Performance Best Practices Paper

for

IBM Tivoli Directory Integrator v7.1

Version 2.1

April, 2010

Table of Contents

1 Introduction

This White Paper provides IBM Tivoli Directory Integrator Performance tuning information, techniques and pointers. This White Paper collects the best practices, tuning guidelines and How-To's with IBM Tivoli Directory Integrator Performance in mind. Often, you can achieve significant improvements using only moderate changes to existing Configs. However, you can plan for, and design for, optimum performance.

This paper also documents several performance benchmarking test results to illustrate the impact of tuning IBM Tivoli Directory Integrator features. These benchmarking tests are carried out on a number of platforms, and test metrics provided are specific to the test environment and the specific test executed to measure the performance.

NOTE: Mileage (and your own benchmark test results) varies depending on the platform IBM Tivoli Directory Integrator is run on, as well as the responsiveness of the target systems IBM Tivoli Directory Integrator is working with. As such, results shown are illustrative of potential performance gains in the specified scenario.

2 Best practices

This section helps you ensure that you are getting the best IBM Tivoli Directory Integrator performance possible even before you do any tuning. Before you begin IBM Tivoli Directory Integrator performance tuning, it's essential to understand the right ways of making better IBM Tivoli Directory Integrator - based solutions. This section provides general guidelines to follow while you develop and deploy solutions.

Here are some basic practices that can make a significant difference in performance before you begin tuning.

2.1 Preparing the solution environment

Always ensure that the operating system on which IBM Tivoli Directory Integrator will be running is up-to-date with all the latest patches, fix-packs and maintenance patches applied to it. Here is a link to the IBM Tivoli Directory Integrator support page where you can search for content pertaining to your IBM Tivoli Directory Integrator version.

IBM Tivoli Directory Integrator is a cross-platform; Java based application dependent on the Java Virtual Machine (JVM) it runs in. Although IBM Tivoli Directory Integrator installs its own JVM for most platforms, others operating systems such as AIX and z/OS must provide a JVM for IBM Tivoli Directory Integrator to use.

Furthermore, the JVM is dependent upon operating system (OS) services, so your operating system must be running with the required patches and fix-packs. You can find the recommended patch level for all supported platforms in the Platform Support Matrix for your version.

Also ensure that third-party applications, data sources (for example, directory servers, database systems, and message queues) and services to which IBM Tivoli Directory Integrator sends and receives data are adequately patched.

As you would expect, the performance of IBM Tivoli Directory Integrator (and therefore resulting benchmark results) is dependent on the activity and performance of other applications. OS processes and other applications introduce CPU, memory, disk and network resource contention that may interfere with your measurements.

Before you begin to measure performance of your IBM Tivoli Directory Integrator - based solution, try to assess the behavior of other systems that may affect your results, for example, other applications that share resources (such as network links and data stores) with IBM Tivoli Directory Integrator.

2.2 Move EventHandlers to Connectors in Server Mode

With IBM Tivoli Directory Integrator version 6.1, almost all EventHandlers are transitioned to Connectors in Server mode or in Iterator mode. The IBM Tivoli Directory Integrator version 6.1.1 completes this transition, and future releases do not support EventHandlers. The best practice is therefore to use the corresponding Connector rather than an EventHandler. The ChangeLog EventHandlers are transitioned to Connectors in Iterator mode. For example, AdChangeLog EventHandler is transitioned to Active Directory Change Detection v2 Connector in Iterator mode. Other EventHandlers (like the HTTP Server EventHandler) have been replaced with Server mode Connector (for example, the HTTP Server Connector). These Iterator and Server mode Connectors provide considerably more functionality, flexibility and performance than the corresponding EventHandlers did – more reasons to transition your solutions as soon as possible.

When using Server mode connectors, you can also configure your AssemblyLine to use a pool of AssemblyLine flow sections, allowing your server solution to handle more traffic.

2.3 Solution Design Considerations

When using IBM Tivoli Directory Integrator to develop high performance integration services, it is essential to keep performance in mind when you design your solution.

Each IBM Tivoli Directory Integrator AssemblyLine is a Java thread. By dividing a data set up among multiple AssemblyLines running in parallel, you leverage the multi-tasking features of the platform, as well as those of connected systems.

2.4 Starting multiple AssemblyLines

In cases where your IBM Tivoli Directory Integrator solution is composed of multiple AssemblyLines and the solution requires repeated invocation of AssemblyLines, you should use the AssemblyLine Connector or AssemblyLine Function Component (FC) for this purpose. You can also launch AssemblyLines using script API calls provided by IBM Tivoli Directory Integrator, but you can achieve better performance by using AssemblyLine Connectors.

The AssemblyLine Connector and FC provide a number of advanced features, such as log sharing and AssemblyLine initialization parameters.

2.5 Tuning external data sources

IBM Tivoli Directory Integrator solution usually communicates with various third party systems, services, and data repositories. Tune these repositories according to product documentation and best practices to achieve optimal performance with your IBM Tivoli Directory Integrator solution.

For example, if an AssemblyLine uses a JDBC Connector to connect to a database table, then tuning the database (including measures like indexing the appropriate columns in the table) can improve the speed and throughput of your solution.

2.6 Deciding deployment topology

One important aspect of creating a IBM Tivoli Directory Integrator solution is deciding the deployment topology.

In most of the cases in which IBM Tivoli Directory Integrator acts a client that connects to various third party data sources, IBM Tivoli Directory Integrator does not have any requirement to reside on the same system where the data source is running. This allows flexibility in deciding where IBM Tivoli Directory Integrator can be deployed. Depending on the hardware configurations of the system, other running applications, and memory use, you must decide whether IBM Tivoli Directory Integrator should run on the same system or on some other system in the same network area. However, the cost of network latency or delay must also be considered with the solution deployment topology. For example, when deciding whether to deploy the LDAP Connector that connects to the LDAP server on the same system (that is running IBM Tivoli Directory Integrator) or to a remote server, consider the network latency or delay as a performance issue.

3 IBM Tivoli Directory Integrator Performance tuning tips

This section provides details about various tuning parameters that can improve the performance of IBM Tivoli Directory Integrator solutions, including their effect on execution, and with a few examples.

3.1 Overall tuning parameters

These parameters are not specific to a connector or component of IBM Tivoli Directory Integrator, but are applicable in general to IBM Tivoli Directory Integrator - based solution.

3.1.1 Global Connector Pooling

IBM Tivoli Directory Integrator contains a Connector library so that users can pre-configure connectors to be reused in AssemblyLines. IBM Tivoli Directory Integrator version 6.1 extends the Connector library functionality adding the ability to define a pool for a specific connector configuration. When a pool is defined for a connector in the Connector library, the server creates and supports a pool of instances of this connector and provides them to AssemblyLines as they are needed.

When an AssemblyLine uses a connector from a connector pool, it receives an instance of that connector that is already initialized. As a result, the AssemblyLine can therefore skip the initialization of the connector. When the AssemblyLine terminates it does not terminate the connector as usual, but simply return it to the connector pool.

Global connector pooling improves overall execution performance where AssemblyLines are running repeatedly, accessing the same target system. The following test demonstrates the benefit of global connector pooling with repeated AssemblyLine executions.

Test:

AssemblyLine without Global Connector Pooling

An AssemblyLine uses a JDBC Connector that connects to DB2 installed on a remote machine and tries to iterate an empty table. The AssemblyLine is executed 1000 times in succession.

AssemblyLine with Global Connector Pooling

The same AssemblyLine as above is used, but the JDBC Connector is enabled with the Connector Pooling feature. The Pooling parameters are: minimum pool of five, maximum of ten. The AssemblyLine is executed 1000 times in succession.

With the Connector Pooling feature enabled, we got approximately 65 times faster results.

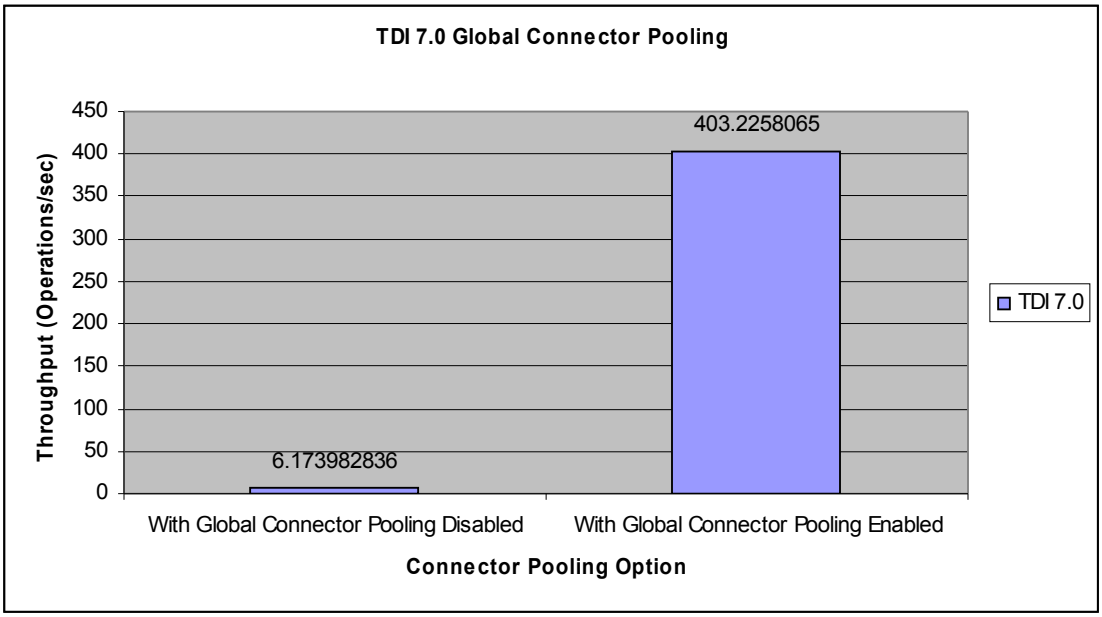

Figure: 3.1

3.1.2 AssemblyLine Pooling

The AssemblyLine Pooling feature allows you to build high performance solutions that do not incur a thread and connection cost for each processed event. The feature is only available if you have a Server Mode Connector in the Feeds section of your AssemblyLine.

The AssemblyLine Pool is a pool of identical threads that can be used to increase efficiency and where reuse strategy can be configured. When you set this parameter, you can achieve a significant performance gain for the AssemblyLines having Server Mode Connector.

3.1.3 Tuning the IBM Tivoli Directory Integrator System Store

The System Store serves as one of the important features of IBM Tivoli Directory Integrator that has been used by various IBM Tivoli Directory Integrator features such as Change Log connectors, Delta feature and persistent data storage. As a result, you must tune the System Store so as get better performance with IBM Tivoli Directory Integrator solutions. The following subsections describe various parameters for tuning the IBM Tivoli Directory Integrator System store.

3.1.3.1 Allow auto commit disabled

To provide more options to IBM Tivoli Directory Integrator developers, from IBM Tivoli Directory Integrator version 6.1, a new parameter, "Commit", has been added to the Connector Delta tab. The setting of this parameter now controls when the Delta Engine commits snapshots taken of incoming data to the System Store.

Adding the commit parameter was done in part to provide a performance boost in those scenarios where an AssemblyLine could safely be re-run in case of failure, and where remembering processed records were of little importance, and in part to reduce the chances of skipping processing of records that caused errors[1].

The Commit options are:

After every database operation	The default (and backwards compatible) value that causes snapshots to be committed to the System Store as they are made during the iteration.
On end of AL cycle	Commit when the AL has completed the current cycle. This is the recommended setting to use.
On Connector close	Commit is delayed until the AL is finished and connectors are closed. Although delaying Commit until the end of AL processing boosts performance, it can also result in situations where some changes have been successfully propagated, but the Delta Engine snapshots do not reflect this.
No autocommit	Committing of the snapshot must be handled manually through script with the commitDeltaState() method of the connector in Iterator Mode, such as: myIterator.commitDeltaState();

---

[1] In previous versions of IBM Tivoli Directory Integrator (before version 6.1), snapshots written to the Delta Store (a system feature that uses the System Store) during Delta Engine processing were committed immediately - that is, before the AssemblyLine got to process the entry. As a result, the Delta Engine would consider a changed entry as handled even though processing the AL Flow section had may have failed.

A secondary run would then skip the entry as if it were already processed.

The following tests show the results of setting different options with the new “Commit” parameter.

Test: A File System connector configured with a simple parser with Delta feature enabled iterates 10000 entries (2 attributes per entry). It performs add and update tests to Delta Store according to the setting of the “Commit” flag.
The overall CPU usage measured during the tests was 20-25%.

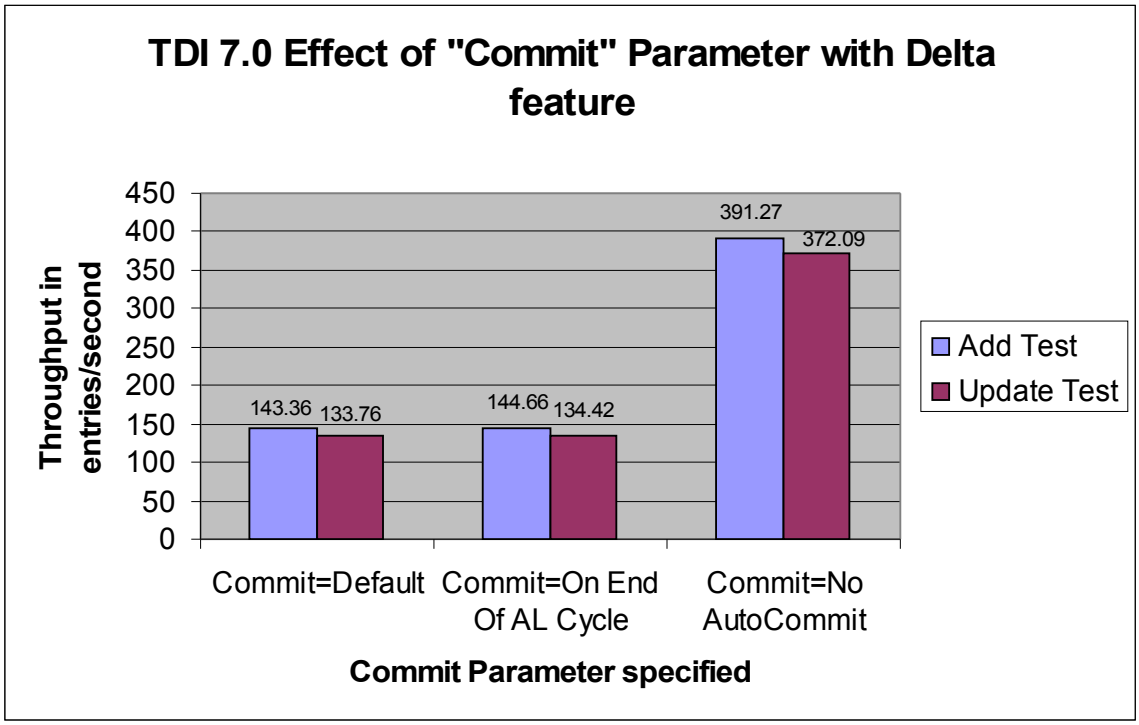

Figure: 3.2

3.1.3.2 Delta feature with “faster algorithm”

With IBM Tivoli Directory Integrator version 6.1.1, the Delta feature provides newer algorithms for performing updates with underlying system stores. This algorithm is specially designed for the data sources having few changes to be detected by the delta feature. A data source with a few changes has higher performance with the new faster algorithm (as long as there is enough memory to hold the internal hash table).

So whenever the Delta feature is applied with data sources containing large data entries and fewer changes, use the “Faster Algorithm” option with it.

The following tests show the impact of setting the “Faster Algorithm” option with the Delta feature.

Test: File System Connector with simple parser having the delta feature enabled. Initially iterate a file having 200,000 entries. Delete only ten entries from the source file. Run this test without enabling the "Faster Algorithm" option (that is, use the Old Algorithm). Re-run the same test, with "Faster Algorithm" option enabled. (Keep Read Deleted checkbox enable)

Performance of this feature also depends on mode which Derby database is operating. We can get better performance when the Derby database is operating in Embedded Mode.

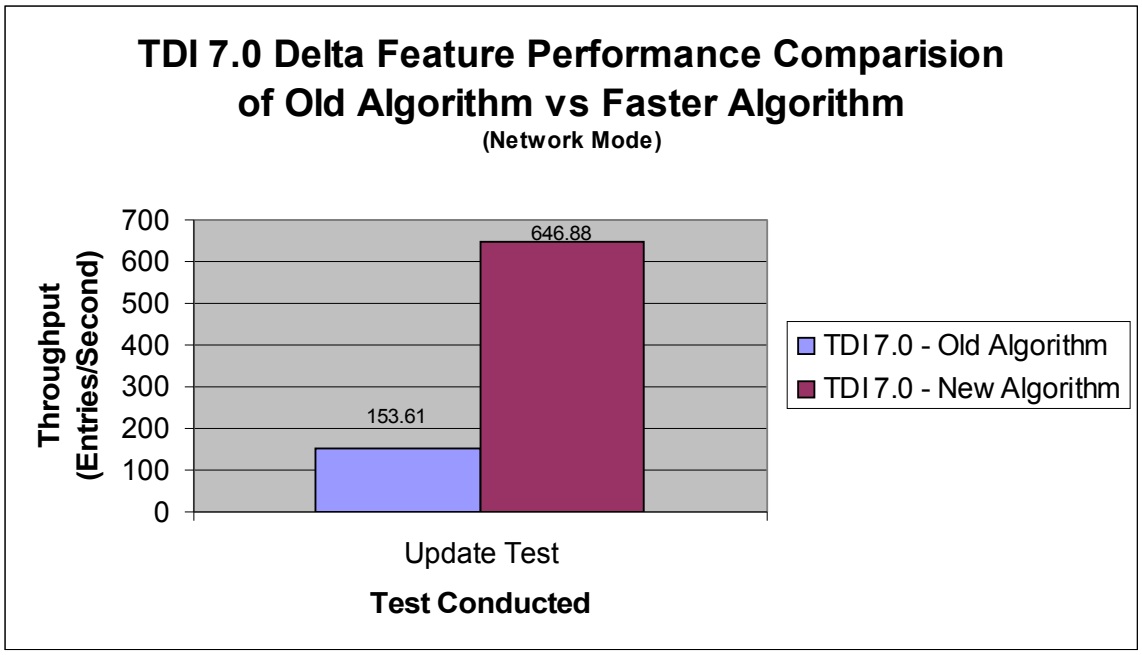

Figure: 3.3

As from IBM Tivoli Directory Integrator version 7.0, Derby database get started by default in Network Mode and hence in order to get optimum performance IBM Tivoli Directory Integrator solution developers need to enable Embedded Mode option manually.

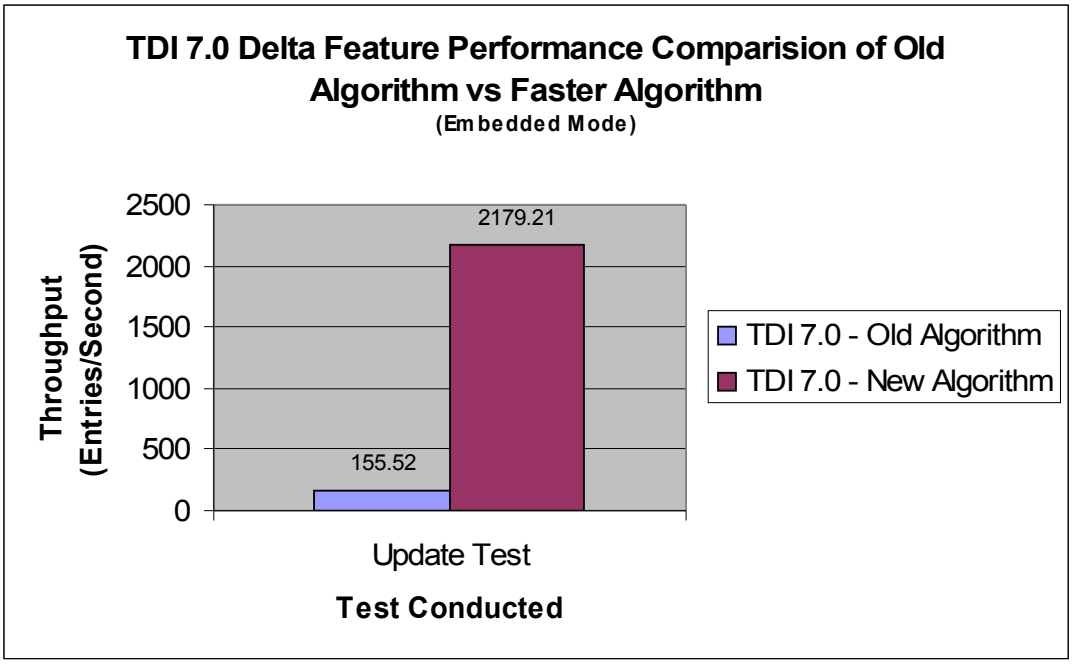

Figure: 3.4

3.1.3.3 Effect on system store with “Change Detection Mode”

From IBM Tivoli Directory Integrator v7.1, Introducing two new configuration parameters in Delta feature (Attribute List and Change Detection Mode), to leverage the ability of the Delta Engine to detect changes only in specific attributes instead of in all received attributes.

The performance of Delta store varies with the combination of these two parameters. When using the Delta Engine, sometimes the received entries contain attributes that you consider as not important and wish to ignore. In such cases, these attribute must not affect the result of the Delta computation, as when several Entries differentiate only by these attribute it leads to unnecessary updates of the Delta Store table. Therefore with the careful selection of key attributes, which really goes into Delta computation, results improved performance.

3.1.3.4 Effect on system store with “Row Locking”

To improve the reliability of Delta Store further, from IBM Tivoli Directory Integrator v7.1, A new configuration parameter “Row Locking” is introduced in the Delta tab for Iterator connectors and the Delta Function Component configuration. It allows you to set the transaction isolation level used by the connection established to the Delta Store database.

Each database server sets a default transaction isolation level; Some database servers may not support all transaction isolation levels, therefore please refer to the specific database documentation for accurate information about supported transaction isolation levels.

Note: Transaction isolation levels are maintained by the database server itself for every connection established to the database.

This parameter addresses the need for row locking in a Delta Store table when multiple DB Clients access the same data, by setting a Transaction Isolation Level. Setting a higher isolation level reduces the transaction anomalies known as 'dirty reads', 'repeatable reads' and 'phantom reads' by using row and table locks

This parameter has the following values

- READ_UNCOMMITTED
- READ_COMMITTED
- REPEATABLE_READ
- SERIALIZABLE

For more information on these configuration parameters, refer Deltas chapter in IBM Tivoli Directory Integrator *7.1* Users Guide.

Performance of Delta store varies with the combination of Row Locking parameter & Commit parameter. As we all know, performance comes at the cost of reliability. With the combination of Row Locking = Serializable and Commit = After every database operation, we can get highly reliable results

but have to pay penalty on its performance. On the other side, we can get faster results with the combination of Row Locking = Read UnCommitted and Commit = On Connector Close but the results will go unreliable if multiple DB Clients access the same data. Hence to get optimum performance with good reliability, it is advised to set Row Locking = Read Committed and Commit = On end of AL cycle.

3.1.3.5 Effect of IBM SolidDB as System store instead of Derby as default

IBM SolidDB is provided as a soft bundle with the IBM Tivoli Directory Integrator v7.1. IBM SolidDB is an in-memory database base and is perfect for both the Changelog connectors and the general usage of the delta engine when processing large data sets.

The beauty comes when we configure IBM SolidDB as IBM Tivoli Directory Integrator's system store. The below test scenario clearly indicates the performance boost gained by IBM Tivoli Directory Integrator when configured IBM SolidDB as its default system store in comparison to Derby in network mode (which is default to IBM Tivoli Directory Integrator from v7.0).

Test Scenario:
The objective of the below test is to analyze the throughput of IBM Tivoli Directory Integrator Delta store when configured with various backend databases.

A test config with two assemblylines is made, out of which first assemblyline will try to add keys in delta table and the second assemblyline will try to update the keys in delta table.

The test is carried out on following types of backend databases as IBM Tivoli Directory Integrator's System store, Derby database (in network mode as well as in embedded mode) and IBM SolidDB.

Test results are shown below.

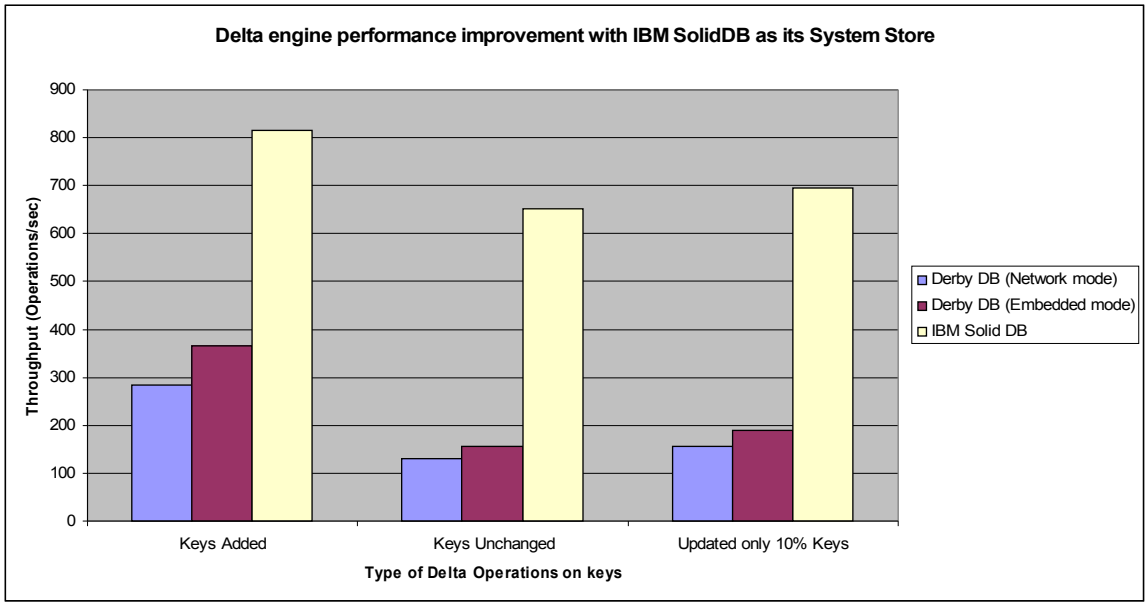

Figure: 3.5

From the above chart it is clear that, we can get approximately 3 times faster results with IBM SolidDB as default IBM Tivoli Directory Integrator's System store.

3.1.4 Log settings

There are several places within a IBM Tivoli Directory Integrator solution such as connectors, parsers, IBM Tivoli Directory Integrator server, AssemblyLine, and so on, where you can configure logging options. The detailed logging feature helps to debug execution flow as well as data flow throughout the execution. Use the logging option where applicable for debugging purposes only, otherwise it affects overall execution time. The following list describes several logging check points that can be considered before running a IBM Tivoli Directory Integrator - based solution for production use:

- **IBM Tivoli Directory Integrator Component Level Logging:** Make sure that the "Detailed Log" parameter with connectors, parsers, AssemblyLine, Functional Components is not enabled explicitly unless it is required. Detailed logging should be used for debugging purposes.

- **Logging feature for Config, AssemblyLine:** Ensure the settings with the "Logging" tab of the Config and AssemblyLine are not redirecting logging events to various log files, or to the console. This slows down the overall execution.

- **Logging by Solution Scripting:** Ensure that the scripting used within the solution at various places such as hooks and script components are logging or dumping minimum messages, and objects with the execution log.

- **Server level logging:** The file <*Install_dir*>/etc/log4j.properties controls the logging strategy for the server (ibmdisrv) when started from the command line. Make sure that the appropriate logging level is set with it. Logging level "DEBUG" slows down the server execution.

3.1.5 Server API settings:

IBM Tivoli Directory Integrator provides the ability to access configuration files remotely using the Server API that was introduced in the IBM Tivoli Directory Integrator version 6.0. The following parameters from global.properties/solution.properties for Server API can be ensured before running IBM Tivoli Directory Integrator - based solution:

- **Accessing IBM Tivoli Directory Integrator Configurations remotely:** The parameter "api.remote.on" enables the Remote Method Invocation (RMI) layer with IBM Tivoli Directory Integrator servers, enabling users to access configuration files remotely using RMI clients. The parameter "api.jmx.on" allows JMX based clients to access configuration files remotely. Ensure that these parameters are set in accordance with the requirement for accessing configuration files remotely.

- **Tombstone manager settings:** Tombstone Manager allows keeping track of configurations or AssemblyLines that have terminated. This feature creates tombstones for each AssemblyLine and configuration as they terminate. These tombstones contain exit status and other information that later can be requested through the Server API. To achieve this, the Tombstone Manager uses IBM Tivoli Directory Integrator System Store for data persistence. Enabling this feature can slow down overall execution of a IBM Tivoli Directory Integrator - based solution. Ensure that tombstone settings are configured at appropriate levels and cleanup of older tombstones is done properly. See "com.ibm.di.tm.on" for tombstone manager settings.

3.1.6 Setting -Xmx and -Xms options

While processing large sets of data within IBM Tivoli Directory Integrator, especially when using memory-based components like the XML Parser, you may run out of memory in the Java Virtual Machine even though you have more memory available on your computer system. When you run out of memory in the JVM on your computer system under these circumstances, it may help to increase the heap size in IBM Tivoli Directory Integrator.

To provide a value for these parameters, edit `ibmditk/ibmdisrv` to set the "`-Xmx`" and "`-Xms`" JVM options.

"-Xms" is the initial heap size in bytes and "-Xmx" is the maximum heap size in bytes. You can set these values according to your needs.

In general, the IBM Tivoli Directory Integrator 's JVM does not use all the available memory. You can increase the memory available to the IBM Tivoli Directory Integrator JVM by setting these values.

3.1.7 Connector initialization parameters

This section describes some of the tuning guidelines for initializing and using connectors within single or multiple AssemblyLines:

1. **Flexible connector initialization for big AssemblyLines:** Starting with IBM Tivoli Directory Integrator version 6.1, a configurable parameter, “Initialize”, is newly introduced with the connector configuration panel. Prior to IBM Tivoli Directory Integrator version 6.1, all Connectors in an AssemblyLine were initialized during the AL initialization phase. This feature allows users to decide when connector initialization should happen during an execution of its AssemblyLine. Configuring this parameter to an appropriate value can impact overall AssemblyLine execution. For example, setting the initialization parameter value to “Only when used” means that a complex AL only initializes the connectors that are actually used.

2. **Connector reuse inside single AssemblyLine:** In case of a single AssemblyLine that requires the same connector (connecting to same data source) to be used multiple times within the same AssemblyLine, the suggested approach is to reuse the existing connector.

3. **Connector reuse across multiple AssemblyLines:** If an IBM Tivoli Directory Integrator solution consists of multiple AssemblyLines, and more than one AssemblyLine is using the same connector to the same data source, then the suggested approach is to reuse a single connector within multiple AssemblyLines. This reduces the cost of connector initialization when one AssemblyLine is invoked repeatedly.

3.1.8 Skip Lookup Checkbox in Update/Delete Modes

To boost the Connector’s performance further, from IBM Tivoli Directory Integrator version 7.0 a new option is introduced for the connectors to skip the initial lookup when in Update/Delete mode and directly do the actual modification. Skip Lookup is a parameter and is supported for only certain connectors like JDBC Connector, LDAP Connector, JNDI Connector, TAM Connector, TIM DSMLv2 Connector, SAP ABAP Application Server Business Object Repository Connector, and SAP ABAP Application Server User Registry Connector.

This option is provided as a checkbox in the corresponding Connector’s User Interface. It appears only in Update/Delete mode next to the “Compute Changes” checkbox. When it is checked the behavior is as follows:

- **Update Mode** - With this option turned on only search criteria is build and directly Modify is called. This differs from the default connector behavior in Update mode because if no entry is found with the criteria, it is not added as a new one.
- **Delete Mode** – With this option turned on, only search criteria is built and directly Delete is called.

Note: When the option is checked,

- No “Before/After Lookup” hooks and “On Multiple Entries” hook are invoked.
- Entry on which operation is going to be done should exist in the data source.
- This option should be used only user wants to update/delete the entries without doing lookup ahead.
- Proper care should be taken with this option while operating on data sources like LDAP server else performance will degrade (Test Scenario 2).

Test Scenario 1:

A test has been carried out to test the performance increase with skip lookup option enabled for JDBC Connector on Machine A where IBM Tivoli Directory Integrator version 7.0 is installed & on Machine B where DB2 is configured with a test table containing 100000 records.

IBM Tivoli Directory Integrator config on machine A, iterates through a text file, containing input data for doing modifications in DB2, & updates corresponding records in the test table on Machine B. Capture the average elapsed time taken for completing the Assembly line.

Similarly, capture the average elapsed time taken for updating 100000 records from Machine A using IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5.

The chart below summarizes the test results for JDBC connector in update mode where 100000 modifications made from IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5 & from IBM Tivoli Directory Integrator version 7.0 with Skip Lookup option enabled.

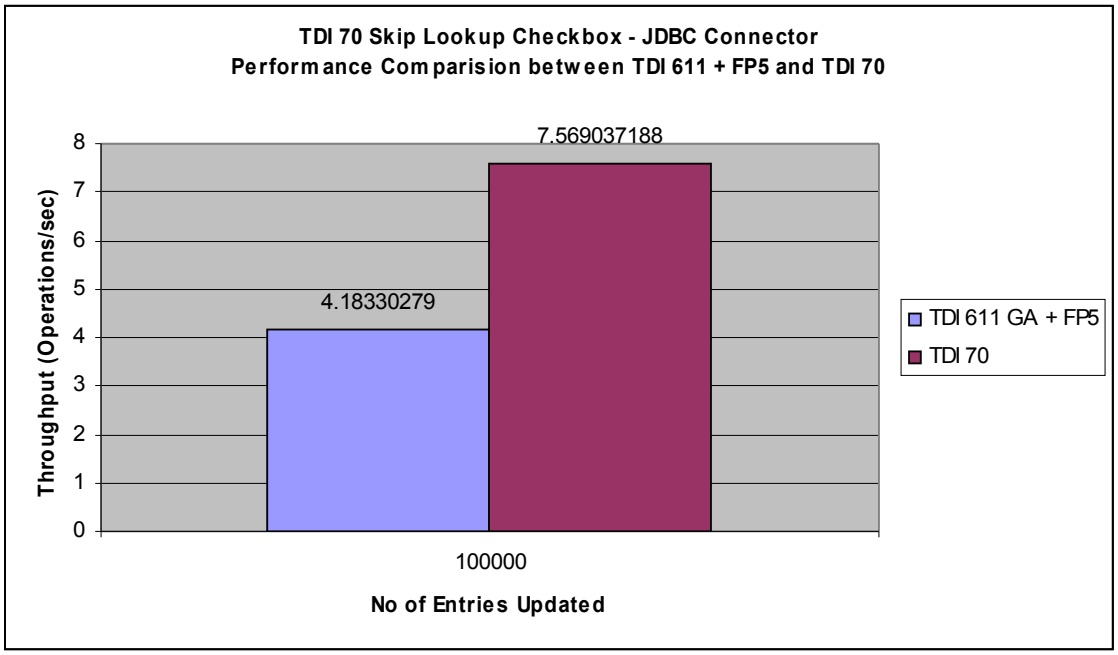

Figure: 3.6

From the above chart it is clear that IBM Tivoli Directory Integrator version 7.0, with Skip Lookup option enabled, has shown 44.7% increase in performance from IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5. In other words we got approximately 1.8 times faster results with skip lookup option enabled in IBM Tivoli Directory Integrator version 7.0.

When the similar scenario is provided to LDAP connector, the test results are as follows:

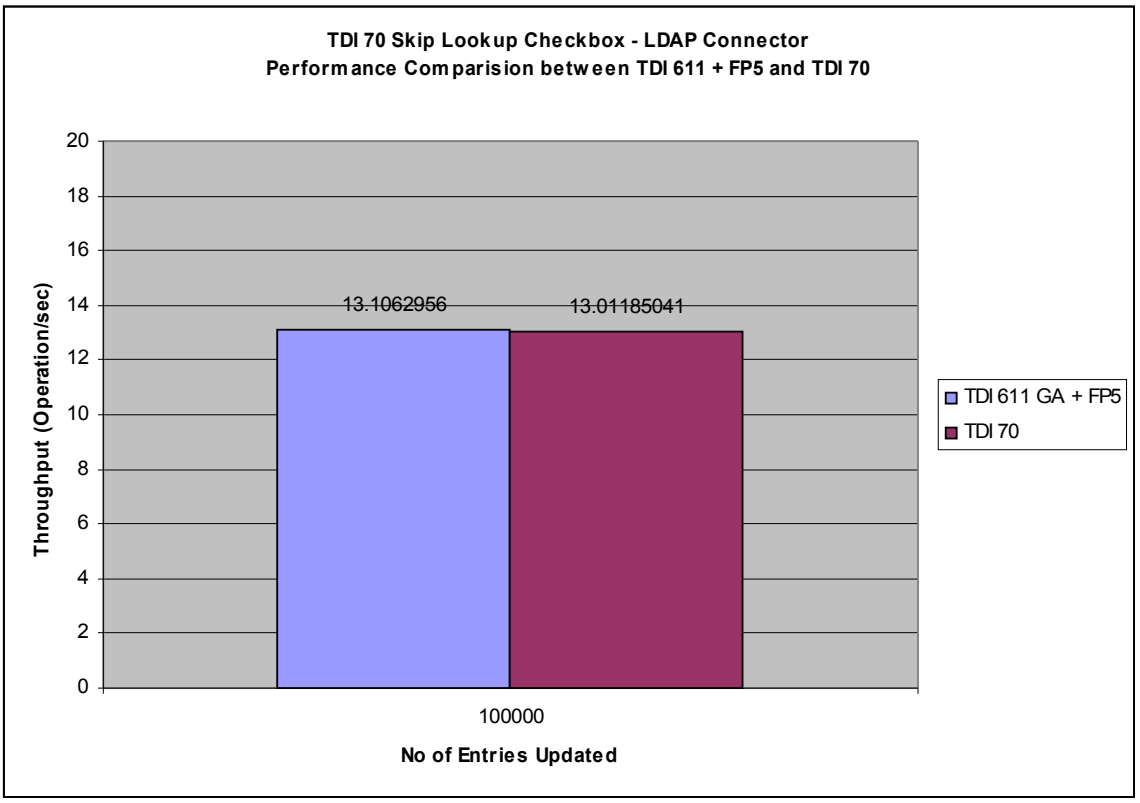

Figure: 3.7

Both IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5 & IBM Tivoli Directory Integrator version 7.0 shown almost equal numbers, the minor discrepancy might be because of side effects of other system processes running on the test machine during test. From the above chart, it is clear that in IBM Tivoli Directory Integrator version 7.0 with skip lookup option enabled has neither improvement nor degradation in performance in comparison to IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5. This is because; the cost of lookup is almost negligible in LDAP Server in contrast to the cost of lookup in JDBC data sources.

Test Scenario 2:

A test has been carried out to test the performance increase with skip lookup option enabled for LDAP Connector on Machine A where IBM Tivoli Directory Integrator version 7.0 is installed & on Machine B where LDAP server is configured and having 100000 entries.

IBM Tivoli Directory Integrator config on machine A, iterates through a text file, containing input data for doing modifications on LDAP server **(70% of input data is repeated)** , and updates corresponding entries on LDAP server on Machine B. Capture the average elapsed time taken for completing the Assembly line.

Similarly, capture the average elapsed time taken for updating 100000 entries from Machine A using IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5.

The chart below summarizes the test results for LDAP connector in update mode where only 30% modifications made by IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5 whereas IBM Tivoli Directory Integrator version 7.0 (with Skip Lookup option enabled) made 100% modifications.

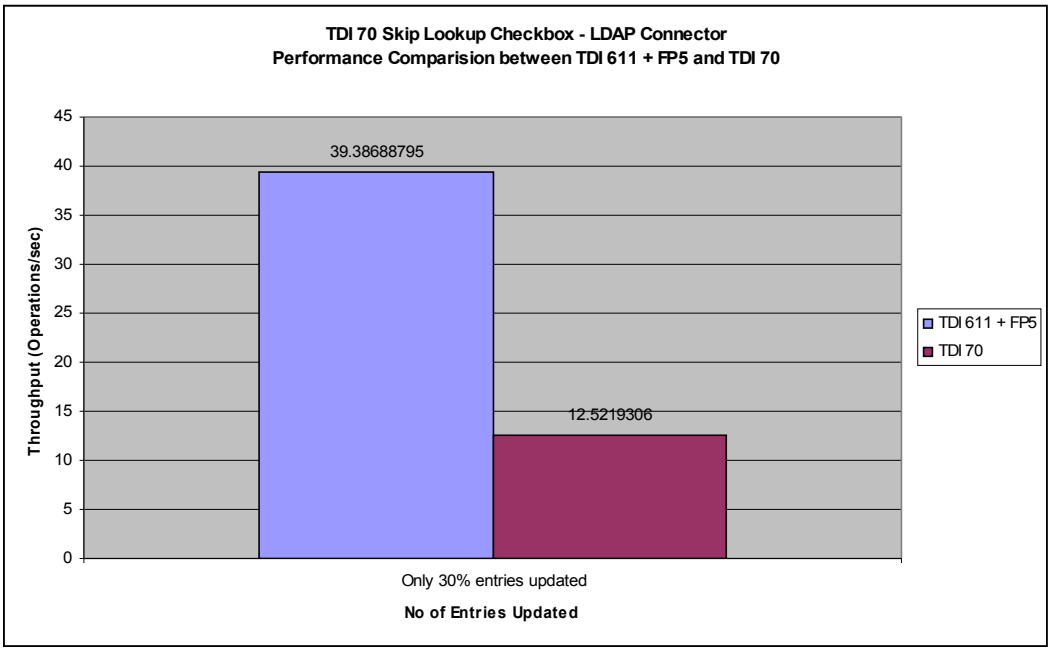

Figure: 3.8

From the above chart, we can see degradation in performance with IBM Tivoli Directory Integrator version 7.0 from IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5. This is because cost of write operation is more than cost of lookup operation in case of LDAP servers. In the given scenario, we have 70% of input data is repeated & hence IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5 skipped all those repeated entries by doing lookup ahead whereas IBM Tivoli Directory Integrator version 7.0 with skip lookup option enabled, updated all the entries irrespective of repetitiveness in input data.

When the similar scenario is provided to JDBC connector, the test results are as follows:

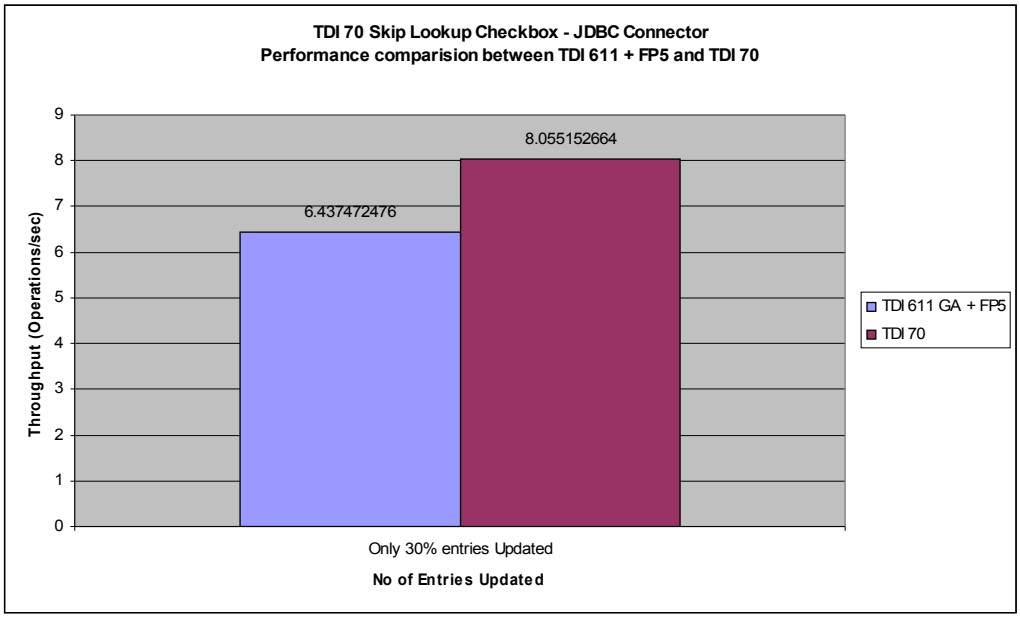

Figure: 3.9

From the above chart it is clear that IBM Tivoli Directory Integrator version 7.0, with Skip Lookup option enabled, has shown 20.08% increase in performance from IBM Tivoli Directory Integrator version 6.1.1 + Fix

Pack 5. In the given scenario, IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5 updated only 30% of input data & Skipped 70% of repeated entries by doing lookup ahead before updating them whereas IBM Tivoli Directory Integrator version 7.0 with skip lookup enabled updated 100% of entries, irrespective of repetitiveness in input data, but with 20.08% improvement in performance in comparison to IBM Tivoli Directory Integrator version 6.1.1 + Fix Pack 5.

3.2 Connector-specific tuning

This section describes tuning tips for some of the IBM Tivoli Directory Integrator components such as connectors and parsers that are most commonly used.

1. **Tuning an LDAP Connector configuration:** This section describes some of the parameters with the LDAP Connector that can be set properly to achieve a performance gain during connector execution.

    • Search Filter: Make sure that the LDAP Connector is configured with a precise search filter when needed. Specifying a very generic search filter such as “objectclass=*” can cause retrieval of all the entries from the directory server, thereby slowing down the execution of the LDAP connector.

    • Referrals: Decide upon whether you want referrals to be chased by the LDAP connector. If not needed, set the Referral value to “Ignore”.

    • Page Size: LDAP servers such as Active Directory support the Paged Search extension. The Paged Search extension enables you to retrieve a page (the number of objects to return at a time), and this is the preferred way to handle big return sets. You can always test whether a server supports the Paged Search by clicking to the right of the Page Size parameter in the LDAP Connector Configuration tab.

2. **Tuning the JDBC Connector configuration:** This section describes some of the JDBC connection parameters, that, when set properly, can achieve a performance gain during its execution.

    • Commit Parameter: Make sure that the “Commit” parameter from the JDBC Connector configuration is configured with proper options as required. The settings on the “Commit” determine how often to perform the “Commit” operation on the target database.

    • Prepared Statements: The JDBC Connector uses PreparedStatement to efficiently execute an SQL statement on a connected RDBMS server. However, there maybe cases when the JDBC driver may not support PreparedStatements. As a fall back mechanism this parameter is added to the configuration of the JDBC Connector.

    Note: From IBM Tivoli Directory Integrator v7.1, we are caching (Re-using) prepared statements wherever possible. If an operation can re-use same prepared statement as last time unlike earlier versions of IBM Tivoli Directory Integrator, theoretically we can get improved performance.

    With the enablement of caching on prepared statements in JDBC Connector, from IBM Tivoli Directory Integrator v7.1, we noticed the performance of JDBC Connector over DB2 got boosted by 50% when compared to Tivoli Directory Integrator v7.0.

3. **Tuning the Domino Change Detection Connector configuration:** This section describes some of the parameters for Domino Change Detection Connector, that, when set properly, can achieve a performance gain during its execution.

    • State Key Persistence: Make sure that the “State Key Persistence” parameter is configured with the required options. The connector stores the entries in the System Store depending on the State Key Persistence parameter. The “State Key Persistence” parameter affects the execution time.

- Deliver sorted: This parameter is used to tell the Connector whether the user wants the delivered notes document to be sorted or not. If there are too many documents to sort it might take a lot of time. The notes documents are sorted chronologically by date modified. Solution developer has to decide whether sorting is needed or not.

4. **Tuning the HTTP Server Connector configuration:** This section describes some of the parameters with HTTP Server Connector which can be set properly to obtain performance gain during its execution.

   - Connection Backlog: This parameter sets the maximum queue length for incoming connections. If a connection request arrives when the queue is full, the connection will be is refused. This parameter determines how large a connection buffer must be set up for this server socket by the TCP/IP driver of the operating system. Setting this parameter affects the reliability and scalability of a solution.

4 IBM Tivoli Directory Integrator Performance utilities

There are two performance tools included with IBM Tivoli Directory Integrator version 7.0: a performance test utilities tool and a performance debugging tool. These tools monitor and log system and server information at specified intervals of time. The information gathered can then be used for throughput measurement and capacity planning.

Refer to the IBM Tivoli Directory Integrator *7.1 Problem Determination Guide* for using these tools.

5 Benchmark results of various IBM Tivoli Directory Integrator Components

This section describes benchmark results of various IBM Tivoli Directory Integrator v7.1 Components with various backend data sources.

5.1 JDBC Connector with various backend databases

A test is conducted to analyze JDBC Connector's throughput in all of its supported modes modes on various backend databases like Apache Derby database in network mode, IBM SolidDB and DB2. The input feed needed for the test, taken from a CSV File system.

Test bench details: The test is conducted on machine with configuration, 3GB RAM, 40GB HDD, Intel Xeon Processor x86, Microsoft Windows 2003 Enterprise Edition. IBM SolidDB v6.3 and Apache Derby database v10.5.3 is running on same test machine whereas DB2 v95 is running on a remote machine.

Test results of JDBC Connector in all of its modes are shown below.

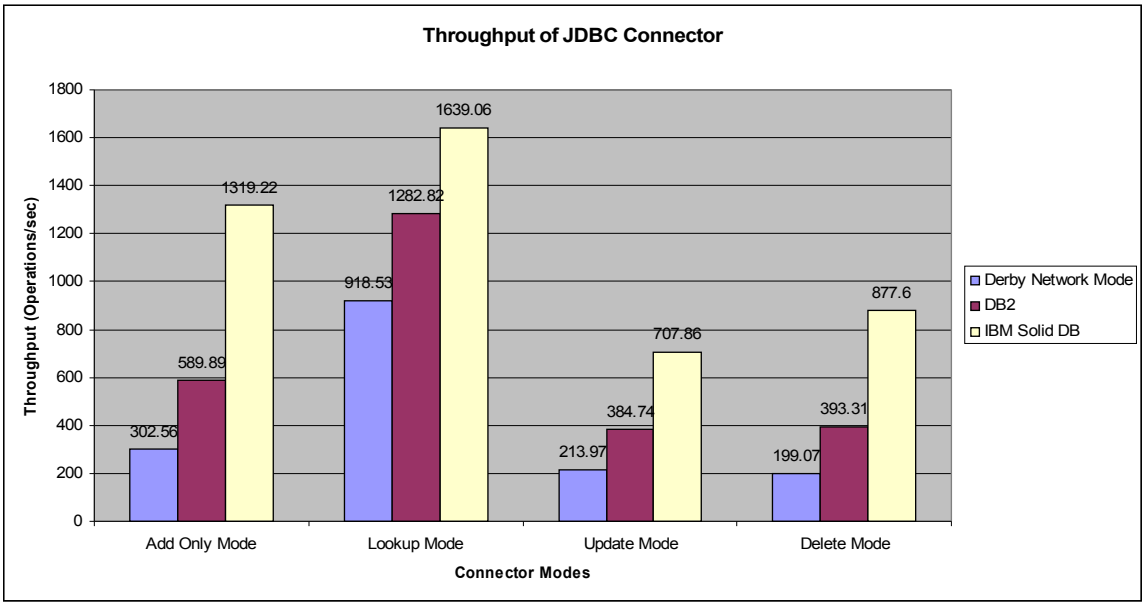

Figure: 5.1

The below chart comprises the test results of JDBC Connector in Iterator mode on various back end databases.

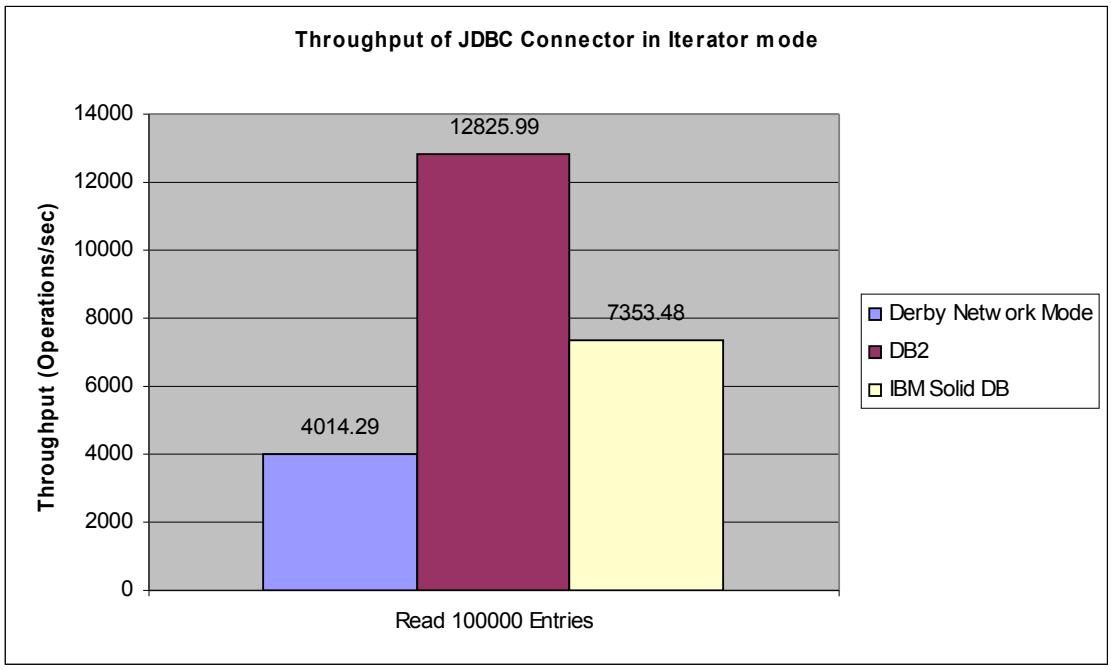

Figure: 5.2

5.2 LDAP Connector with various backend data sources

A test is conducted to analyze LDAP Connector's throughput in all its supported modes on various backend data sources like IBM Tivoli Directory Server and Microsoft Active Directory Server. The input feed needed for the test is taken from LDIF file. Test results of LDAP Connector in all its supported modes are shown below.

Test bench details: The test is conducted on machine with configuration, 3GB RAM, 40GB HDD, Intel Xeon Processor x86, Microsoft Windows 2003 Enterprise Edition. IBM Tivoli Directory Server v6.2 and Microsoft Active Directory are running on two different remote machines.

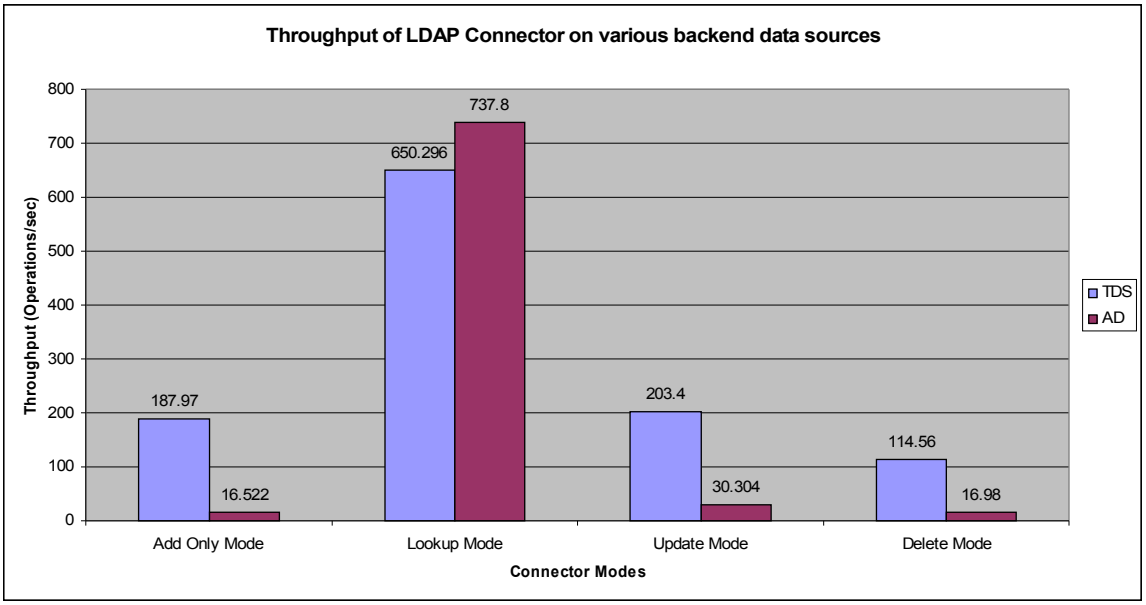

Figure: 5.3

The below chart comprises the test results of LDAP Connector in Iterator mode on various back end data sources.

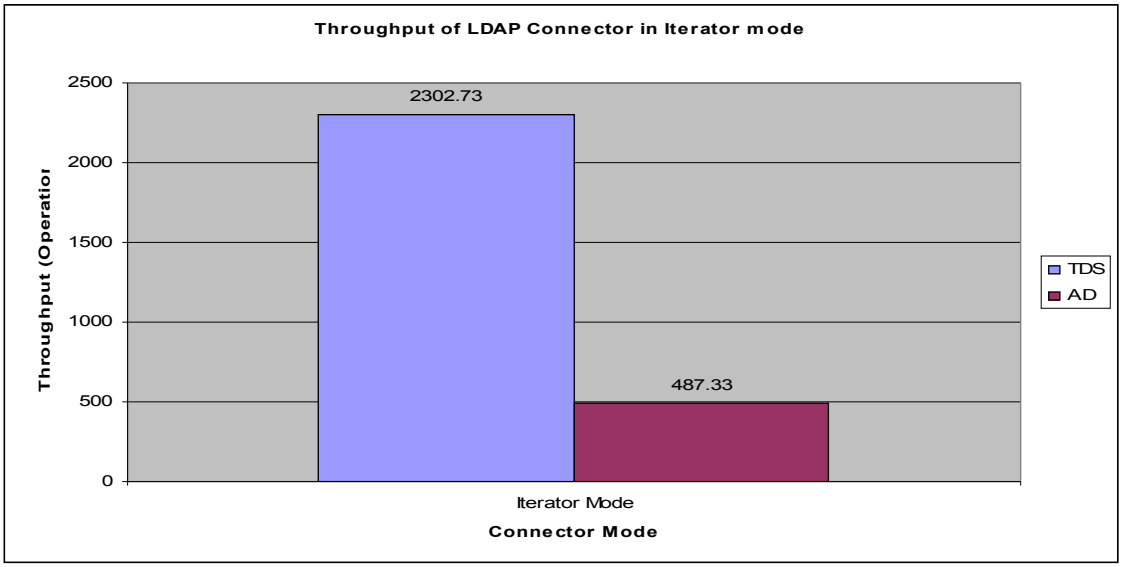

Figure: 5.4

5.3 File System Connector with various Parser configurations

A test is conducted to analyze File System connector's throughput in all its supported modes with various input/output parser configurations. The below chart depicts throughput of file system connector in iterator mode with various input parser configurations. As we can see CSV parser performs faster execution when compared other parsers

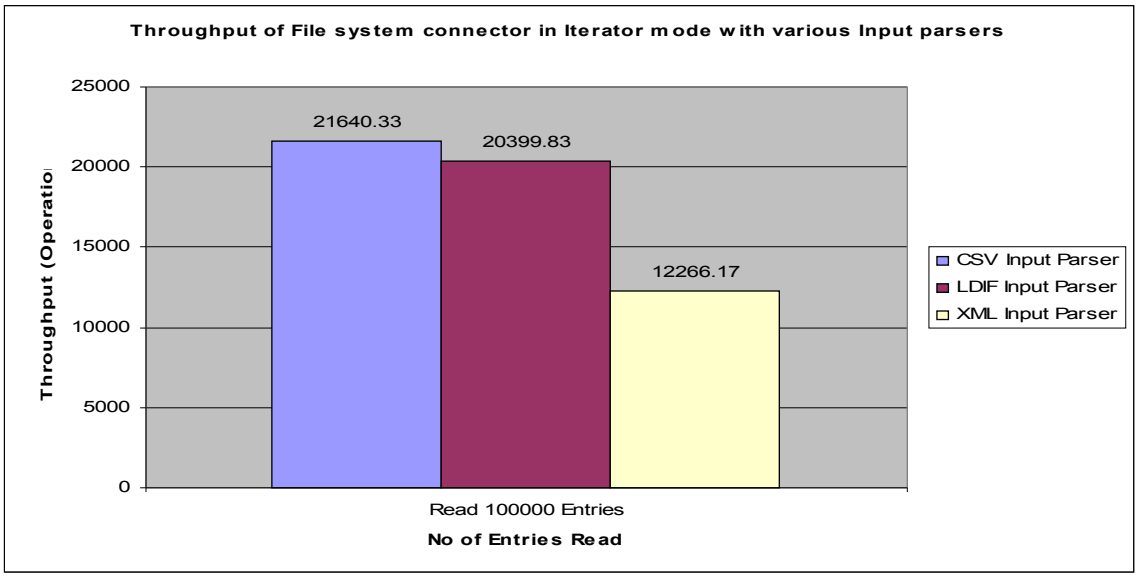

Figure: 5.5

The below chart depicts throughput of file system connector in Add only mode with various output parser configurations. The input feed needed for the test is taken from various formats of File system.

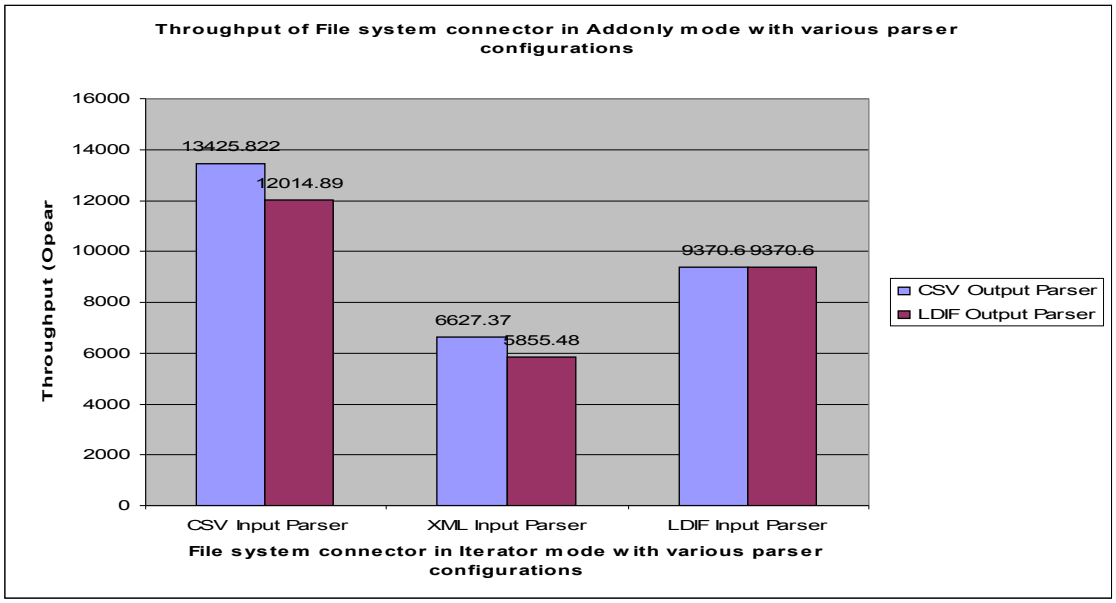

Figure: 5.6

6 Pointers to tune external Data Sources

This section documents links, pointers, references to various third party data sources, and repositories to which IBM Tivoli Directory Integrator connects. Tuning these data sources can improve the performance of IBM Tivoli Directory Integrator.

- DB2:
  http://www.redbooks.ibm.com/abstracts/sg246417.html
  http://www-128.ibm.com/developerworks/db2/library/techarticle/anshum/0107anshum.html

- Domino:
  ftp://ftp.software.ibm.com/software/lotus/lotusweb/product/domino/Dom_AIX_Perf_Tuning_Tips.pd
  http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4182.html?Open

- WebSphere Application Server:
  http://www.redbooks.ibm.com/abstracts/SG246392.html

- Tivoli Directory Server:
  http://www.redbooks.ibm.com/abstracts/redp4258.html
  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.IBMDS.doc/toc.xml

- Tivoli Identity Manager:
  http://publib.boulder.ibm.com/tividd/td/ITIM/SC32-5272-00/en_US/PDF/sc23-5272-00.pdf

- Tivoli Access Manager:
  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame3.doc_5.1/am51_perftune.pdf

- IBM SolidDB:
  http://publib.boulder.ibm.com/infocenter/soliddb/v6r5/index.jsp?topic=/com.ibm.swg.im.soliddb.inmemory.doc/doc/optimizing.and.tuning.the.server.html