRY table, run the **db2 describe** command. The **show detail** parameters are optional.

```
!db2 describe table ldap_entry show detail
```

To find the EID of a particular DN, run the following command. The dn_trunc value must be in uppercase.

```
!db2 "select eid from ldap_entry where dn_trunc = 'CN=USER1,O=SAMPLE'"
```

To find the DN entry name of a particular EID, run the following command:

```
!db2 "select dn_trunc from ldap_entry where eid = 100"
```

To find the LDIF definition of a particular DN, run the following command:

```
!db2 "select ENTRYDATA from dap_entry where dn_trunc = 'CN=USER1,O=SAMPLE'"
```

To find the DN entries for the first 10 rows in the LDAP_ENTRY table, run the following command:

```
!db2 "select dn_trunc from ldap_entry fetch first 10 rows only"
```

To find the DN entries for the next 10 rows in the LDAP_ENTRY table, run the following command:

```
!db2 "select dn_trunc from ldap_entry where eid > 10 fetch first 10 rows only"
```

To find all LDAP suffixes, run the following command:

```
!db2 "select dn_trunc from ldap_entry where peid = -1"
```

To find the DN entries of all the immediate child entries (one level search) of the LDAP entry with DN O=SAMPLE, run the following command:

```
!db2 "select dn_trunc from ldap_entry where peid in \
(select eid from ldap_entry where dn_trunc = 'O=SAMPLE')"
```

Subtree tables

The subtree category consists of the LDAP_DESC table and the LDAP_GRP_DESC table. You can evaluate the subtree scope on a **ldapsearch** command with the LDAP_DESC table. This table contains a list of parent and child LDAP entry relationships in two columns:

- A Descendant EID or DEID column
- An Ancestor EID or AEID column

For each LDAP entry, there is a full list of parents for that LDAP entry in the LDAP_DESC table. Parent in this case includes immediate parent and all ancestors. For example, the following command lists all the parents or ancestors of EID 100:

```
!db2 "select * from ldap_desc where deid = 100"
```

An example output that the command generates:

```
DEID   AEID
------ -------
100     11
100     17
100     23
100     24
100    100
```

The output indicates that the EID is four levels deep in the directory information tree.

To find all the parent entries of an entry with the EID value 100, run the following command:

```
!db2 "select dn_trunc from ldap_entry where eid = 100"
DN_TRUNC
---------------------
CN=TESTUSER1,CN=USERS,OU=HRGROUP,OU=MYCITY,O=SAMPLE
```

The parent entries along with the entry that match the filter is generated.

```
CN=TESTUSER1,CN=USERS,OU=HRGROUP,OU=MYCITY,O=SAMPLE
CN=USERS,OU=HRGROUP,OU=MYCITY,O=SAMPLE
OU=HRGROUP,OU=MYCITY,O=SAMPLE
OU=MYCITY,O=SAMPLE
O=SAMPLE
```

You can join DB2 tables in a single command to list all the parents of an LDAP entry with the CN=TESTUSER1,CN=USERS,OU=HRGROUP,OU=MYCITY,O=SAMPLE DN. For example:

```

!db2 "select * from ldap_desc where aeid in \
(select eid from ldap_entry where dn_trunc = \
'CN=TESTUSER1,CN=USERS,OU=HRGROUP,OU=MYCITY,O=SAMPLE')"
```

The LDAP_DESC table is also used in subtree searches. To find all the child LDAP entries (both immediate and all descendants) for the CN=USERS,O=SAMPLE LDAP entry, run the following command:

```
!db2 "select * from ldap_desc where aeid in \
(select eid from ldap_entry where dn_trunc = \
'CN=USERS,O=SAMPLE')"
```

An example output:

```
DEID      AEID
-------- -----------
     12    12
2000042    12
2000043    12
2000044    12
2000056    12
2000057    12
2000058    12
```

You can use the LDAP_GRP_DESC table to track nested group relationships.

Attribute tables

The attribute tables consist of one table per attribute that is used in the Directory Server. The purpose of the attribute tables is to improve the performance of the LDAP searches with the search filters, particularly when the attribute is indexed. The attribute tables are named by the attributes they represent. For example, the DB2 table for the cn attribute is named cn.

To describe the `cn` table, run the following command:

```
!db2 describe table cn
```

The example output of the command is as follows:

```
Column name    Data type   Data type   Column  Scale  Nulls
               schema      name        Length
-------------- ----------     ------------ ------- ----- ------
EID            SYSIBM      INTEGER          4      0    No
CN             SYSIBM      VARCHAR        256      0    No
CN_T           SYSIBM      VARCHAR        240      0    No
RCN_T          SYSIBM      VARCHAR        240      0    No
 4 record(s) selected.
```

The `CN` column contains the full name for the attribute. The values in column, Column name, with names that end with T are truncated to 240 character attribute name that is used for searching. The column name beginning with R is the attribute name in reverse. This column is used for searching for attributes that are specified with a trailing wildcard.

ACL tables

The ACL tables consist of the SRC, `ACLPROP`, `OWNPROP`, `ENTRYOWNER`, `ACLPERM`, and `ACLINHERIT` tables. The SRC table identifies from which LDAP entry a particular LDAP entry obtains the source for or inherits its ACL and owner information. The SRC table is also the attribute table for the `aclsource` and `entryowner` attributes.

To describe the `src` table, run the following command:

```
!db2 describe table src
```

The example output of the command is as follows:

```
Column name          Data type Data type name      Column     Scale Nulls
                     schema                        Length
-------------------- --------- ------------------ ---------- ----- ------
EID                  SYSIBM     INTEGER                   4      0    Yes
ACLSRC               SYSIBM     INTEGER                   4      0    Yes
OWNSRC               SYSIBM     INTEGER                   4      0    Yes
ACLTYPE              SYSIBM     INTEGER                   4      0    Yes

4 record(s) selected.
```

The `ACLPROP` and `OWNPROP` are the attribute tables for the `aclpropagate` and `ownerpropagate` attributes. The `ACLPERM` table is the attribute table for the `aclentry` attribute. The `ACLINHERIT` table is the attribute table for the `ibm-filterAclEntry` attribute.

Replication tables

The replication tables consist of the REPLSTATUS, REPLCHGnnnn, REPLERROR, and several other tables. There is one REPLCHGnnnn table for each replication context. Where, *nnnn* is `EID` of the base entry of the replication context. The REPLCHGnnnn implements the replication change table.

The REPLSTATUS table is a pointer to the REPLCHGnnnn table that indicates the last replicated operation.

Database connections

You can improve the performance of a Directory Server by increasing the DB2 connections. The performance improvement also depends on the server load and the nature of connections.

The default number of connections that are created between a directory server and DB2 is 15. The default value suffices for most environment in which a directory server is used. Based on the requirement, you can increase the database connection to unlimited. The upper limit that you can assign is *INT_MAX* (2147483647). You can modify the `ibm-slapdDbConnections` attribute to set the database connections. If you assign a value of less than or equal to zero, or greater than *INT_MAX*, then the default value of 15 is set.

You must determine the number of worker threads in use to assign an appropriate value for database connections. To find the number of worker threads, use the `available_workers` threads value in the monitor search result. To increase the workers threads or to increase the back-end connections, you must set `ibm-slapdDbConnections` under the `cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration` DN entry. To view the monitor search result, run the **`idsldapsearch`** command:

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

The performance tuning tool

You can use the **`idsperftune`** tool to tune directory caches, DB2 buffer pools, and DB2 parameters to improve the Directory Server performance.

The **`idsperftune`** tool (the performance tuning tool) is available in IBM Security Directory Suite. You must run the **`idsperftune`** tool against a directory server instance that is configured with DB2 database. If the tool is run against a Proxy Server instance, the tool generates an appropriate error message and exits.

You must provide inputs to the **`idsperftune`** tool to tune the Directory Server. To provide inputs to the **`idsperftune`** tool, you must update the `perftune_input.conf` property file by using the administrator credentials. If the inputs are not provided, then the default values are considered.

You can manually update the `perftune_input.conf` property file for a Directory Server instance at the following location:

`instance_home``/idsslapd-`*`instance_name`*`/etc`.

You can also find the installed version of `perftune_input.conf` file (for reference only) at the following location. The default location for the `perftune_input.conf` file is `/opt/ibm/ldap/V8.0.1.x/etc/`.

For information about the performance tuning tool, see the *Command Reference* section of the IBM Security Directory Suite documentation.

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

Basic tuning mode

You can use the performance tuning tool to run basic tuning on a Directory Server with the existing server statistics.

In the basic tuning mode, the **`idsperftune`** tool tunes the following server components:

LDAP caches
The LDAP caches include entry cache, filter cache, group member cache, and group member cache bypass limit.

DB2 buffer pools
These include `IBMDEFAULTBP` and `LDAPBP`.

The basic tuning mode suggests the optimum tuning values for LDAP caches and DB2 buffer pools. The tool also updates the LDAP cache and DB2 buffer pool parameters with the values that are determined. These recommendations are based on the following inputs:

Amount of free system memory (%) to allot to a Directory Server instance
It specifies the total memory that is allocated to an instance. This value is also used as an input to the tool when you tune the size of entry cache, filter cache, and group member cache. If a value is not specified, when **`idsperftune`** is run the default value of 90% of system memory available is allocated.

The number of entries and the average size of an entry that is in a Directory Server instance
The total number of entries that can be stored in a directory is used to estimate the required cache size.

Another value that is used as input for the tool is the average size of an entry in bytes.

The average size of an entry and the total number of entries are used by the **idsperftune** tool to calculate the total size of the directory. Based on these values, the size that must be allotted to the Entry cache and Filter cache are calculated.

Note: The **idsperftune** tool provides a parameter to calculate the total number of entries and average size of entries that are present in the directory. For example, if you run **idsperftune** with the **-s** parameter, the tool computes the total number of entries and average size of entries and logs the details in the `perftune_input.conf` file. If you do not provide the parameter, then the total number of entries is set to 10000 by default.

Update frequency:
You must specify whether frequent updates or only batch updates to run. If you specify frequent updates are expected, then the filter cache is set to 0. Otherwise, it is set to 1 KB.

Total number of groups to be cached:
You can tune this value by providing the approximate value for the total number of groups whose members must be cached. This value must be the number of groups frequently used. If not specified, the default value of 25 is set.

Average number of members in a group:
You can tune the total number of members within a group to cache. If the value is not specified, the default value of 25000 is set.

Server instance name:
The instance name is retrieved from the *IDS_LDAP_INSTANCE* environment variable. If the environment variable is not set, then the server instance name is set with an available Directory Server instance name. If more than one instance is available and no instance name is provided by the administrator, then an appropriate error message is generated.

When you run **idsperftune** with the **-u** parameter to update the configuration settings, the DB2 buffer pools `IBMDEFAULTBP` and `LDAPBP` are set to AUTOMATIC. The DB2 Self Tuning Memory Manager (STMM) dynamically adjust the sizes of the DB2 buffer pools when the DB2 buffer pools are set to AUTOMATIC. When you run the **idsperftune** tool, the memory size to allocate for LDAP entry cache is calculated. When you run the **idsperftune** tool in basic mode by providing the **-B** and **-u** parameters, the DB2 buffer pools are set to AUTOMATIC.

If the available memory can cache at least 80% of the entries, then 80% or more of total entries is cached in the entry cache. To override the default requirement to cache at least 80% of entries in LDAP entry cache, run **idsperftune** with the **-E** parameter and the target percentage of entries to be cached. For example, to cache a minimum of 50% of entries in the entry cache run the following command:

```
idsperftune -I instance_name -E 50
```

If the allotted memory is not enough to cache 80% of total entries, then the entry cache is set to a minimum value of 1000.

SYS_MEM_AVL

If the *SYS_MEM_AVL* variable in the `perftune_stat.log` file is set to TRUE, then 80% of the directory entries are cached. If the *SYS_MEM_AVL* variable is set to FALSE, a minimum amount of system memory is allotted to LDAP entry cache. The remaining system memory is allotted to DB2 buffer pools.

Examples

Example 1:
To retrieve the basic tuning recommendations, run the **idsperftune** command with the following parameters:

```
idsperftune –I instance_name -B
```

Example 2:
To update the database with the suggested values during basic tuning, run the **idsperftune** command with the following parameters:

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

or

```
idsperftune –I instance_name –u
```

Example 3:
To update the property file with the total number of entries and average entry size, run the **idsperftune** tool with the following parameters:

```
idsperftune -I instance_name -s
```

Advanced tuning mode

You can use the performance tuning tool to run advanced tuning on a Directory Server with the values set in the property files.

For the advanced tuning, you must configure the Directory Server and populate the server with entries. To obtain better values to tune, you must ensure that the Directory Server is servicing client requests for some time. You can run the **idsperftune** tool against a server on which no operations are run to obtain values for basic or advanced tuning mode. In advanced tuning mode, when you set the DB2 monitor switches to ON it initiates data collection. You can use the data to tune DB2. To set DB2 monitor switches to ON, you must run the **idsperftune** command with the **-m** parameter. If you want tune the server that is loaded with a typical workload, you must run **idsperftune** with the **-A** and **-m** parameters. This command sets the DB2 monitor switches to ON and then waits for some minutes before it collects the statistics. The tuning is based on the workload you ran against the server before you collected the DB2 monitor information. You can also set the DB2 monitor switches to OFF by running the **idsperftune** command with the **-o** parameter. You can run **idsperftune** with the appropriate advanced tuning options to monitor differentDB2 parameters. For more information about the**idsperftune** tool, see the *Command Reference* section of IBM Security Directory Suite documentation.

You can monitor the following DB2 parameters at server run time:

PCKCACHESZ
The **PCKCACHESZ** parameter is allocated out of the database shared memory. It is used for caching of sections for static and dynamic SQL and XQuery statements on a database.

LOGFILSIZ
The **LOGFILSIZ** parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written before they become full.

LOGBUFSZ
The **LOGBUFSZ** parameter specifies the amount of the database heap (defined by the **dbheap** parameter). You can use the parameter as a buffer for log records before you write these records to disk.

SORTHEAP
The **SORTHEAP** parameter defines the maximum number of private memory pages to use for private sorts. It also defines the maximum number of shared memory pages to use for shared sorts.

MAXFILOP
The **MAXFILOP** parameter specifies the maximum number of file handles that can be opened for each database agent.

DBHEAP
The **DBHEAP** parameter specifies the maximum memory that is used by the database heap.

CHNGPGS_THRESH
The **CHNGPGS_THRESH** parameter specifies the percentage of changed pages at which the asynchronous page cleaners start.

NEWLOGPATH
The **NEWLOGPATH** parameter specifies the location where the log files are stored. The path must not be more than 242 bytes in length.

When you run the **idsperftune** tool, it checks whether the self-tuning memory is enabled. When you run **idsperftune** with the **-A** and **-u** parameters, the tool enables the self-tuning memory if it is disabled. Self tuning memory sets the values for memory configuration parameters automatically and buffer pools size. When the self-tuning memory is enabled, the memory tuner dynamically distributes available memory resources among several memory consumers. The memory consumers include the sort, package cache, lock list areas, and buffer pools.

The **idsperftune** tool checks if DB2 parameters, such as **SELF_TUNING_MEM**, **AUTO_MAINT**, **AUTO_TBL_MAINT**, and **AUTO_RUNSTATS**, are set to ON. When you run **idsperftune** with the **-A** and **-u** parameters, then these parameters are automatically set to ON if the values of the variables are OFF. The **AUTO_TBL_MAINT** parameter is the key parameter for other table maintenance parameters (**AUTO_RUNSTATS**, **AUTO_STATS_PROF**, **AUTO_PROF_UPD**, and **AUTO_REORG**). To enable **AUTO_MAINT**, the **AUTO_TBL_MAINT** parameter must be set to ON.

The **idsperftune** tool also checks for DB2 parameters, such as **LOCKLIST**, **NUM_IOSERVERS**, and **NUM_IOCLEANERS**, are set to AUTOMATIC. When you run **idsperftune** with the **-A** and **-u** parameters, then these variables are set to AUTOMATIC.

Table 1. DB2 parameters with their type and values

DB2 parameters	Type	Value in the `perftune_stat.log` file
NUM_IOCLEANERS	AUTOMATIC	AUTOMATIC / Numeric value
NUM_IOSERVERS	AUTOMATIC	AUTOMATIC / Numeric value
LOCKLIST	AUTOMATIC	AUTOMATIC / Numeric value
SELF_TUNING_MEM	AUTOMATIC	ON / OFF
AUTO_MAINT	AUTOMATIC	ON / OFF
AUTO_RUNSTATS	AUTOMATIC	ON / OFF
AUTO_TBL_MAINT	AUTOMATIC	ON / OFF
IBMDEFAULTBP	AUTOMATIC	AUTOMATIC
LDAPBP	AUTOMATIC	AUTOMATIC

To monitor DB2 parameters **SORTHEAP**, **MAXFILOP**, **DBHEAP**, **CHNGPGS_THRESH**, **NUM_IOSERVERS**, and **NUM_IOCLEANERS**, you must enable monitor switches BUFFERPOOL and SORTHEAP. To enable the monitor switches, run the **idsperftune** tool with the **-m** parameter. DB2 collects more run time data when you set the monitor switches. However, enabling these monitor switches might negatively affect the performance of a Directory Server. If you do not set monitor switches BUFFERPOOL and SORTHEAP, then the status of these parameters is set to `"Not Collected"` in the `perftune_stat.log` file. If you enable monitor switches, then the suggested values are updated for the parameters in the log file.

When the **idsperftune** tool updates the Directory Server and DB2 configuration parameters, it stores the previous values in the `perftune_stat.log` file. The initial values that existed in the file are recorded under the INITIAL TUNING PARAMETER VALUE section with the prefix `I_`. For example, `I_MAXFILOP`. When the tool updates any DB2 configuration settings, the previous values are stored under the OLD DB2 PARAMETER VALUE section with the prefix `O_`. For example, `O_LOGFILSIZ`.

The **idsperftune** tool determines the values for DB2 parameters and lists in the following format:

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

The parameter might be updated with one of the following values:

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

For example, the **idsperftune** tool suggests increasing the value that is set in the **PCKCACHESZ** parameter.

```
PCKCACHESZ=1533:Increase
```

With this input, you can find the current value and the action you must take.

The description of the DB2 parameters status is listed.

Not Collected
Specifies that the value of DB2 parameter is not monitored.

OK
Specifies that the value currently set for the DB2 parameter is optimal.

Increase
Specifies that the value of DB2 parameter must be increased to achieve optimal performance.

Decrease
Specifies that the value of DB2 parameter must be decreased to achieve optimal performance.

Examples

Example 1:
To retrieve advanced tuning values without setting the monitor switches to ON, run the **idsperftune** command with the following parameters:

```
idsperftune -I instance_name -A
```

Example 2:
To set monitor switches to ON for DB2 parameters, run the **idsperftune** command with the following parameters:

```
idsperftune -I instance_name -m
```

Example 3:
To set monitor switches to OFF for DB2 parameters, run the **idsperftune** command with the following parameters:

```
idsperftune -I instance_name -o
```

Example 4:
To update the database with the suggested DB2 parameter values without setting the monitor switches to ON, run the **idsperftune** command:

```
idsperftune -I instance_name -A -u
```

Example 5:
To obtain advanced tuning recommendations with the monitor switches set to ON, run the **idsperftune** command:

```
idsperftune -I instance_name -A -m
```

After the **idsperftune** command completes the operation, the monitor switches are set to OFF.

Example 6:
To update the database with the suggested DB2 parameter values with the monitor switches set to ON, run the **idsperftune** command:

```
idsperftune -I instance_name -A -u -m
```

After the **idsperftune** command completes the operation, the monitor switches are set to OFF.

The performance tuning tool input file

To tune a Directory Server with the **idsperftune** command, you must update the `perftune_input.conf` property file with the configurable values.

The `perftune_input.conf` property file contains a list of inputs for the **idsperftune** command in the form attribute-value pairs. To update the `perftune_input.conf` file, you must log in with Directory Server instance owner credentials. If you do not want to specify values for the attributes, then you must leave the values as None.

You must update the attribute values as per your requirements, and then run the tool by providing the property file as an input. You must ensure that the attribute names are not modified in the property file. If the attribute names are modified, the **idsperftune** tool generates an appropriate error message and exits. The `perftune_input.conf` file is in *instance-home*/idsslapd-*inst_name*/etc. The format of an example `perftune_input.conf` file is as follows:

```
#-----------------------------------------------------
#Admin Input
#-----------------------------------------------------
# Amount of system memory (%) to be allotted to TDS instance
TDS_SYS_MEM=90
# Total number of entries that will reside in the directory
TDS_TOTAL_ENTRY=10000
# Average size of entry (Bytes)
TDS_AVG_ENTRY_SZ=2560
# Update Frequency
#1. Frequent updates expected, or
#2. Only Batch Updatesexpected
TDS_UPDATE_FREQ=1
#Total number of Groups to be cached
TDS_GROUP_CACHE=25
# Maximum number of members in a group that will be referenced frequently
TDS_GROUP_MEMBER=25000
#
#------------------------------------------------------
# DB2 PARAMETER INPUT
#------------------------------------------------------
# NEWLOGPATH allows you to specify a string of up to 242 bytes to change
# the location where the log files are stored. Eg, NEWLOGPATH="/newdevice"
NEWLOGPATH=None
# LOGFILSIZ defines the size of each primary and secondary log file. The size
# of these log files limits the number of log records that can be written to
# them before they become full and a new log file is required.
LOGFILSIZ=None
# DBHEAP determines the maximum memory used by the database heap.
DBHEAP=None
# PCKCACHESZ is allocated out of the database shared memory, and is used for
# caching of sections for static and dynamic SQL and XQuery statements on
# a database.
PCKCACHESZ =None
# LOGBUFSZ allows you to specify the amount of the database heap (defined
# by the dbheap parameter) to use as a buffer for log records before writing
# these records to disk.
LOGBUFSZ=None
# MAXFILOP specifies the maximum number of file handles that can be open
# for each database agent.
MAXFILOP=None
# CHNGPGS_THRESH specifies the level (percentage) of changed pages at which
# the asynchronous page cleaners will be started, if they are not currently active.
CHNGPGS_THRESH=None
# SORTHEAP defines the maximum number of private memory pages to be used
# for private sorts, or the maximum number of shared memory pages to be
# used for shared sorts.
SORTHEAP=None
```

The performance tuning tool statistics file

You can use the `perftune_stat.log` property file to store the performance values that are suggested by the **idsperftune** command. Based on the values in the `perftune_stat.log` file, you must update the values tune a Directory Server instance.

The information that is gathered during the basic tuning and advanced tuning phases are logged in the `perftune_stat.log` property file. The log file provides what action to take on a DB2 parameter value to get better performance from the Directory Server. The `perftune_stat.log` file is in *`instance-home`*`/idsslapd-`*`inst_name`*`/logs`.

A sample `perftune_stat.log` file is generated from a Directory Server instance on a Linux system. The Directory Server is loaded with entries from the `/opt/ibm/ldap/V8.0.1.x/examples/sample.ldif` file.

The format of a sample `perftune_stat.log` file is as follows:

```
#-------------------------------------------------------
# Perftune Basic tuning parameters
#-------------------------------------------------------
#-------------------------------------------------------
# Directory Cache
#-------------------------------------------------------
TDS_ENTRY_CACHE=1000
TDS_FILTER_CACHE=0
TDS_GROUP_CACHE=25
TDS_GROUP_MEMBER=25000
#------------------------------------------------------
# DB2 BUFFERPOOL (Number of pages)
#------------------------------------------------------
IBMDEFAULTBP=AUTOMATIC
LDAPBP=AUTOMATIC
#------------------------------------------------------------------
# System memory allotted to Directory Server Instance (Kilo Bytes)
#------------------------------------------------------------------
SYSTEM_MEMORY=102417.12
#-------------------------------------------------------
# Will be set to True if enough system memory is available to
# the directory instance to make directory caching effective
#-------------------------------------------------------
SYS_MEM_AVL=FALSE
#-------------------------------------------------------
# Perftune Advance tuning parameters
#-------------------------------------------------------
# NEWLOGPATH allows you to specify a string of up to 242 bytes to
# change the location where the log files are stored.
NEWLOGPATH=None
#-------------------------------------------------------------------
# DB2 PARAMETER STATUS
# <DB2 parameters>=<Current Value>:<Recommendation>
# Recommendation can be <Not Collected>/<OK>/<Increase>/<Decrease>
#-------------------------------------------------------------------
# LOGFILSIZ defines the size of each primary and secondary log file.
# The size of these log files limits the number of log records that can
# be written to them before they become full and a new log file is required.
LOGFILSIZ=2000:OK
# PCKCACHESZ is allocated out of the database shared memory, and is used
# for caching of sections for static and dynamic SQL and XQuery statements
# on a database.
PCKCACHESZ=2299:Increase
# LOGBUFSZ allows you to specify the amount of the database heap
# (defined by the dbheap parameter) to use as a buffer for log records
# before writing these records to disk.
LOGBUFSZ=98:OK
# MAXFILOP specifies the maximum number of file handles that can be open
# for each database agent.
MAXFILOP=Not Collected
# CHNGPGS_THRESH specifies the level (percentage) of changed pages at
# which the asynchronous page cleaners
# will be started, if they are not currently active.
CHNGPGS_THRESH=Not Collected
# SORTHEAP defines the maximum number of private memory pages to be used for
# private sorts, or the maximum number of shared memory pages to be used for
# shared sorts.
SORTHEAP=Not Collected
#------------------------------------------------------------
# DB2 parameters whose value will be automatically set by
# DB2 self tuning memory manager
# <DB2 parameters>=<Current Value>/<AUTOMATIC>
#------------------------------------------------------------
# Indicates the amount of storage that is allocated to the lock list.
# There is one lock list per database and it contains the locks held by all
# applications concurrently connected to the database.
LOCKLIST=AUTOMATIC
# Number of I/O servers configuration parameter
NUM_IOSERVER=AUTOMATIC
# Number of asynchronous page cleaners configuration parameter
```

```
NUM_IOCLEANER=AUTOMATIC
#-------------------------------------------------------
# DB2 AUTOMATIC PARAMETERS
# <DB2 parameters>=<ON>/<OFF>
#-------------------------------------------------------
# SELF_TUNING_MEM determines whether the memory tuner will dynamically distribute
# available memory resources as required between memory consumers that
# are enabled for self tuning.
SELF_TUNING_MEM=ON
# AUTO_MAINT Automatic maintenance configuration parameter
AUTO_MAINT=ON
# This parameter is the parent of all table maintenance parameters
# (auto_runstats, auto_stats_prof, auto_prof_upd, and auto_reorg).
AUTO_TBL_MAINT=ON
# AUTO_RUNSTATS Automatic table maintenance configuration parameter
AUTO_RUNSTATS=ON
#-------------------------------------------------------
# OLD TDS CACHE PARAMETER ( Prior to last Update Operation )
#-------------------------------------------------------
O_TDS_ENTRY_CACHE=25000
O_TDS_FILTER_CACHE=25000
O_TDS_GROUP_CACHE=25
O_TDS_GROUP_MEMBER=25000
#-------------------------------------------------------
# OLD DB2 PARAMETER VALUE ( Prior to last Update Operation )
#-------------------------------------------------------
O_IBMDEFAULTBP=AUTOMATIC
O_LDAPBP=AUTOMATIC
O_PCKCACHESZ=2299
O_LOGBUFSZ=98
O_MAXFILOP=64
O_CHNGPGS_THRESH=80
O_SORTHEAP=355
O_DBHEAP=2333
O_NEWLOGPATH=None
O_LOGFILSIZ=2000
#-------------------------------------------------------
# INITIAL TUNING PARAMETER VALUE ( Prior to First Update Operation )
#-------------------------------------------------------
I_TDS_ENTRY_CACHE=25000
I_TDS_FILTER_CACHE=25000
I_TDS_GROUP_CACHE=25
I_TDS_GROUP_MEMBER=25000
I_IBMDEFAULTBP=AUTOMATIC
I_LDAPBP=AUTOMATIC
I_PCKCACHESZ=2299
I_LOGBUFSZ=98
I_MAXFILOP=64
I_CHNGPGS_THRESH=80
I_SORTHEAP=355
I_DBHEAP=2333
I_NEWLOGPATH=None
I_LOGFILSIZ=2000
```

The database maintenance tool

You can use the **idsdbmaint** tool to run database maintenance activities, such as DB2 index reorganization and DB2 row compression.

You must stop the Directory Server instance before you run the **idsdbmaint** tool. When you stop the server and run the **idsdbmaint** tool, it ensures that the database remains in a consistent state after all the database maintenance activities.

Table spaces

A table space is a set of volumes on disks that hold the data sets in which tables are stored. A table space can have one or more tables.

The Directory Server uses four DB2 table spaces.

Table space 0: SYSCATSPACE
The SYSCATSPACE table space stores a description of the database and its structure and contents. The disk requirements for this table space do not change with the size of the directory server. The disk space requirements are covered by the default directory server disk requirements.

Table space 1: TEMPSPACE1

The TEMPSPACE1 table space holds temporary data for sorting and collating DB2 results. The disk requirements for this table space grow at run time if a complex search is run against the Directory Server. The disk space requirement for this table space grows with the usage utilities, such as the **bulkload** tool.

When the **bulkload** utility is run, the disk requirement for the table space is approximately 2 GB. The **bulkload** utility uses the maximum space when millions of entries are loaded into the Directory Server.

Table space 2: USERSPACE1

USERSPACE1 holds the portion of the database that contains the attribute tables and attribute table indexes of the Directory Server. These tables are used for optimizing searches on specific attributes.

Table space 3: LDAPSPACE

LDAPSPACE contains the portion of the database that contains the LDAP entry table and LDAP entry table indexes of the Directory Server. The LDAP entry table contains a few searchable attributes, such as distinguished name (DN), and a full, non-searchable definition of each LDAP entry.

The LDAP entry table returns the attributes for an LDAP search operation.

The default tablespace for the Directory Server is automatic storage tablespace. From DB2, Version 10.1 Fix Pack 1 onwards the database managed spaces (DMS) table space type and system managed spaces (SMS) table space type is deprecated for permanent table spaces that are defined by the user.

DB2 index reorganization

You can run DB2 index reorganization on a Directory Server to eliminate fragmented data in the database table and to restructure the index data. You must run the DB2 index reorganization operation on the Directory Server instance with the instance owner credentials.

When you run the **idsdbmaint** command to initiate DB2 index reorganization, the following operations are run:

- Queries DB2 `sysibm.sysindexes` on the tables of the Directory Server instance, and then fetches all the tables for which indexes are defined.
- Runs index reorganization on all the indexes.
- Updates the table statistics after index reorganization is run on the table.

To optimize the database, you can run the **idsdbmaint** command with the index reorganization parameter. For example:

```
idsdbmaint -I instance_name -i
```

DB2 row compression

You can use DB2 row compression to save on storage requirement. You must run the DB2 row compression operation against a Directory Server instance with the instance owner credentials.

DB2 row compression uses a static dictionary-based compression algorithm to compress data by row. DB2 row compression can replace the repeating patterns that span multiple column values within a row with shorter symbol strings. The row compression on a table is applied by reconstructing the compression dictionary and compressing the information to eliminate fragmented data. Data compression reduces space that is required for the Directory Server, reduces I/O, and improves performance. When you run the **idsdbmaint** command to compress rows, the following operations are run:

- Queries DB2 `syscat.tables` and fetches all the tables of the Directory Server instance.
- Inspects the table and fetches the row compression estimates for each table.
- Runs the following operations, if the compression estimate is more than 30 percent:
  - Alters the table to enable `ROW COMPRESSION`.
  - Runs the DB2 `reorg` command on the table and builds a new compression dictionary.

- Runs the DB2 `runstats` command on the table to update all the statistics on the table.
- Creates a compression dictionary.

To optimize the database, you can run the **`idsdbmaint`** command with the row compression parameter. For example:

```
idsdbmaint -I instance_name -r
```

Optimization and organization of database

You can optimize and organize the DB2 database with the DB2 commands to reduce data access time and retrieval time.

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. The database manager maintains a set of statistics to represent the distributed data.

A Directory Server creates a number of indexes for tables in the database. These indexes minimize the time that is required to access data and locate a particular row in a table.

In a read-only environment, the distribution of the data does not change much. With updates and additions to the database, it is not uncommon for the distribution of the data to change significantly. Similarly, it is possible for data in tables to become ordered in an inefficient manner.

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

Optimization

You must optimize the database and update the table statistics to improve query performance and to reduce data retrieval time.

You must optimize the database periodically or after database updates. For example, after you import entries into a Directory Server instance. You can optimize the database by using the **`idsrunstats`** command or from the **Optimize database** panel of IBM Security Directory Suite **Configuration Tool**. The tool internally calls the **`idsrunstats`** command to update database statistics that is used by the query optimizer for all the LDAP tables.

Note: You can also use the **`reorgchk`** command to update statistics. If you are planning to run **`reorgchk`**, optimizing the database is unnecessary. For more information about the **`reorgchk`** command, see "Database organization (reorgchk and reorg)" on page 50.

You can run the **`idsrunstats`** command to update the DB2 system catalog statistics. The statistics include the physical characteristics of tables and associated indexes in the database of a Directory Server instance. The physical characteristics of a table include number of records, number of pages, and average record length. To collect database statistics, the following parameters, **DB2RUNSTATS_ALL_COLUMNS|DB2RUNSTATS_ALL_INDEXES**, are passed to the `db2Runstats` API. You can run the **`idsrunstats`** command against a Directory Server instance even when the instance is up and running.

To optimize the database, run the following command:

```
idsrunstats –I instance_name
```

The *instance_name* variable is the directory server instance name and is an optional parameter. If your computer contains more that one instance, you must specify the instance name.

After you complete the optimization, you might require to restart the Directory Server if you do not see performance benefits after you flush the package cache.

To know more about the usage of the **`idsrunstats`** command, see the *Command Reference* section of the IBM Security Directory Suite documentation.

Viewing DB2 system statistics settings

You can view the reports of all the system statistic settings in a database. You can use the database statistics to compare with any other database that has better performance and to updates values based on your database usage.

Procedure

1. Log in with the database instance owner credentials.
2. Run the **db2look** command with the **-m** parameter to produce a report that contains the DB2 commands to reproduce the current system statistic settings.

   ```
   !db2look -m -d sdsinst1 -u sdsinst1 -o /userdata/directory/CustomOut/output_file
   ```

Database organization (reorgchk and reorg)

You can tune the organization of the data in a DB2 database to improve database performance and to save disk space.

To tune the organization of the data in DB2 database, you can use the **reorgchk** and **reorg** commands. You can reorganize table spaces to improve access performance and to reorganize indexes so that they are more efficiently clustered.

The **reorgchk** command does the following operations:

- Updates the DB2 optimizer with database statistics to improve database performance.
- Reports statistics on the organization of the database tables.

Based on the statistics that are generated by the **reorgchk** command, you must run the **reorg** command to update database table organization. The **reorgchk** and **reorg** commands can improve both search and update operation performance.

The DB2 **reorg** command reorganizes the table and its indexes by rebuilding the index data into unfragmented and contiguous area on the disk. This operating requires locking the data for movement. If there are applications that access this data, then the application performance might be degraded. You must not run the DB2 **reorg** command while the Directory Server and DB2 database is in production phase. You must run the **reorg** command against the Directory Server and the DB2 database during the maintenance phase.

After you complete the database organization operation, you might require to restart the Directory Server if you do not observe performance benefits. If the SQL statements are cached by DB2 and directory server instance, you might require to clear the package cache and restart the server.

You must run the **reorgchk** command periodically. For example, you must run **reorgchk** after you update a Directory Server instance with entries. The performance of the database must be monitored and you must run **reorgchk** if server performance starts to degrade.

In a replication environment, you must run the **reorgchk** command on all the replica servers because each replica uses a separate database. If database is optimized on a master server, the replication process does not propagate database optimizations to replica servers.

Guidelines for reorganization

You must consider the following guidelines before you run DB2 database reorganization:

- You can skip reorganization of a table or index, if you meet the following conditions:
  - A reorganization attempt is already done.
  - The number on the column that contains an asterisk is close to the suggested value that is described in the header of each section.
- From the LDAP_ENTRY_TRUNC index and the SYSIBM.SQL index in the SDSINST1.LDAP_ENTRY table, preference must be given to the SYSIBM.SQL index.

- When an attribute length is defined with less than or equal to 240 bytes, the attribute table contains three columns: EID, attribute, and reversed attribute. In this case, the forward index is created by using the EID and attribute columns as index keys. For example, the SN attribute is defined with the maximum length, which is less than or equal to 240 bytes. The attribute table contains the EID, SN, and RSN columns and the following indexes are created for the attribute table:

  ```
  SDSINST1.RSN <------A reverse index whose defined index keys are the EID
      and RSN columns.
  SDSINST1.SN<------A forward index whose defined index keys are the EID
      and SN columns.
  SDSINST1.SNI <------An update index whose defined index key is the EID column.
  ```

- Reorganize all the attribute tables that you want to use in searches. In most cases, you might want to reorganize to the forward index. For searches that begin with * (wildcard), reorganize to the reverse index.
- When an attribute length is defined with greater than 240 bytes, the attribute table contains four columns: EID, attribute, truncated attribute, and reversed truncated attribute. In this case, the forward index is created by using the EID and truncated attribute columns as index keys. For example, the CN attribute is defined with the maximum length, which is greater than 240 bytes. The attribute table contains the EID, CN, CN_T, and RCN_T columns and the following indexes are created for the attribute table:

  ```
  SDSINST1.RCN <------A reverse index whose defined index keys are the EID
      and RCN_T columns.
  SDSINST1.CN<------A forward index whose defined index keys are the EID
      and CN_T columns.
  SDSINST1.CNI <------An update index whose defined index key is the EID column.
  ```

An example output with reverse, forward, and update indexes:

```
Table: SDSINST1.SECUUID
SDSINST1 RSECUUID <— This is a reverse index
SDSINST1 SECUUIDI <— This is an update index
SDSINST1 SECUUID <— This is a forward index
```

Running `reorgchk` on a DB2 database

You can retrieve database statistics and use the values to determine whether tables, indexes, or both must be reorganized.

About this task

The DB2 database can become suboptimal and the server performance can degrade, when many updates are made to database. You must run the DB2 **reorgchk** to update the database statistics.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. Connect to the database.

   ```
   !db2 connect to sdsinst1
   ```

3. Run the **reorgchk** command.

   ```
   !db2 reorgchk update statistics on table all
   ```

4. To generate an output file with the database statistics, run the **reorgchk** command and redirect the output to a file.

   If you plan to run the **reorg** command, you must save the database statistics that are generated from the **reorgchk** command.

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

   A sample database statistics from the **reorgchk** command.

```
!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

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

SDSINST1   ACLPERM             2       0    1    1     138    0   -   100   ---

SDSINST1   ACLPROP             2       0    1    1      40    0   -   100   ---

SDSINST1   ALIASEDOBJECT       -       -    -    -       -    -   -     -   ---

SDSINST1   AUDIT               1       0    1    1      18    0   -   100   ---

SDSINST1   AUDITADD            1       0    1    1      18    0   -   100   ---

SDSINST1   AUDITBIND           1       0    1    1      18    0   -   100   ---

SDSINST1   AUDITDELETE         1       0    1    1      18    0   -   100   ---

SDSINST1   AUDITEXTOPEVENT     1       0    1    1      18    0   -   100   ---

SDSINST1   AUDITFAILEDOPONLY   1       0    1    1      18    0   -   100   ---

SDSINST1   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) *
(INDEXPAGESIZE-96) / (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) < 100

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

Table: SDSINST1.ACLPERM
SDSINST1 ACLPERM_INDEX             2     1     1      6     2  100  -   -  ---
Table: SDSINST1.ACLPROP
SDSINST1 ACLPROP_INDEX             2     1     1      6     2  100  -   -  ---
Table: SDSINST1.ALIASEDOBJECT
SDSINST1 ALIASEDOBJECT             -     -     -      -     -    -  -   -  ---
SDSINST1 ALIASEDOBJECTI            -     -     -      -     -    -  -   -  ---
SDSINST1 RALIASEDOBJECT            -     -     -      -     -    -  -   -  ---
Table: SDSINST1.AUDIT
SDSINST1 AUDITI                    1     1     1      4     1  100  -   -  ---
Table: SDSINST1.AUDITADD
SDSINST1 AUDITADDI                 1     1     1      4     1  100  -   -  ---
Table: SDSINST1.AUDITBIND
SDSINST1 AUDITBINDI                1     1     1      4     1  100  -   -  ---
Table: SDSINST1.AUDITDELETE
SDSINST1 AUDITDELETEI              1     1     1      4     1  100  -   -  ---
Table: SDSINST1.AUDITEXTOPEVENT
...
Table: SDSINST1.SN
SDSINST1  RSN                  25012   148     2     14 25012   99 90   0  ---
SDSINST1  SN                   25012   200     3     12 25012   99 61 119  --*
```

```
SDSINST1  SNI                  25012    84     2      4 25012   99 87   1  ---
...
Table: SDSINST1.TITLE
SDSINST1  TITLEI                   -     -     -      -     -    -  -   -  ---
Table: SDSINST1.UID
SDSINST1  RUID                 25013   243     3     17 25013    0 62  79  *--
SDSINST1  UID                  25013   273     3     17 25013  100 55  79  ---
SDSINST1  UIDI                 25013    84     2      4 25012  100 87   1  ---
Table: SDSINST1.UNIQUEMEMBER
SDSINST1  RUNIQUEMEMBER        10015   224     3     47 10015    1 60  44  *--
SDSINST1  UNIQUEMEMBER         10015   284     3     47 10015  100 47  44  -*-
SDSINST1  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.
```

Running `reorg` on a DB2 database

You can reorganize indexes on a table by rebuilding the index data into unfragmented, physically contiguous pages. Reorganizing indexes and tables reduce the time that is required to resolve a query and retrieve data that match the query filter.

Before you begin

You must retrieve organizational information about the database with the **`reorgchk`** command. For more information about retrieving DB2 database statistics, see "Running reorgchk on a DB2 database" on page 51.

About this task

You must use the database statistics to determine whether the tables and indexes require reorganizing. The time that is required to reorganize a table depends on the DB2 database size. Reorganizing a table takes more time than updating statistics. Therefore, it is best to update statistics first and confirm whether it improves database performance. Usually, data in LDAP is accessed by index, reorganizing tables is not as beneficial as reorganizing indexes.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. Connect to the database.

   ```
   !db2 connect to sdsinst1
   ```

3. To reorganize the indexes and table in the database, run the appropriate command that is based on your requirement:
   - To reorganize the tables with an asterisk in the last column, run the following command:

     ```
     !db2 reorg table table_name
     ```

     where, *table_name* is the name of the table to be reorganized. For example, `SDSINST1.LDAP_ENTRY`.
   - To reorganize a table to match the order of a particular index, run the following command:

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

where, *index_name* is the name of the index. For example, SDSINST1.SNI.

- To reorganize the indexes with an asterisk in the last column, run the following command:

```
!db2 reorg index index_name
```

DB2 selectivity

You can set the environment variables to influence the DB2 optimizer to make better choice about how to access data in the LDAP tables.

You can use the following environment variables to control the DB2 optimizer:

***LDAP_MAXCARD* = YES | ONCE | NO**

You can use the *LDAP_MAXCARD* environment variable to set the cardinality of the LDAP_DESC table. When you set this variable, a cardinality of 9E18 is assigned to the LDAP_DESC table. The cardinality value influences the data access sequence of the DB2 optimizer. DB2 resolves all attribute filters before it considers the LDAP_DESC table for query evaluation.

If the variable is set to YES, the cardinality statistic for the LDAP_DESC table is tuned to prevent expensive scans of large subtree data.The cardinality statistic is set at the server startup and periodically thereafter.

If the variable is set to ONCE, the cardinality is set during the server startup and not later when the server is running.

If the variable is set to NO or not set, the cardinality statistic is not set during the server startup.

***IBMSLAPD_USE_SELECTIVITY* = NO | YES**

If the *IBMSLAPD_USE_SELECTIVITY* variable is not set to any value or is set to NO, selectivity is not used to influence DB2 access sequence.

If *IBMSLAPD_USE_SELECTIVITY* is set to YES and *LDAP_MAXCARD* is not set to YES, selectivity is used to influence the data access sequence of DB2 during the subtree search on a large subtree.

Note: If *LDAP_MAXCARD* and *IBMSLAPD_USE_SELECTIVITY* are set to YES, the Directory Server generates a message and does not use selectivity.

You can improve the performance of subtree searches on search bases that are high in a directory tree by using SELECTIVITY in Structured Query Language (SQL). The inclusion of SELECTIVITY in SQL enables the DB2 optimizer in the formation of data access sequence to resolve the search requests. The data access sequence identifies which tables to access first during searches. Identifying the entries that are high in the tree (having many subentries) is based on DB2 statistics. If a subtree search is done by using one of these entries as the search base, the SELECTIVITY clause is added to the SQL query. When the SELECTIVITY clause is added, DB2 uses the search filter to narrow down the search results. DB2 narrows down the search results before it reads from the table that identifies the entries that are descendants of a base in a search.

To use SELECTIVITY, DB2_SELECTIVITY must be set to YES in the DB2 registry for the database instance. You must set DB2_SELECTIVITY in addition to the environment variables. You can set DB2_SELECTIVITY when you create a database instance.

Examples

Example 1:

To check the status of DB2_SELECTIVITY for a Directory Server instance, myinst1, run the following commands:

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

Example 2:

To set DB2_SELECTIVITY for the Directory Server instance, `myinst1`, run the following commands:

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

Example 3:

To set DB2_SELECTIVITY in the configuration file of the Directory Server instance, `myinst1`, run the following commands:

```
idsldapmodify -h host -p port -D adminDN -w adminPW
dn: cn=Front End, cn=configuration
changetype: modify
add: ibm-slapdSetEnv
ibm-slapdSetEnv: IBMSLAPD_USE_SELECTIVITY=YES
```

DB2 query optimization for subtree searches

If a Directory Server hierarchy is deep nested, the subtree search might take considerable time to return the results. To retrieve subtree search results faster, you must optimize the SQL queries that access the LDAP_DESC table and other tables.

To optimize SQL queries for improved search performance, the DB2 optimizer analyzes the distribution statistics and creates a data access plan for each search. You can use environment variables to influence the data access plan or the query access plan that the DB2 optimizer generates. You must run the **`idsrunstats`** command against the Directory Server instance.

Environment variables

To influence the data access plan, use the following variables:

LDAP_MAXCARD

Use the *LDAP_MAXCARD* environment variable to set the cardinality value.

If you set cardinality, the attribute indexes are resolved first and the LDAP_DESC index is considered towards end for query evaluation. The following search performance might be observed:

- Improvement in subtree search performance against large subtrees with many descendants.
- The disadvantage is that all subtree searches are treated as search against a large subtree. Therefore, you might observe a performance degradation if the subtree search is against a small subtree with fewer descendants.

IBMSLAPD_USE_SELECTIVITY

Use the *IBMSLAPD_USE_SELECTIVITY* variable to set DB2 selectivity.

If you set DB2 selectivity, the following performance might be observed:

- Improvement in subtree search performance against top 10 subtrees that contains the most descendants. For top 10 subtrees with most descendants, the attributes filters are resolved first and the LDAP_DESC table is considered towards end for query evaluation.
- If a subtree search is run against a subtree that is not in top 10 with most descendants, you might observe a performance degrade. For the subtrees that are not in the top 10 with most descendants, the LDAP_DESC index is resolved first followed by the attribute indexes.

Query access plan

The *LDAP_MAXCARD* and *IBMSLAPD_USE_SELECTIVITY* environment variables are ways to improve subtree search performance in specific cases. In certain subtree search scenarios, even if you set the environment variables you might not obtain the required performance improvements. Instead of using the environment variables, use the query access plan that the DB2 optimizer generates. IBM Security

Directory Suite provides an option to use the DB2 optimizer to improve the subtree search performance in all subtree search scenarios.

When you run a subtree search against a Directory Server, the following process occurs:

1. The LDAP search query is converted to a series of SQL queries that evaluate the `LDAP_DESC` index and attribute indexes.
2. The SQL query that contains the `LDAP_DESC` index is evaluated by using a parameterized value for an ancestor EID (`AEID`). The parameterized value for `AEID` is replaced with the evaluated `AEID` literal value for further processing of the SQL queries.
3. The DB2 optimizer uses the `AEID` literal value to identify whether the subtree is large or small. The DB2 optimizer uses the collected distribution statistics to decide whether to resolve the attribute indexes or the `LDAP_DESC` index first in its data access plan.
4. Depending on the type of subtree and filter, the DB2 optimizer creates an optimal data access plan for subtree searches against both large and small subtrees.

DB2 query optimization for one level searches

To retrieve one level search results faster, you must optimize the SQL queries that access the `LDAP_ENTRY` table and other tables.

For one level searches, entry IDs are resolved by using the `PEID` column of the `LDAP_ENTRY` table. The SQL query that contains the `LDAP_ENTRY` table is evaluated by using a parameterized value of `PEID`. If a Directory Server hierarchy is deep nested or large, the one level search might take considerable time when the parameterized value of `PEID` is used. You must run the **`idsrunstats`** command against the Directory Server instance to optimize the database and update DB2 statistics.

When you run a one level search against a Directory Server, the following process occurs:

- The parameterized value for `PEID` is replaced with the evaluated `PEID` literal value for further processing of the SQL queries.
- The DB2 optimizer uses the `PEID` literal value to determine the subset of data in the large table. The DB2 optimizer uses the collected distribution statistics to decide whether to resolve the attribute indexes or the `LDAP_DESC` index first in its data access plan.
- Depending on the type of subtree and filter, the DB2 optimizer creates an optimal data access plan for subtree searches against both large and small subtrees.

DB2 indexes

You can use DB2 indexes to improve the performance of search operations against a directory server if the requested attributes are indexed.

When you index attributes that are frequently searched, it takes less time to locate the requested data. DB2 indexes improve the performance of directory operations, such as start time, subtree search of an LDAP entry, and finding all subentries of an LDAP entry. It is beneficial from a performance standpoint to index all relevant attributes used in searches.

The Directory Server contains a set of attributes that are indexed by default. The attributes that are indexed by default can be found in the directory schema files. An attribute with the `EQUALITY` keyword in the schema files indicates that the attribute is indexed. Attributes can also be indexed by running the DB2 commands on the DB2 table that implement the attribute.

Use the following DB2 commands to verify whether a particular index is defined. In the example, the `seeAlso` attribute is searched to verify whether the attribute is indexed.

```
!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 the database.

If the commands do not return three entries, the index is not properly defined. The command must return the following results:

```
IndexSchema    Index Name    Unique Rule    Number of Columns
-------------  ------------  ------------   -----------------
SDSINST1        SEEALSOI       D               1
SDSINST1        SEEALSO        D               2
SDSINST1        RSEEALSO       D               2

3 record(s) selected.
```

It is important to run the **runstats** command on a table after you create an index. The **runstats** command provides statistical information to the DB2 optimizer that aids the optimizer to take decisions about optimizing searches on that table.

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

Creating an index for an attribute by using Web Administration Tool

You can create attribute indexes by using **Web Administration Tool** to locate and retrieve the frequently used attributes faster.

Procedure

1. Log in with the Directory Server administrator credentials to a Directory Server by using **Web Administration Tool**.
2. In the navigation pane, expand **Schema management** > **Manage attributes**.
3. Click **Edit attribute**.
4. Under **Indexing rules** on the **IBM extensions** tab, select the **Equality** check box.
5. To save the changes, click **Apply**.

   You can click **OK** to save the changes and to exit the page. You can click **Cancel** to exist the page without saving any changes.

What to do next

Run the **runstats** command against the Directory Server to update the DB2 statistics after you create an index.

```
idsrunstats -I instance_name
```

where, *instance_name* is the Directory Server instance name.

Creating an index for an attribute from the command line

You can create attribute indexes to locate and retrieve the frequently used attributes faster.

Procedure

1. Log in with the Directory Server instance owner credentials.
2. To create an index for the `seeAlso` attribute, run the following command:

   ```
   ldapmodify -p port -D adminDN -w adminPW -i filename
   ```

   where, *filename* contains the following entries:

   ```
   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 )
```

3. Run the **runstats** command against the Directory Server to update the DB2 statistics after you create an index.

   ```
   idsrunstats -I instance_name
   ```

   where, *instance_name* is the Directory Server instance name.

DB2 configuration parameters

You can improve Directory Server performance by tuning the DB2 configuration parameters for the database that is associated with the directory server.

To improve the Directory Server performance, you can set the DB2 configuration parameters such as DBHEAP and LOGFILSIZ. To update the parameters, run the following commands:

```
!db2 update database configuration for database_name \
using parm_name parm_value
!db2 force applications all
!db2stop
!db2start
```

where, the variable specifies:

- *database_name* is the name of your database.
- *parm_name* is the parameter to change.
- *parm_value* is the value to assign the parameter.

You can also use the **idsperftune** tool to set the DB2 configuration parameters. To update the DB2 configuration parameter with the values in the `perftune_input.conf` file, run the following command:

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

To retrieve DB2 configuration parameters and their values, you can use the DB2 commands. You can retrieve the DB2 parameter and their current values with the following command:

```
!db2 get database configuration for database_name
```

where, *database_name* is the name of your database. For example, the following output shows the default settings of a Directory Server instance, `sdsinst1`:

```
 !db2 get database configuration for sdsinst1 | egrep'HEAP|MAXLOCKS|MINCOMMIT'

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

You must not modify the default settings of the parameters; unless you determined a value to assign for a parameter. For example, if you set `MINCOMMIT` to a value other than 1, you might get poor performance results. You can change the parameter values by using the following commands:

```
!db2 update db cfg for sdsinst1 using parm_name parm_value
!db2 terminate
!db2 force applications all
```

where, *parm_name* is the name of the parameter. You can set a parameter that is generated from the `db2 get database configuration` command with a value. The *parm_value* variable indicates the new value to assign.

Incorrect settings for some database parameters can cause database failures. If database failure occurs, check the following files for DB2 error messages:

- *db2instance_owner_home_directory*/idsslapd-*instance_name*/logs/db2cli.log
- *db2instance_owner_home_directory*/sqllib/db2dump/db2diag.log

You can also run the following command to retrieve the DB2 configuration parameter values. The command shows the DB2 configuration parameters for the entire database instance.

```
!db2 get database manager configuration
```

Changes to DB2 configuration parameters do not take effect until you restart the database with the **db2stop** and **db2start** commands.

Note: If applications are connected to the database, you must run the **db2 force applications all** command before you run **db2stop**.

For tuning DB2 buffer pools, you can use the DB2 utility, `DB2 AUTOCONFIGURE`. This DB2 utility calculates and provides the values for the DB2 buffer pools, database configuration, and database manager configuration parameters. For example, if you specify `NONE` with `DB2 AUTOCONFIGURE` , the utility returns the values that are required to be modified in the configuration but does not apply them.

```
!db2 AUTOCONFIGURE USINGMEM_PERCENT 60 WORKLOAD_TYPE simpleNUM_STMTS 500
ADMIN_PRIORITY performanceIS_POPULATED YESNUM_LOCAL_APPS 20NUM_REMOTE_APPS 20
ISOLATION RRBP_RESIZEABLE YES APPLY NONE
```

The `DB2 AUTOCONFIGURE` utility with the `DB AND DBM` parameter returns and applies the changes to the buffer pool settings, database manager configuration, and the database configuration.

```
!db2 AUTOCONFIGURE USINGMEM_PERCENT 60WORKLOAD_TYPE simpleNUM_STMTS 500
ADMIN_PRIORITY performanceIS_POPULATED YESNUM_LOCAL_APPS 20NUM_REMOTE_APPS 20
ISOLATION RRBP_RESIZEABLE YES APPLY DB AND DBM
```

For more information, search `DB2 AUTOCONFIGURE` in http://www-01.ibm.com/support/knowledgecenter/SSEPGG/welcome.

For a list of DB2 parameters that affect database performance, see the DB2 documentation at http://www-01.ibm.com/support/knowledgecenter/SSEPGG/welcome.

Note: In a DB2 database analysis, if a parameter is configured insufficiently then the problem is logged in the `db2diag.log` file (diagnostic log). For example, if the DB2 buffer pools are too large then DB2 overrides the buffer pool settings and uses a minimal configuration. In such cases, no notice of the change in buffer pool sizes is logged except in the diagnostic log. It is important to view the log if you are experiencing poor performance. The `db2diag.log` file is in the `sqllib/db2dump` directory under the instance owner home directory. For example, for the `sdsinst1` instance on a Linux system you can find the `db2diag.log` file in the `/home/sdsinst1/sqllib/db2dump` directory.

Database backup and restore considerations

You can use the DB2 commands to back up and restore a DB2 database that is associated with a Directory Server instance. The DB2 command reduces the time that is required for the backup and restore operations, and provides the flexibility to specify the database file location.

You can use the **db2 backup** command that is provided by DB2 to back up the DB2 database that is associated with the Directory Server instance. You can use the **db2 restore** command can be used to distribute the database across multiple disks or to move the database to another directory. An important consideration for using the **db2 backup** and **db2 restore** commands is the preservation of DB2 configuration parameters and system statistics in the backed-up database. The restored database has the same performance optimizations as the backed-up database. When you use the **db2ldif**, **ldif2db**, or **bulkload** commands, the DB2 system statistics and performance optimization are not maintained.

It is important to keep in mind that when you restore over an existing database, any tuning that are done on the existing database is lost. Check all the DB2 configuration parameters after you complete the database restore. If the **db2 runstats** command was not run before the database was backed up, tune the DB2 system statistics after the restore operation. You can use the following DB2 commands to back up and restore a database:

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

where, `dsrdbm01` is the name of the directory server instance; and *directory_or_device* is the name of a directory or device to store the backup.

When you run the **db2 restore** command, you might observe file permission error. The reason for the error and steps to prevent the error are as follows:

- The DB2 instance owner might not have the required permissions to access the specified directory or file. To resolve the error, you must change the directory and file ownership to the DB2 instance owner. For example, enter the following command:

  ```
  chown dsrdbm01 file_or_device
  ```

- The backed-up database is distributed across multiple directories and those directories do not exist on the target system of the restore. Distributing the database across multiple directories can be accomplished with a redirected restore. To prevent this problem, create the same directories on the target system or run a redirected restore to specify the directories on the new system. When you create the directories, ensure that the owner of the directories is the DB2 instance owner.

Hardware tuning

To improve the update performance of a Directory Server, you must improve disk drive speed.

Consideration to improve disk speed

If a Directory Server contains millions of entries, it can become impossible to cache all of the entries in memory. Even if you are able to cache a directory with small number of entries, update operations require to access disk. The speed of disk operations is important. You can consider the following points to improve the 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
- Store log files and data on separate physical disk drives
- Use raw devices for storing the table space data

Tuning Directory Server features

You can configure and tune the features in IBM Security Directory Suite based on your requirements. You can also monitor the Directory Server performance and decide whether you want use certain Directory Server feature or not.

You can configure the following Directory Server features and monitor the Directory Server performance:

- Bulk loading (**`bulkload`**)
- Replication
- Performance monitoring
- Directory Server change log

Bulkload

You can use the **`bulkload`** utility to load data directly into the DB2 database from an LDIF file. The **`bulkload`** utility is a faster and better method to load large amount of data.

The **`bulkload`** utility parses the LDIF file, creates DB2 load files, and then loads the data directly into the DB2 database, bypassing the Directory Server.

The advantage to using the **`bulkload`** utility as compared to the **`ldif2db`** or **`ldapadd`** commands is the performance. The **`bulkload`** utility is many times faster than any other method of loading users. The **`bulkload`** utility might not be the right choice for every situation, such as to load small number of entries.

Consider the following points to determine whether to use the bulkload utility:

- When you load data by using the **`bulkload`** utility, the Directory Server might not be available for other LDAP-related operations. In a replication environment, ensure that one or more replicas or peer servers are available when data is loaded to a server with the bulkload utility.
- Updates that are made to a server by using the **`bulkload`** utility do not get automatically replicated in a replication environment. Each server in the replication environment must be updated by using the **`bulkload`** utility to synchronize all servers with the same data.
- In a replication environment, you can back up a server to other servers before you load data to the server by using the bulkload utility. For the backup and restore, you must re-create the replication agreements on the restored servers. After the restore, all servers contain the same database.

You can run the bulkload utility with the appropriate parameters based on the data to load into a Directory Server. You must not specify certain parameters as they are deprecated. The defaults for the following parameters are set for optimal performance and must be considered before you specify them.

`-A yes|no`
Specifies whether to process the ACL information that is contained in the LDIF file. The default value is **`yes`**. If you specify **no** with the parameter, it loads the default ACLs.

`-c | -C yes|no`
Specifies to skip index recreation. You can skip index recreation between the loads and postpone the index creation until the last **`idsbulkload`** operation. Run the last **`idsbulkload`** with the **`-c yes`** parameter.

`-e yes|no`
Drops indexes before the load operation.

Effects of using the -k parameter

You can use the **`-k`** parameter with **`idsbulkload`** to load data in smaller chunks. You can use this parameter on systems where memory is limited. With the **`-k`** parameter, **`idsbulkload`** parses and loads data in smaller increments.

Note: If you specify small chunk size to parse and load, it might take considerable time to load data.

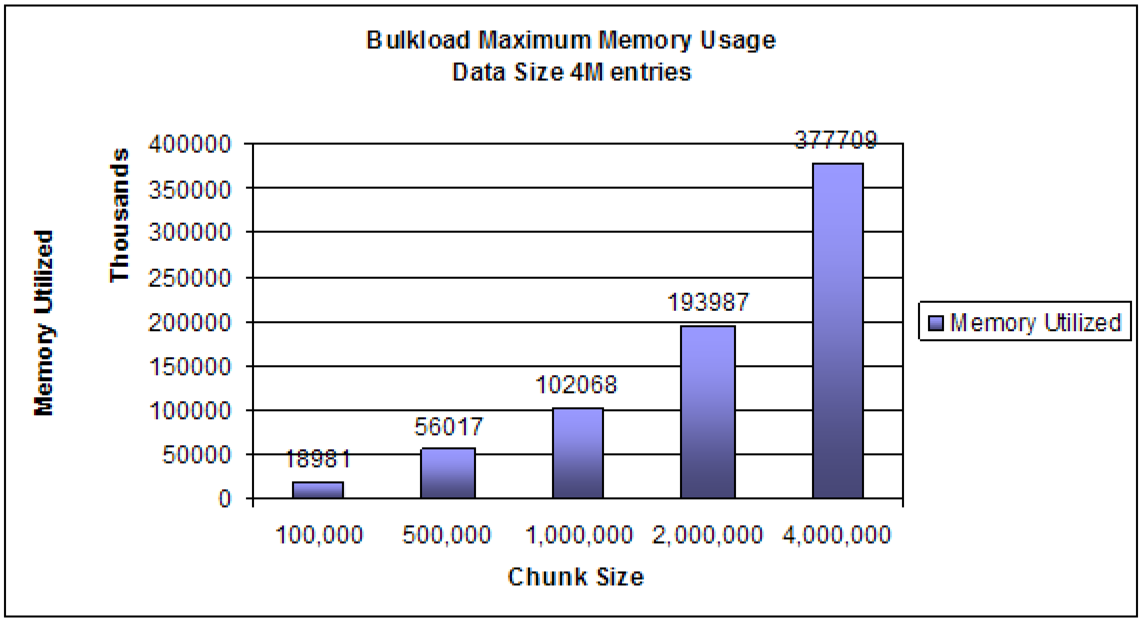

Figure 7. The relationship between data size and memory used

The graph illustrates the effects on memory usage. As the chunk size increases, the memory utilization increases.

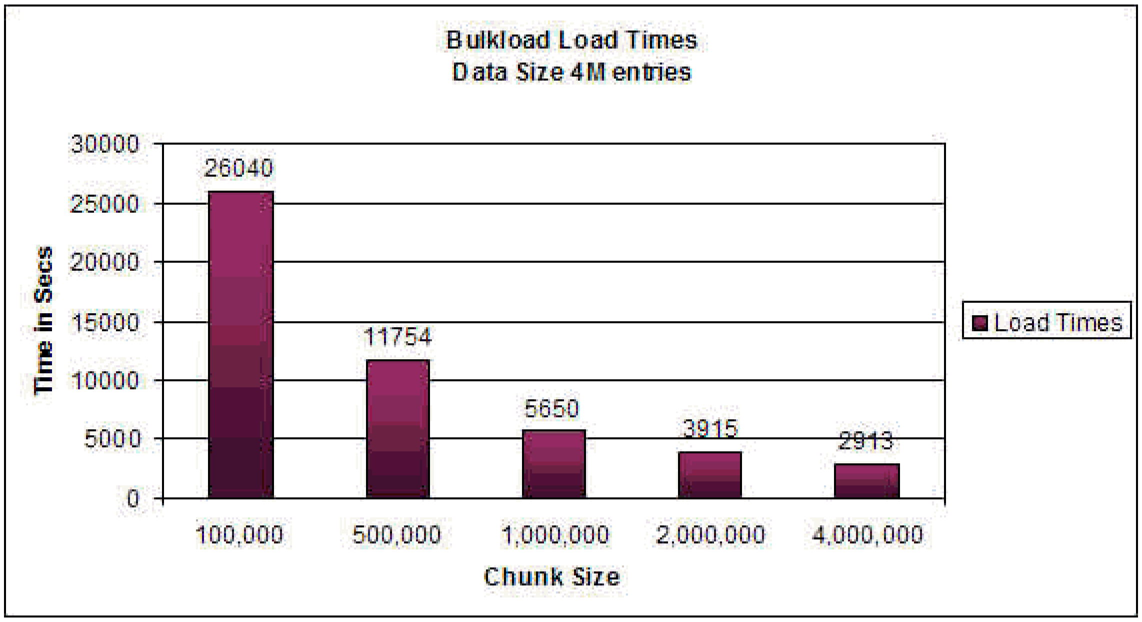

Figure 8. The relationship between data size and time to load data

The graph illustrates that as the chunk size increases, the load time decreases. The recommendation is to use chunk sizes of at least 1 million entries.

Replication

You can use the replication feature in a Directory Server to improve performance, availability, and reliability of the server. The replication process synchronizes data that are stored in multiple Directory Servers.

By using multi-threaded replication (asynchronous), you can activate replication to use multiple threads during the replication process. Multi-threaded replication improves the overall throughput of replication.

Multi-threaded replication is difficult to administer if servers or networks are not reliable. You can consider switching to multi-threaded replication for the following replication environments:

- A high update rate
- No previous versions of Directory Servers
- Common Advanced Encryption Standard (AES) salt and synchronization if the encryption is AES and passwords are updated often
- Small fanout (for example, eight 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 systems

If replication errors occur, the errors are logged in the log files. You must monitor the error logs and fix any replication issues. To search for the replication backlog for all agreements that are supplied by a server, run the following command:

```
ldapsearch -h sup_host -p port -D adminDN -w ? -s sub -b repl_context\
    objectclass=ibm-replicationagreement     ibm-replicationpendingchangecount ibm-
replicationstate
```

If the replication state is active and the pending count is growing, then the replication backlog might not decrease. For the replication backlog to decrease, the update rate must decrease or change the replication mode 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 sends the changes to all replica servers. If your application or users do not depend on immediate replication, then you must avoid setting the replication schedule at peak activity times. Scheduling replication at idle time can minimize the degrading the throughput on the master server.

You can tune the following configurable values to improve the replication performance:

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

Number of replication threads

You can tune the number of replication threads by setting the `ibm-replicaconsumerconnections` attribute with an appropriate value. The attribute value specifies the number of connections to use for each replication agreement.

If you increase the number of threads on both the supplier and consumer servers, the transaction rate also increases. In the following graph, a transaction is defined as a queued replication record that is sent to the supplier. In this example, the `ldap_modify` operation is used. The queued replication records (`ldap_modify`) are run with replication in the pending state. The replication state is then changed to *resume*, which starts the replication process.

Note: As the number of threads increases, the processor usage on both supplier and consumer systems increases. Tune the attribute as required based on an acceptable processor usage value and the required throughput.

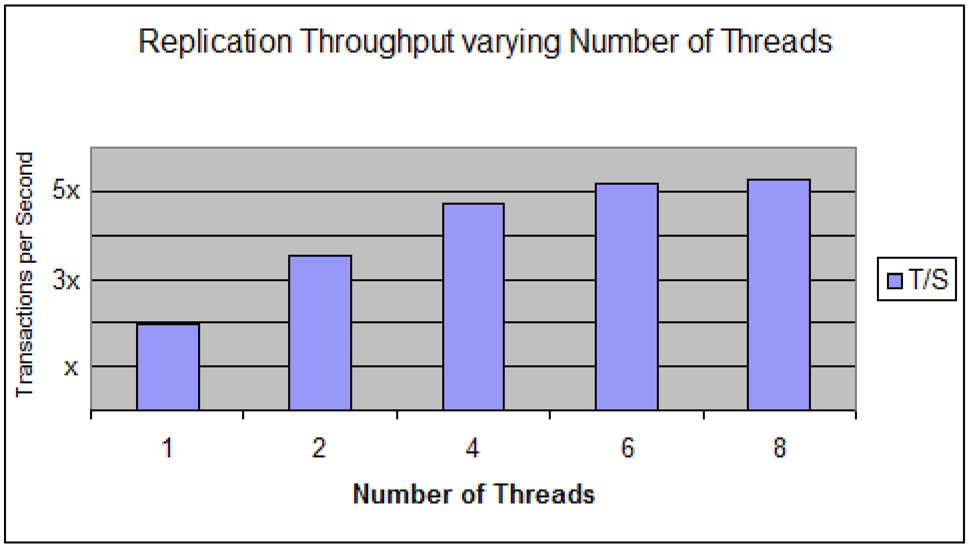

Figure 9. The relationship between the number of replication threads and transactions per second

If the throughput increases, the processor usage on both the supplier and consumer server increases. The processor cost per transaction on the consumer increases slightly when you add threads because there are more threads to manage now.

Replication context cache size

You can tune the memory size that you want to allocate to the replication context cache. The replication context cache stores replication update entries.

You can tune the replication context cache size by setting the `ibm-slapdReplContextCacheSize` attribute with an appropriate value in bytes. The default value is 100,000 bytes. The `ibm-slapdReplContextCacheSize` attribute is not a dynamic attribute.

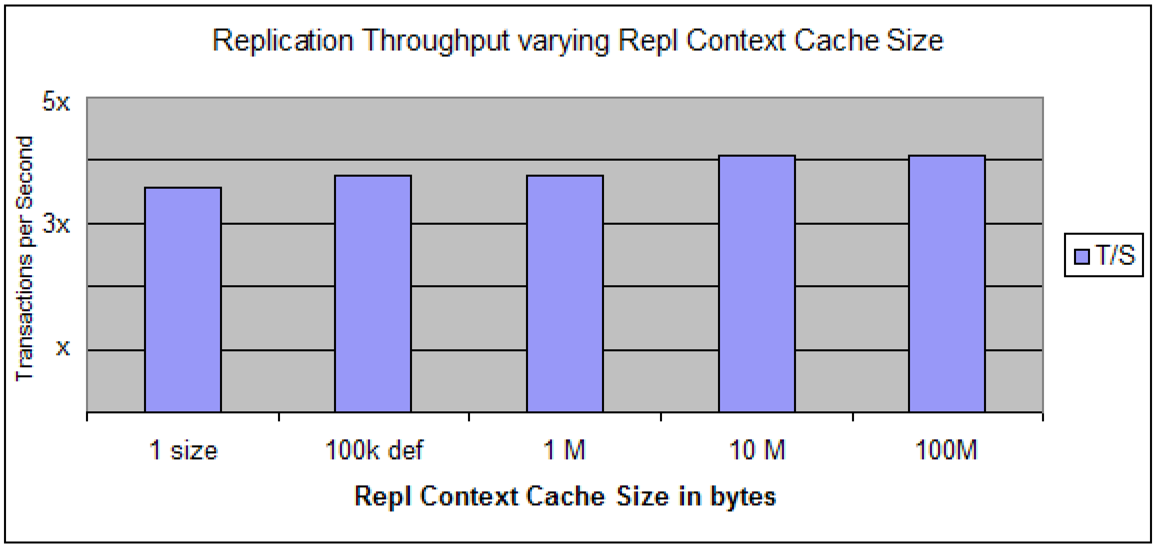

Figure 10. The relationship between the replication context cache size and transactions per second

Replication ready size limit

You can tune the replication ready size limit by setting the *IBMSLAPD_REPL_READY_SIZE_LIMIT* environment variable. The replication ready size limit controls the queue size of the replication operations from the list of replication update entries.

The default size limit is 10. There is one queue per connection to a replica. Updates that are related, such as modifications or subentries of an entry, are placed in the same queue. If the queue size exceeds the set size limit, the main replication thread waits for the queue size to be lesser than the limit. When the queue size becomes lesser that the limit, it reduces the processor usage by the main replication thread to determine the dependencies between the updates. In test environment, the size of the queue was varied from one entry up to 200 entries. Although, an increase in raw throughput was not evident but processor usage was reduced at certain settings of this variable. The following graph shows that the throughput normalized to 100% of processor usage. In the test environment, the absolute throughput did not change. Bigger the replication ready size limit value in the graph, lesser processor usage for transactions per second. In this graph, a transaction is defined as a queued replication record (`ldap_modify`) that is sent to the supplier.

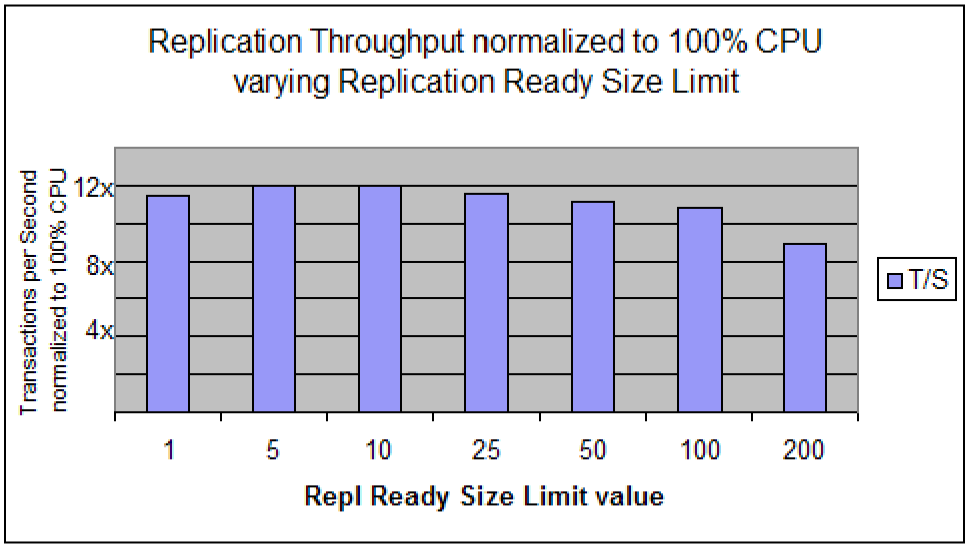

Figure 11. The relationship between the replication ready size limit and processor usage for transactions per second

Audit log

You can configure the audit log to audit operations that run against a Directory Server instance.

If you configure the audit feature, the performance of directory server instance can degrade based on the operations that are set for audit. If you do require auditing of operations against a directory server, it is advisable to turn off auditing of all operations. To disable the audit, you must ensure that the Directory Server instance is running. To check the status of the audit feature, run the following command:

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

where, `ibm-audit=false` indicates that the auditing is off. If the value of the `ibm-audit` attribute is `true`, set the value to `false`. To set the attribute to false, run the following command:

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

Proxy Server

You can use a Proxy Server in an LDAP environment where the data size exceeds the processing power and physical capacity of a single server. You can use a Proxy Server to distribute data across multiple Directory Servers.

Directory sizes that are greater than 40 million entries can be candidates for a distributed directory environment.

In a Proxy Server environment, the size of the connection pool can affect the throughput performance of a Proxy Server. You can configure the connection pool size for a Proxy Server. For best results, you must consider the following guidelines when you set the connection pool size:

- Configure more than one connection to the back-end server.
- Limit the connection pool size to the number of connections that the operating system can support. The connection pool is a static pool of connections that the Proxy Server sets up when the Proxy Server starts up. The operating system can impose a limit on the number of open file descriptors. The connection pool size must be less than the operating system limit.
- Ensure that the connection pool size is less than the number of database connections that are configured with the back-end server. You must keep a buffer for replication and change log.

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

Directory Server status

You can check the status of a Directory Server by searching the `cn=monitor` base.

You can retrieve the number of operations that are initiated and completed by the server from the time server is running. You can use the **`ldapsearch`** command to find the Directory Server status.

Search with the cn=monitor base

You can search the `cn=monitor` base to determine the Directory Server status, such as cache configuration status, operation counts, connection type counts, and status of service.

To retrieve the Directory Server status, run the **`ldapsearch`** command with the `cn=monitor` base. For example:

```
ldapsearch -h ldap_host -p port -s base -b cn=monitor objectclass=*
```

where, *ldap_host* is the host name or IP address of the LDAP host.

The `cn=monitor` search returns some of the following attributes of the Directory Server:

`CN=MONITOR`
The search base for the command.

`version=IBM Security Directory Suite, Version 8.0.1.x`
The version of IBM Security Directory Suite.

`totalconnections`
The total number of connections since the server was started.

`currentconnections`
The number of active connections.

`maxconnections`
The maximum number of active connections allowed.

`writewaiters`
The number of threads that send data back to the client.

`readwaiters`
The number of threads that read data from the client.

livethreads
The number of worker threads that are used by the server.

filter_cache_size
The maximum number of filters that are supported by the cache.

filter_cache_current
The number of filters currently in the cache.

filter_cache_hit
The number of filters that are 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 that are supported by 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 that indicates whether 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 of the server. The current time is in the following format:

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

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 following format:

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

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 that are requested since the server was started.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

unknownopscompleted
The number of unknown operations that are completed since the server was started. Unrecognized operations are rejected with a message is sent to the client, which might include 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 that are 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 associated with the server log counts:

slapderrorlog_messages
The number of server messages that are recorded since the server started or since a reset.

`slapdclierrors_messages`
The number of DB2 error messages that are recorded since the server was started or since a reset.

`auditlog_messages`
The number of audit messages that are recorded since the server was started or since a reset.

`auditlog_failedop_messages`
The number of failed operation messages that are recorded since the server was started or since a reset.

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. If the server is set to collect trace data, the value is TRUE, or else the value is FALSE.

`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 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 process run.

The following attribute is for alias dereference processing:

`bypass_deref_aliases`
The server runtime value that indicates whether alias processing can be bypassed. The value is 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 that is used by the directory attribute cache, in KB. This number includes the additional more memory that is used to manage the cache and is not charged to the individual attribute caches. If the total is larger than the sum of the memory that is used by all the individual attribute caches.

`cached_attribute_configured_size`
The maximum amount of memory, in KB, for use by the directory attribute cache.

`cached_attribute_hit`
The number of times the attribute is used in a filter and is processed by the attribute cache. The following value is generated in output:

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

cached_attribute_size

The amount of memory that is used for this attribute in the attribute cache. The following value in KB is generated in output:

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

cached_attribute_candidate_hit

A list of up to 10 most frequently used non-cached attributes that are used in a filter that can be processed by the directory attribute cache. The following value is generated in output:

```
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

You can use the following examples to calculate throughput and workload on the server from the output that is returned by the **ldapsearch** command.

Throughput example

You can calculate the throughput of the server by monitoring the Directory Server statistic called `opscompleted`. This statistics value indicates the number of operations that are completed since the LDAP server started.

To determine the throughput, find the values for the `opscompleted` attributes by issuing two **ldapsearch** commands. To monitor the performance statistics, run the first **ldapsearch** at time `t1` and the other at time `t2`. For example, `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)
```

A value of 3 is subtracted to account for the number of operations that are of the **ldapsearch** command.

Workload example

You can determine the server workload from the output values from the **ldapsearch** command with the `cn=monitor` base. For example, you can calculate the number of add operations that were completed in a certain amount of time.

To determine the number of add operations against the server in a time interval, find the values for the `addscompleted` attributes by issuing two **ldapsearch** commands. To monitor the performance statistics, run the first **ldapsearch** at time `t1` and the other at time `t2`. For example, `addscompleted` (`t1`) and `addscompleted` (`t2`).

The number of add operations that are completed by the server during the interval between `t1` and `t2` can be calculated as:

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

You can find the workload on the server for the other operations, such as `searchescompleted`, `bindscompleted`, `deletescompleted`, and `modifiescompleted`.

Search with the `cn=workers,cn=monitor` base

You can retrieve the information about the operations that the worker threads are running and when the threads started the operations. You must run the monitor search command with the administrator credentials.

To retrieve the search results for the `cn=workers,cn=monitor` base, run the following command:

```
ldapsearch -p port -D adminDN -w adminPWD -b "cn=workers,cn=monitor" -s base objectclass=*
```

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

The `cn=workers,cn=monitor` base search result is useful when a server performance is poor or not functioning as expected. You must use the information to get the insight into what the server is doing.

Attention: The `cn=workers,cn=monitor` search suspends all server activity until it is completed. For this reason, any application that runs a search with the `cn=workers,cn=monitor` base must issue a warning before the search. The response time for the search command increases as the number of server connections and active workers increase.

Search with the `cn=connections, cn=monitor` base

You can retrieve the server connections information of a server by searching the `cn=connections, cn=monitor` base. You must run the search command with the administrator credentials.

To retrieve server connection information, run the **`ldapsearch`** command with the `cn=connections, cn=monitor` base.

```
ldapsearch -h server -p port -D adminDN -w adminPWD  \
-b cn=connections,cn=monitor -s base objectclass=*
```

The search 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 the Directory Server is configured for secure connection, an SSL or a TLS indicator as appropriate is added on the connections.

For more information, see the Administering section of the IBM Security Directory Suite documentation.

Search with the `cn=changelog, cn=monitor` base

You can obtain the change log attribute cache information with a search against the `cn=changelog, cn=monitor` base. You must run the search command with the administrator credentials.

To retrieve the change log attribute cache information, run the **`ldapsearch`** command with the `cn=changelog, cn=monitor` base.

```
ldapsearch -h server -p port -D adminDN -w adminPWD  \
-b cn=changelog, cn=monitor -s base objectclass=*
```

The **`ldapsearch`** command with the `cn=changelog, cn=monitor` base returns the following attributes:

`cached_attribute_total_size`
The amount of memory that is used by the change log attribute cache, in KB. The memory that is used includes memory to manage the cache that is not charged to the individual attribute caches. This total is larger than the sum of the memory that is used by all the individual attribute caches.

`cached_attribute_configured_size`
The maximum amount of memory in KB that is set for use by the change log attribute cache

`cached_attribute_hit`

The number of times the attribute is used in a filter that might be processed by the change log attribute cache.

The value is in the following format:

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

`cached_attribute_size`

The amount of memory that is used for this attribute in the change log attribute cache.

The value is reported in KB and is in the following format:

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

`cached_attribute_candidate_hit`

A list of up to 10 most frequently used non-cached attributes that are used in a filter. The change log attribute cache might process these attributes if the attributes used in the filter are cached.

The value is in the following format:

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

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

Considerations to configure change log

You can configure the change log to record all updates to a Directory Server in a separate change log DB2 database. You can use the change log to query and track updated to the directory server instance.

When you configure the change log, it might degrade the Directory Server update performance. You must configure the change log only if needed. By default, the change log is disabled.

The change log database is different from database that holds the directory information tree (DIT) data.

You can check whether the change log is configured by searching for the `CN=CHANGELOG` suffix in a root DSE search result. If it exists, the change log is configured.

Capacity planning

Before you install IBM Security Directory Suite on a system, you must decide the hardware to use for setting up the existing capacity and scaling up when required.

The hardware resources that you must consider before you install and deploy IBM Security Directory Suite are:

- Hard disk
- Memory
- Processors

IBM Security Directory Suite performance on various hardware configurations might vary. It is important to understand the hardware configurations that can provide the system resources to the Directory Server for best performance.

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

IBM Security Directory Suite tuning

LDAP entry cache

DB2 tuning

DB2 buffer pools

- LDAP buffer pool

- `IBMDEFAULT` buffer pool

Optimization and organization (**`reorgchk`** and **`reorg`**)

DB2 configuration parameters

Backing up and restoring the database (backup and restore)

Splitting of database

The capacity planning information includes results that are observed from uploading data with the **`bulkload`** utility and running specific benchmarks.

The results that are provided are obtained through the following methods:

- Two types of Lightweight Directory Interchange Format (LDIF) files were used. The first type of LDIF file contains small entries with a flat tree structure. The other LDIF file contains larger entry sizes and a deeper tree structure.
- LDIF files with 100,000 and 1 million entries were created. The first type of LDIF file was used only for search operations. The second type was used for both searches and updates.

In each case, a Directory Server instance is created on the system. The LDIF file is loaded into the database and the recording were made. Operating system information and information that is related to the Directory Server instance is collected at intervals throughout the load and the performance test run.

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

The Directory Server performance was monitored by running various workloads on different hardware configurations.

Disk requirements

You must determine the space requirement for your directory server to store the database and the log files. You can use the space information to determine the required disk space.

You can store large amounts of LDIF data in a Directory Server. You must determine the hard disk capacity that is required to store the LDAP data. The required space can be determined based on the size of an average entry and the number of entries. You must also determine the processor speed, memory, and time that is required to load the data into the Directory Server from an LDIF file. The two most important factors you must consider for the disk requirements are:

- Time that is required to load the data into the Directory Server.
- Space that is required to store the data on the hard disk.

Time and space requirements

You must create an LDIF entry to determine the amount of space an entry require on a system. You can use this data to determine the amount of space that is required for an LDIF file with similar entries.

In the examples test runs, two types of LDIF files were used to gather data statistics. The first file contains a flat structure LDIF file with entries of small size. The second file contains a deeper tree with entries of large size. The entries in the LDIF file do not contain access control lists (ACLs) or groups that are associated with them.

An example entry of small size (approximately 415 bytes):

```
dn: cn=Joline Hickey, ou=Accounting, o=IBM.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
cn: Joline Hickey
sn: Hickey
description: Joline_Hickey
facsimileTelephoneNumber: +1 71 631-7308
l: Palo Alto
ou: Accounting
```

```
postalAddress: IBM.com$Accounting$Dept # 363$Room # 890
telephoneNumber: +1 408 995-7674
title: Supreme Accounting Mascot
userPassword: yekciHenil
seeAlso: cn=Joline
```

An example entry of large size (approximately 10,510 bytes):

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

The `mdsdata` attribute contains binary data. Binary data is not stored in attribute tables and can put a significant storage constraint on total disk space and on parsing and loading time.

Time information

You must determine the time that is required to load the data in a Directory Server from an LDIF file. You can use the load time to decide whether to load the file at one time or in smaller chunks.

The following graphs and tables show results of the **bulkload** utility when LDIF files with 100,000 entries and 1 million entries (for small and large entries) are loaded. The **bulkload** utility was run in two passes, first the parse phase and then the load phase. No tuning was done on the database settings before the **bulkload** utility was run. For the smaller entry, an index for the `seeAlso` attribute was added.

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

Time to bulkload LDIF files with 100,000 entries

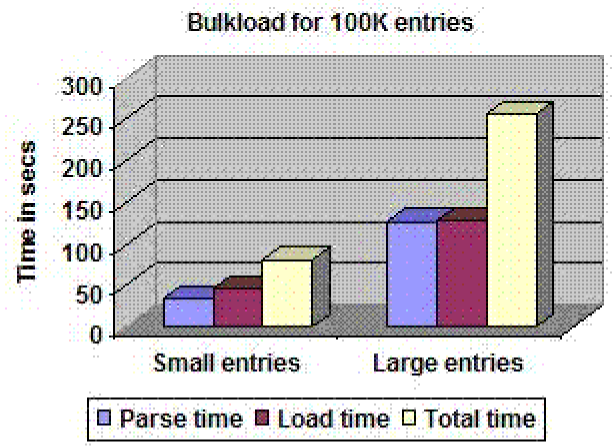

Table 2. Time to bulkload 100,000 entries in 2 stages		
Time in seconds	**Small entries**	**Large entries**
Parse time	35	127

Table 2. Time to bulkload 100,000 entries in 2 stages (continued)

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

Time to bulkload LDIF files with 1,000,000 entries

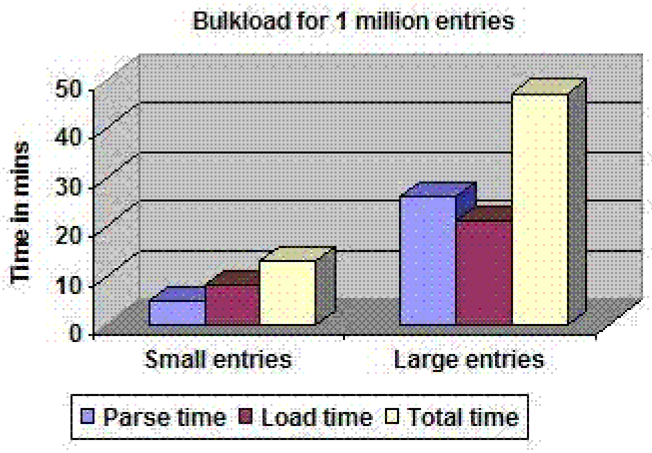

Table 3. Time to bulkload 1,000,000 entries in 2 stages

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

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

System 1
AIX 6.1 with 2 1499-MHz POWER 5 processors and 7967 MB of memory (for one logical partition (LPAR)

System 2
AIX 7.1 with 2 3,612 MHz POWER 5 processors and 16,268 MB of memory

Time to bulkload LDIF file with 1,000,000 small entries

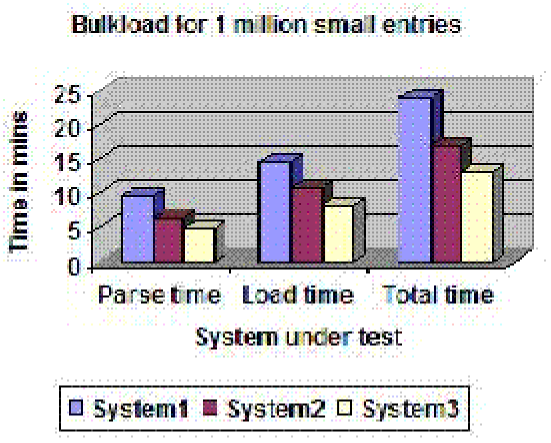

Table 4. Time to bulkload in 2 stages for 1,000,000 small entries on 2 different systems		
Time (minutes)	**System 1**	**System 2**
Parse time	5	5.12
Load time	8	5.41
Total time	13	10.53

On the AIX 6.1 system, the **bulkload** parse operation is run in same time as on AIX 7.1 system. On AIX 6.1 time to load the same amount of data took more time. The **bulkload** utility is a single threaded process, so improvements in time come from the processor speed, hard disk speed of the system, and the amount of memory used.

In general, the parse time increases linearly as the number of entries in the LDIF file increases. For example, the parse time for 100,000 small entries was 35 seconds and the parse time for 1,000,000 small entries was approximately 10 times. Load time does not increase linearly to the data set size.

During the parse phase, intermediate files are generated and these files are used in the load phase. If the input LDIF file is large, more data is written to the intermediate files. The increase in time is because of disk I/O and you can verify it based on the results that are obtained for the different LDIF files.
The average entry parse time per second is greater for the larger entry size LDIF. For an LDIF File with 100,000 entries of small size, an average of 2800 entries were parsed per second. For the larger sized entries, only 780 entries were parsed per second.

Space information

You must determine the space that is required on your system to load data to a Directory Server with the `bulkload` utility. You must ensure that the system has enough space to store the data that are generated during the different phases of bulkload.

On an AIX system, the space that is required for three different file system directories was calculated for each of the four LDIF files. The first is the space in the temporary directory that is used by the parse phase of the `bulkload` operation. You can refer the space temporary as the parse size. The second is the space that is required in the directory where the database is stored. You can refer the space that is required for the directory as the database size. The third is the space that is required to hold a backup of the database and you can refer it as the backup size. The charts that are displayed show the statistics for both the small and large entry files. The small and large-size entries files are created with 100,000, 1,000,000 entries.

Space that is required to bulkload an LDIF file with 100,000 entries

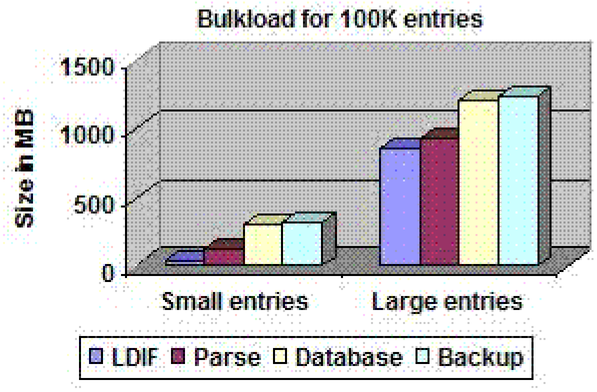

Table 5. Bulkload space for 100,000 entries		
Size in MB	**Small entries**	**Large entries**
LDIF file size in MB	41	859
Parse space in MB	139	927
Database space in MB	304	1213
Back up space in MB	320	1233

Space that is required to bulkload an LDIF file with 1,000,000 entries

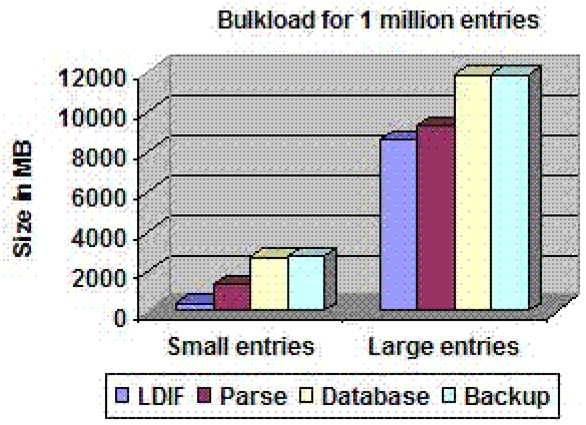

Table 6. Bulkload space for 1,000,000 entries		
Size in MB	**Small entries**	**Large entries**
LDIF size in MB	407	8597
Parse space in MB	1405	9294
Database space in MB	2779	11864
Back up space in MB	2793	11866

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

The space that is required to back up the database is roughly equivalent to the space needed for the database itself. The space that is required during the parse phase of the bulkload is about 3.5 times the size of LDIF file with the smaller entries. The space that is required for the parse phase of the bulkload is about 1.2 times the size of LDIF file for the larger entries.

Memory requirements

For a Directory Server to function properly, you must ensure that the system meets the minimum memory requirement.

The runtime memory requirements for the IBM Security Directory Suite and the requisite product, DB2, vary based on the following conditions:

- Number of users in the Directory Server
- Size of the Directory Server caches
- Size of DB2 buffer pools

When you tune a Directory Server for performance gain, you must tune the size of the LDAP caches and setting the DB2 buffer pools to AUTOMATIC. When you tune the memory size, you must consider the memory capacity of the system. Allocating a large amount for the caches might result in an increase in the overall memory requirement. Therefore, cache sizes must be allocated carefully. For more information, see "Directory Server tuning " on page 6 and "Tuning DB2 and LDAP caches" on page 30.

Processor requirements

To improve the Directory Server performance, you must ensure that the LDAP operations use the processor time efficiently.

You must consider the following guidelines to ensure that the processor is used at its optimum level:

- The required data is available in the caches results in low disk accesses. Accessing data from memory is faster than accessing from disk. If data is accessed from memory, it decreases the time that the processor must wait for input/output to occur.
- Enable simultaneous multithreading (SMT) on systems with hyper-threading capabilities. It increases the processing capability and the application throughput. For more information, see "Simultaneous multithreading" on page 79.
- Systems with fewer processors but greater processor frequency is more efficient than systems with more processors but lower processor frequency. For example, a system with two processors with 1200-MHz speed is faster than a system with four processors with 600-MHz speed.

Database splits on multiple disks

To improve Directory Server throughput for search and update operations, you can split the database across multiple disks.

You can split the database across two different disks instead of placing the database on one disk. The test result shows the improvement in the throughput when the database is split across multiple disks. Processor utilization is an important factor that is responsible for improvement in throughput.

Table 7. Effect on search throughput when database is split across 2 hard disks

Database	Throughput	IB	LB	CPU %	Idle %	Wait %	AVM (MB)	Memory (KB)
On 1 disk	260	95	74	23	68	8	191	389044
Split across 2 disks	416	95	74	46	37	16	179	378464

The update workload consists of search and modify operations on a directory with 1 million entries. The processor idle % is reduced to half when the database is split across two different disks. You can improve the throughput results if you use a high speed hard disk drive to save I/O time.

Table 8. Effect on update throughput when database is split across 2 hard disks

Database	Throughput	IB	LB	CPU %	Idle %	Wait %	AVM (MB)	Memory (KB)
On 1 disk	118	92	73	23	68	9	496	400736
Split across 2 disks	237	93	72	47	36	16	451	398960

Simultaneous multithreading

You can improve the Directory Server performance if you run the server on a system that can run multiple threads to process an instruction.

Simultaneous multithreading (SMT) is a processor design that combines hardware multithreading with superscalar processor technology. Simultaneous multithreading can use multiple threads to issue instructions each cycle. In certain hardware multithreaded architectures only a single hardware context, or thread, is active on any cycle. SMT supports all thread contexts to simultaneously compete and share processor resources. Unlike conventional superscalar processors, which suffer from a lack of per-thread instruction-level parallelism, simultaneous multithreading uses multiple threads to compensate for low single-thread instruction-level parallelism. Simultaneous multithreading use multiple threads to run different instructions in the same clock cycle by using the process units that the first thread left.

For simultaneous multithreading, the changes that are required to the basic processor architecture are:

- The ability to fetch instructions from multiple threads in a cycle.
- A larger register file to hold data from multiple threads.

The performance benefits of a system that can run simultaneous multithreading are:

- Higher instruction throughput
- Programs are faster for various workloads that include commercial databases, web servers, and scientific applications in both multi-programmed and parallel environments

An example workload description

To configure a Directory Server with optimal settings for performance improvements, you must identify the data type and workload of your LDAP environment.

To obtain the performance results for your Directory Server, the performance tests must be a mix of base search and mixed job. The mixed job can include the update, search, and compare operations. You can divide the performance test scenarios in two phases, a warm-up phase and a run phase. During the warm-up phase, the search operations primarily request entries that are not in the LDAP caches. Most of these requests require interaction with the DB2 database. Before you consider recording the test results, you must run all the queries a minimum of one time in the warm-up phase. In the run phase LDAP cache memory might contain all the requested entries if the caches are large enough to hold all entries. The warm-up phase and the run phase can consist of two distinctly different workloads.

In the run phase, configure the client threads to send search requests to the Directory Server from the predetermined scripts (clients). The scripts can include different kinds of operations, including base searches, update, and compare operations that return multiple entries for each request. You can configure the time for which the client threads must run through their scripts. You must measure the throughput on the server at an interval of 1 minute. A 10-minute interval is referred to as a run. You must not restart the Directory Server between the runs.

TCP/IP settings

You can verify and update the TCP/IP parameter values if you observe timeout failures when you access a Directory Server.

If an LDAP server is protected behind a firewall, socket connection requests might timeout and can result in intermittent authentication failures. The socket connection request failures are because of a mismatch between the connection timeout settings of firewall and the frequency of keep alive network packets of an operating system.

If socket connection failures occur, decrease the operating system network parameters that control the interval between the keep alive packets. The interval between two subsequent keep alive packers are also referred as the keep alive interval.

The parameters that control the keep alive frequency vary with each operating system. You must set the keep alive interval value lesser than the connection timeout value of the firewall. If you do not know the value of the firewall setting, set keep alive interval value to 2 minutes and verify.

The closed TCP/IP connections between the client and the LDAP server are cleaned at system-specified intervals. The LDAP server performance might degrade in environments where the connections are opened or closed at a high frequency.

System configurations

To install and configure IBM Security Directory Suite, you must use a system that meets the minimum hardware and software requirements.

To run performance tests and measure the test results, the following system configurations are used:

Client configurations

System with SLAMD server

One 2x3.2 GHz, 2048-MB memory, Intel PRO/100 VE

Red Hat Enterprise Linux (RHEL) 5

System with SLAMD clients

Five 2x3.2 GHz, 2048-MB memory, Intel PRO/100 VE

Red Hat Enterprise Linux (RHEL) 5

Server configurations

4-Way 1.6 GHz, System p Model IBM 9118-575 with 1 or 4 processors active, 31744-MB memory

AIX 6.1

IBM 10/100 Ethernet adapter

IBM Security Directory Suite

`AIXTHREAD_SCOPE=S`

`MALLOCTYPE=buckets`

`NODISCLAIM=true` (one way)

`RDBM_CACHE_SIZE=460000` (except where noted)

`RDBM_FCACHE_SIZE=75000` (except where noted)

`RDBM_CACHE_BYPASS_LIMIT=100` (except where noted)

Created indexes for the frequently accessed attributes

No ACLs are set. By default, any user can search and compare. The directory administrator can update.

Configurations for DB2

IBM DB2, Version 10.5.0.3

`maxlocks 100, sortheap 2500, dbheap 5000, ibmdefaultbp 20000 (4K pages), ldapbp 9800 (32K pages)`

```
logfilsiz 2048, logprimary 6
```

Miscellaneous configuration settings

Run all scripts for a minimum of one time to load caches.

Measurements were taken with 15 threads per client except where noted.

DB2 log files are not on the same disk as the containers.

Appendix A. An example workload description

To configure a Directory Server with optimal settings for performance improvements, you must identify the data type and workload of your LDAP environment.

To obtain the performance results for your Directory Server, the performance tests must be a mix of base search and mixed job. The mixed job can include the update, search, and compare operations. You can divide the performance test scenarios in two phases, a warm-up phase and a run phase. During the warm-up phase, the search operations primarily request entries that are not in the LDAP caches. Most of these requests require interaction with the DB2 database. Before you consider recording the test results, you must run all the queries a minimum of one time in the warm-up phase. In the run phase LDAP cache memory might contain all the requested entries if the caches are large enough to hold all entries. The warm-up phase and the run phase can consist of two distinctly different workloads.

In the run phase, configure the client threads to send search requests to the Directory Server from the predetermined scripts (clients). The scripts can include different kinds of operations, including base searches, update, and compare operations that return multiple entries for each request. You can configure the time for which the client threads must run through their scripts. You must measure the throughput on the server at an interval of 1 minute. A 10-minute interval is referred to as a run. You must not restart the Directory Server between the runs.

Appendix B. TCP/IP settings

You can verify and update the TCP/IP parameter values if you observe timeout failures when you access a Directory Server.

If an LDAP server is protected behind a firewall, socket connection requests might timeout and can result in intermittent authentication failures. The socket connection request failures are because of a mismatch between the connection timeout settings of firewall and the frequency of keep alive network packets of an operating system.

If socket connection failures occur, decrease the operating system network parameters that control the interval between the keep alive packets. The interval between two subsequent keep alive packers are also referred as the keep alive interval.

The parameters that control the keep alive frequency vary with each operating system. You must set the keep alive interval value lesser than the connection timeout value of the firewall. If you do not know the value of the firewall setting, set keep alive interval value to 2 minutes and verify.

The closed TCP/IP connections between the client and the LDAP server are cleaned at system-specified intervals. The LDAP server performance might degrade in environments where the connections are opened or closed at a high frequency.

Appendix C. System configurations

To install and configure IBM Security Directory Suite, you must use a system that meets the minimum hardware and software requirements.

To run performance tests and measure the test results, the following system configurations are used:

Client configurations

System with SLAMD server

One 2x3.2 GHz, 2048-MB memory, Intel PRO/100 VE

Red Hat Enterprise Linux (RHEL) 5

System with SLAMD clients

Five 2x3.2 GHz, 2048-MB memory, Intel PRO/100 VE

Red Hat Enterprise Linux (RHEL) 5

Server configurations

4-Way 1.6 GHz, System p Model IBM 9118-575 with 1 or 4 processors active, 31744-MB memory

AIX 6.1

IBM 10/100 Ethernet adapter

IBM Security Directory Suite

`AIXTHREAD_SCOPE=S`

`MALLOCTYPE=buckets`

`NODISCLAIM=true` (one way)

`RDBM_CACHE_SIZE=460000` (except where noted)

`RDBM_FCACHE_SIZE=75000` (except where noted)

`RDBM_CACHE_BYPASS_LIMIT=100` (except where noted)

Created indexes for the frequently accessed attributes

No ACLs are set. By default, any user can search and compare. The directory administrator can update.

Configurations for DB2

IBM DB2, Version 10.5.0.3

`maxlocks 100, sortheap 2500, dbheap 5000, ibmdefaultbp 20000 (4K pages), ldapbp 9800 (32K pages)`

`logfilsiz 2048, logprimary 6`

Miscellaneous configuration settings

Run all scripts for a minimum of one time to load caches.

Measurements were taken with 15 threads per client except where noted.

DB2 log files are not on the same disk as the containers.

Index

Notices

Trademarks

Terms and conditions for product documentation