Infor LN Performance, Tracing, and Tuning Guide for SQL Server

Publication Information

Release: Infor LN 10.x

Publication date: October 2, 2017

Document code: B0079D

Contents

About this guide

This document provides guidelines to optimize the performance of an Infor LN environment on a SQL Server database by tracing and tuning the environment.

Note: This document is a comprehensive compilation; however, there may be instances wherein relevant information or procedures may have been omitted. Therefore, it is recommended to verify the proposed changes in a test environment before moving to production. The information provided is applicable to the currently supported versions of Infor LN and Microsoft SQL Server and may not hold true for future versions of the SQL Server database.

Intended audience

This document is intended for intermediate to expert Infor LN and database Administrators and Technical Consultants and aims to better the performance of the Infor LN system.

Related documents

Certain sections in this document are described in more detail in other documents. The following documents help to extend the knowledge in specific areas:

- *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*
- *Infor LN - Sizing guide (B0045 US)*
- *Infor LN - Data compression (B0050 US)*
- *Infor Enterprise Server - Technical Reference Guide for* Microsoft SQL Server *Database Driver (U8173 US)*

You can find the documents in the product documentation section of the Infor Xtreme Support portal, as described in "Contacting Infor" on page 8 or navigate to https://docs.infor.com/ln.

This site is a good reference to the SQL Server documentation about tracing and tuning: http://msdn.microsoft.com/en-us/library/ms130214.aspx

Related Knowledge Base articles

Sections in this document are described in additional detail in other Infor Xtreme knowledge base articles. These articles help to extend the knowledge in specific areas:

- *22881401 - Generic Infor LN Performance solution*
- *1183466 - Platform Support Matrix*
- *1915943 - How to retrieve a SQL Server query plan using Extended Events*
- *1660883 - Investigate locking issues - error 107 / 201*

You can find these KB articles in the Infor Xtreme Support portal, as described in "Contacting Infor".

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at www.Infor.com/Inforxtreme.

If we update this document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@Infor.com.

Chapter 1 Tuning SQL Server for Infor LN

1

The performance of Infor LN is highly dependent on the database performance. This chapter describes the important tuning areas of SQL Server.

Introduction

Tuning SQL Server is not the standard method to repair your system. This chapter describes what can be done based on the performance issues you have identified.

Regarding the SQL Server performance tuning, it is important to note that each area of tuning is important for the overall performance, and must be approached using a logical methodology. Proper organization, logical changes, and documentation classify the difference between effective and ineffective tuning. Performance tuning along with capacity planning and sizing allows you to design, setup, implement, and maintain a smooth and optimal system.

In this chapter, the term CPU is used frequently and refers to the Windows Operating system description of this term. In practice, the term can be a CPU core or CPU thread. The term is described explicitly only when an exception applies.

Additionally, these topics are discussed detail to make the database 'tuning' process easier:

- SQL Server Information resources
- I/O Setup
- Storage setup
- Network setup
- Database maintenance
- Data volume
- SQL Server Parameters

SQL Server Information resources

For Generic SQL Server Information and SQL Server Tuning, information is available on the internet and in various Microsoft SQL Server books. However, most of these are copies of the SQL Server Books Online. Therefore, SQL Server Books Online (BOL) is a valuable information source for the SQL Server.

For more Information, see these books (Series):

- *Inside Series MSQL Server by the Microsoft Press*.
- *Microsoft SQL Server Internals* by the Microsoft Press.
- *SQL Server Internals and Troubleshooting* by the Wrox Press
- *SQL Server Magazine*. This magazine lists all the Microsoft SQL Server updates monthly.
- MSDN blogs of the SQL Server Customer Advisory Team (SQLCAT) Web site, where SQL Server Developers include additional information (not part of the official SQL Server build).

For SQL Server product lifecycle, see Microsoft support.

I/O setup

The Infor LN - Performance, Tracing and Tuning Guide (U9357) provides guidelines for a correct I/O setup. For the SQL Server, these additional guidelines are important:

- Configure Stripe sizes for the SQL Server volumes to the recommended size of 64K or a value divisible by 8K.
- Configure NTFS Allocation Unit for the SQL Server used file systems to the recommended size of 64K or a value divisible by 8K.
- Always use the newly configured and formatted disks for the SQL Server data and the log storage. Most SQL Administrators tend to check only the internal SQL Server data and index fragmentation. It is also important to regularly check for the existence of highly fragmented SQL Server file storage on the Windows file system level, when using traditional rotating disks.

Storage setup

A good storage setup is important for the SQL Server performance. See the SQLCAT website and the Microsoft knowledge base, for more information on I/O and Storage Setup best practices. These guidelines help design and implement the storage setup:

Filegroups

It is recommended to spread the disk I/O across multiple disks, based on your requirement. To accomplish this, SQL Server filegroups can be used. Using filegroups on a single disk does not improve performance. Possible scenarios of using SQL Server filegroups for I/O separation:

- 'Hotspots' (for example for a specific, heavily used, table as First Free Numbers) in the Infor LN database that exist for an extended period.

- Locate different Infor LN companies on different disks or volumes.
- Use read-only filegroup for a specific company. SQL Server can backup filegroups separately. There is no need to backup a read-only filegroup regularly, so this can reduce the size of daily backups. Note: always backup the primary filegroup together with backup of other filegroups.

The %BSE%/lib/msql/msql_storage_param supports using filegroups completely. This example displays the appropriate setup of %BSE%/lib/msql/msql_storage_param to separate a table and differentiate companies:

```
tcmcs050:100:*:FILEGROUP hotspot        # First Free Numbers hotspot, no compression
*:000:*:FILEGROUP tools                 # Company 000 separated, no compression
*:100:*:COMPRESS=1;FILEGROUP production # Production data using compression
*:200:*:COMPRESS=1;FILEGROUP archive    # Archive data using compression
*:300:*:COMPRESS=1;FILEGROUP readonly   # Read-only data using compression
*:*:*:COMPRESS=1;FILEGROUP other        # Remaining data using compression
```

Datafiles

Split the Infor LN database into multiple data files. For CPU cores up to 8 it is recommended to use one datafile for each core per filegroup. Above 8 CPU cores, use 8 datafiles and add an additional datafile for every two additional cores, because each data file starts with a header page, PFS, SGAM, and GAM page. Contention on these pages is reduced by creating multiple copies of these pages. This is done by creating multiple data files within each filegroup.

Autogrowth

- The Infor LN database must be created with an appropriate size. For example, if at least 150 GB is required for a year, create the database with the required size at the very beginning. Do not let the size increase to the required size which can cause additional fragmentation in the database and the NTFS volume.
- Set the Autogrowth option to a minimum of 1 GB (for each extent) to avoid fragmentation on file system level. Never use the default size of 1 MB.

  Use Instant File Initialization to allocate storage quickly. This will prevent zero-initializing the new allocated space, which is time consuming. Enabling Instant File Initialization affects these processes: create database, restore, add data file and data file growth.

  To enable the *Instant File Initialization* feature, add SQL Server service account (or group) to the Perform Volume Maintenance Tasks policy (*Administrative Tools/Local/Security Policy/Local Policies/User Rights Assignment*).

  To apply the modification in the Security Policy with immediate effect, execute command:

  ```
  gpupdate /force
  ```
- The *Auto-shrink* option must not be enabled in a production database because this leads to excessive fragmentation; consumption of CPU and I/O resources and generation of many virtual logs files (VLF).

Transaction log

The transaction log is critical to the overall operation of an SQL Server database and the ability to minimize data loss in the event of a disaster. The speed of the log drive is the most critical factor for a write intensive database. Consider these transaction log guidelines:

- Transaction log files primarily perform sequential writes and require the lowest possible write latency. Therefore, it is recommended to separate data file I/O from log file I/O to optimize I/O efficiency.
- The required size for the transaction log file depends on the size of the database, the amount of transactions and the backup frequency. Ideally, the transaction log must be as big as the largest index, to hold the transactions created during an index rebuilt operation.
- Pre-allocate the log file with a reasonable initial size. Consider creating the transaction log in chunks of 8 GB. Infor recommends managing transaction log growth explicitly, rather than letting the SQL Server manage. For safety reasons, you can set the Autogrowth option to 256 MB. It is not recommended to reduce the size of the transaction log and later allow incremental growth. SQL Server adds a predetermined number of VLFs each time the transaction log grows. Log files cannot take advantage of *Instant File Initialization*, so each log growth event is relatively expensive, compared to the data file growth, in terms of resources. Moreover, a fragmented transaction log can impede the performance of operations that read the log (log backups, recovery process, log shipping, and so on).
- The SQL Server transaction log file data increases because the SQL Server does not use circular transaction logging. You must set up an efficient log backup procedure to free up log space and maintain the log within acceptable values.

Tempdb

The I/O demand for the Tempdb by Infor LN is low in normal circumstances. Additional functionality that also includes the use of Infor LN databases, increases the I/O demand of Tempdb. Therefore, separation of Tempdb to its own volume is recommended. Split the SQL Server Tempdb database into multiple data files. It is recommended to use 1 data file for every 2 CPU cores, up to a maximum of 8 data files.

SQL Server connectivity

Infor LN uses ODBC (Open Database Connectivity) to connect to a SQL Server database. For optimal performance, ensure the installation of the latest ODBC drivers for your Operating System on the Infor LN application server. See https://docs.microsoft.com/en-us/sql/connect/odbc/windows/system-requirements-installation-and-driver-files, for more information.

Database maintenance

An optimized and well-performing database requires maintenance. A lot of tasks are automatically performed by the database engine, but other tasks must be performed using the automated scripts, preferably during a maintenance window. During this activity, the system can be utilized, but the user activity must be minimal. You must ensure there is no overlap between your batch window, backup window and your database maintenance window, to prevent concurrency issues. These required maintenance tasks are described in detail:

- Index maintenance
- Database integrity check
- Create/update statistics

Index maintenance

As data changes over time, indexes are fragmented. An index is fragmented when the physical ordering of the index pages on the disk do not match the logical order of the data (as defined by the index key). Heavily fragmented indexes can reduce the speed of data access and decrease query performance. Index fragmentation can be corrected by **reorganizing** or **rebuilding** the index.

When **reorganizing** an index, the index is still accessible for use on the SQL Server Standard Edition and Enterprise Edition. Offline rebuilding or re-creating the index prevents user access to the index. To allow a user access during the index rebuild, an online rebuild is required. Note: Rebuilding indexes online requires the Enterprise Edition of SQL Server.

Checking index fragmentation periodically and taking the required corrective actions is important to maintain the performance of your Infor LN environment. The fragmentation rate depends on the user activity, but it is recommended that you check for index fragmentation during your weekly maintenance window.
For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) Web site. These scripts check the fragmentation rate of the index and, if required, take the corrective action based on certain thresholds.

Database integrity

Although SQL Server ensures data integrity, the data can still get corrupted. For example, because of a hardware problem. Therefore, it is recommended to regularly check the integrity of the database. This can be done using automated scripts during the weekly maintenance window.

For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) web site. If any problems are detected, you can restore data from the backup or use one of the REPAIR options on DBCC CHECKDB.

Database statistics

Creating database statistics on a SQL Server is important. In most circumstances, the default auto create statistics database option creates the required statistics. When disabled, you are required to create statistics manually. Create missing statistics on the newly created data at regular intervals. You must create and update statistics during the maintenance hours, when activity on the system is minimal.

The following procedure creates missing statistics in a particular database:

```
sp_createstats @fullscan = 'fullscan', @indexonly = 'indexonly'
```

This procedure scans all the data and statistics created only for index. In SQL Server terminology, index means statistics for key distribution of the index, that is, the first column of an index key.

The following example creates missing statistics for all columns, including columns in a composite index key. This approach is recommended for creating statistics for a SQL Server. However, additional storage is required for statistics. Using the Fullscan option, the SQL Server Query optimizer can create an optimal plan in most cases:

```
sp_createstats @fullscan = 'fullscan'
```

The following procedure can be used to update statistics in a database:

```
sp_updatestats
```

To prevent shared schema (SCH-S) locks on the database during normal operation hours, it is recommended to set the SQL Server **AUTO_UPDATE_STATISTICS_ASYNC** option to ON. This updates statistics asynchronously with a minimal impact on the database concurrency.

```
ALTER DATABASE [inforlndb] SET AUTO_UPDATE_STATISTICS_ASYNC ON WITH NO_WAIT
```

Apart from the auto create/update facilities, it is recommended to update database statistics on a weekly basis during maintenance hours. For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) website.

Data Volume

The volume of data can increase faster than expected, especially when too much of history is stored or financial logging option is used. To reduce the data and save storage usage, consider these options:

- Database compression
- Varchar data type
- Archive historic data

Database Compression

From porting set 9.0a onwards, Infor LN supports table and index page compression. The benefits of data compression:

- Significant saving in disk storage space.
- More rows can fit on a page which results in less page read activity.
- Reduced disk I/O activity, because more compressed rows than the uncompressed ones are included in a page.
- Ability to compress older data that is not accessed often, such as archived tables, while the frequently accessed data can be stored in the uncompressed form.
- Possibility to free space no longer required for a table.
- Faster backup and restore.

Compression ratios of more than 5x are possible, depending on your data. See *Infor LN Data compression (B0050 US)* for more information.

Database compression is supported on SQL Server Standard Edition from SQL Server 2016 SP1 onwards.

Varchar data type

From porting set 8.4b onwards, the standard data types used by the Infor LN SQL Server database are VARCHAR and NVARCHAR. This significantly reduces data size and data growth. Consequently, the amount of I/O reduces and the overall performance improves.

Archive historic data

Regular archiving **of** history data can have a positive effect on performance. The database growth can be limited and inserts in large tables can be faster. See *Infor LN User Guide for Archiving (U9352 US)*. It is recommended to rebuilt indexes after the archiving process is completed.

SQL Server parameters

In most cases, Infor LN runs efficiently with an out-of-the box SQL Server database with standard parameters. Modifications of parameters can eliminate issues or influence resource usage on high-end SQL Server systems. The parameters used by Infor LN benchmarks and customers are listed in this table.

sp_configure parameter	recommended value	importance
Affinity (64) I/O Mask	Default (0)	Optional (High-end only)
Affinity (64) mask	Default (0)	Optional (Special purpose)
Blocked process threshold	Default (0)	Optional (Tracing purpose)

sp_configure parameter	recommended value	importance
Max degree of parallelism	# cores / 2	Recommended
cost threshold for parallelism	50	Recommended
Min server memory	Default (0)	Recommended
Max server memory	128 GB on Standard Edition	Recommended
Priority boost	Default (0)	Optional (Special purpose)
Recovery interval	Default (0)	Optional (Special purpose)
Backup compression	Default (0)	Recommended
Lock Pages in Memory	According value	Recommended
Large Pages	Trace flag 834	Optional (High-end only)
Plan cache size	Trace flag 8032	Optional (High-end only)

Caution: Although this does not occur often, setting one or more of these parameters in the database can lead to issues in other products or Infor LN sessions. Therefore, it is recommended that you perform a validation test before the implementation.

You can use these commands to change the most commonly used SQL Server options for Infor LN:

```
--Make sure you don't have any pending changes
IF EXISTS(SELECT * FROM sys.configurations WHERE value <> value_in_use)
        RAISERROR ('You have pending changes.', 0, 1)
ELSE
        BEGIN
                    exec sp_configure 'show advanced options', 1
                    reconfigure with override
                    exec sp_configure 'max server memory (MB)', 262144  -- 256GB
                    exec sp_configure 'max degree of parallelism', 8
                    exec sp_configure 'cost threshold for parallelism', 50
                    exec sp_configure 'remote admin connections', 1
                    exec sp_configure 'backup compression default', 1
                    reconfigure with override
        END
```

Affinity (64) I/O Mask

The Affinity I/O Mask option binds the SQL Server disk I/O to a specified subset of CPUs. In high-end online transactional processing (OLTP) environments, this extension can enhance the performance, of the SQL Server threads issuing I/Os by storing the threads on the CPUs which process the I/O and at the Windows level. This must be used in conjunction with the ‘affinity mask’ setting. This enhancement does not support hardware affinity for individual disks or disk controllers.

Affinity (64) mask

By segregating the SQL Server threads to run on specific CPUs/Cores, Windows can evaluate the system's handling of processes in a better way. For example, on an 8-CPU server running two instances of SQL Server (instance A and B), the system administrator can use the Affinity Mask option to assign the first set of 4 CPUs to instance A, and the second set of 4 to instance B. The same is applicable when segregating the SQL Server from Infor LN in a 2-Tier setup.

Blocked Process Threshold

Use the Blocked Process Threshold option to specify the threshold in seconds, after which the blocked process reports are generated. The threshold can be set from 0 to 86,400. By default, no blocked process reports are generated. This event is not generated for system tasks or tasks waiting on resources that do not generate detectable deadlocks.

By default, in SQL Server, the deadlock detector checks every 5 seconds if your queries are in a 'deadly embrace'. You can use an option that asks SQL Server to use this resource to check for long term blocking, and issue a report.

You can use this command to set the blocked process threshold for 20 seconds or longer:

```
sp_configure 'show advanced options', 1
go
reconfigure with override
go
sp_configure 'blocked process threshold', 20
go
reconfigure with override
go
```

Caution: You must only configure the blocked process report to be issued for values of five seconds or higher. Microsoft issues a warning that you could cause the deadlock detector to run continuously and kill your performance when you set this from 1-4: http://technet.microsoft.com/en-us/library/bb402879.aspx

You can define an alert to be implemented when this event is generated. For example, you can (choose to) page the administrator to take appropriate action to handle the blocking situation.

Max degree of parallelism

When the SQL Server runs on a computer with more than one CPU, the server detects the degree of parallelism, that is, the number of processors employed to run a single statement, for each parallel plan execution. You can use the Max degree of parallelism option to limit the number of processors that must be used in the parallel plan execution. The default value of 0 uses all available processors. Set the Max degree of parallelism to 1 to suppress parallel plan generation. Set the value to a

number greater than 1 (up to a maximum of 64) to restrict the maximum number of processors used by a single query execution. If a value greater than the number of available processors is specified, only the actual number of available processors is used. If the computer has only one processor, the Max degree of parallelism value is ignored.

It is recommended to set the Max degree of parallelism to `number of cores / 2` when running a separate database server and `number of cores / 4` when running Infor LN in host mode. The maximum value must not exceed 8. Infor LN queries are usually short and do not benefit much from parallelism. However, when Infor LN is used with different applications, such as reporting; parallelism can be beneficial. Parallelism can also be beneficial for database maintenance tasks.

The default value of 0 can affect the Infor LN Performance if a heavy query or a query with performance issues is used on the SQL Server.

You can use this command to set the Max degree of parallelism to 8:

```
exec sp_configure 'max degree of parallelism', 8
go
reconfigure with override
go
```

From SQL Server 2016 onwards, the MAXDOP parameter can be configured for each database using a new command:

```
ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 8;
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET MAXDOP=4;
```

Cost threshold for parallelism

The SQL Server optimizer uses the cost threshold for parallelism parameter to determine when the evaluation of the plans, that can use multiple threads, must be started. The cost refers to an estimated elapsed time (in seconds) required to run the serial plan on a specific hardware configuration. The default value of 5 is too low. It is recommended to increase this value to 50 to prevent creating parallel execution plans too frequently.

You can use this command to set the Cost threshold for parallelism to 50:

```
exec sp_configure 'cost threshold for parallelism', 50
reconfigure
go
```

Max server memory

By default, SQL Server dynamically allocates system memory. However, the memory size can increase to the system maximum which can be an issue, because memory is released only when the memory pressure is high.

Use the Max server memory option, to reconfigure the SQL Server instance memory (in megabytes). These rules provide a general guideline for tuning SQL Server memory for Infor LN:

- At least 4GB of memory must be available for the Operating System.
- Memory is inexpensive and not subject to licensing. As memory access is much faster than disk access, it is recommended to access data from memory as much as possible. Therefore, a large SQL Server buffer cache is beneficial for the Infor LN performance. The size depends on the SQL Server edition, the size of the database and the amount of data pages accessed frequently. Consider configuring up to 128 GB of memory when running SQL Server Standard Edition (SQL Server 2016 SP1 onwards) and more when running the SQL Server Enterprise Edition.
- With more than one instance (or when running Failover Cluster Instances), ensure that the max server memory setting is adapted for each instance, and the total of all parameters must be less than the total amount of physical memory.
- By default, the Min server memory setting is set to 0 and it recommended to not modify this value.

You can use this command to set the Max server memory to 256 GB:

```
exec sp_configure 'max server memory (MB)', 262144
go
reconfigure with override
go
```

Max worker threads

Use the Max worker threads option to configure the number of worker threads available for the SQL Server processes. SQL Server uses the native thread services of the Windows operating system, so that one or more threads support each network that SQL Server supports simultaneously. Another thread manages the database checkpoints, and a pool of threads manages all the users.

By default, this parameter value is set to 0 and it is recommended to not modify this value. This allows SQL Server to consider the appropriate number of worker threads.

Priority boost

Use the Priority boost option to specify if the SQL Server must run at a higher Windows scheduling priority when compared to the other processes, on the same computer. If you set this option to 1, the SQL Server runs at a priority base of 13 in the Windows scheduler. The default is 0, for which the priority base is 7. This parameter can be used, in specific scenarios, for a 2-tier system with a high load.

Recovery internal

Use the Recovery interval option to set the maximum number of minutes per database that the SQL Server requires, to recover databases. Each time an instance of SQL Server starts, each database

is recovered, transactions that are not committed are rolled back, and transactions that are committed but the changes are not yet written to the disk when the SQL Server stopped, are rolled forward. This configuration option sets an upper limit on the time required to recover each database. The default is 0, indicating an automatic configuration by the SQL Server, which means a recovery time of less than one minute and a checkpoint of approximately one minute, for the active databases.

Retain the Recovery interval set to 0 (self-configuring) unless you notice that the checkpoints hinder performance because of frequent occurrence. If so, it is recommended that you increase the value in small increments. Increasing the recovery interval results in 'heavier' checkpoints and creates additional pressure on the I/O subsystem.

Backup compression

You can enable SQL Server Backup compression to reduce disk I/O. The size of the compressed backup is significantly smaller and the backup/restore duration, shorter.

You can use this command to enable Backup compression:

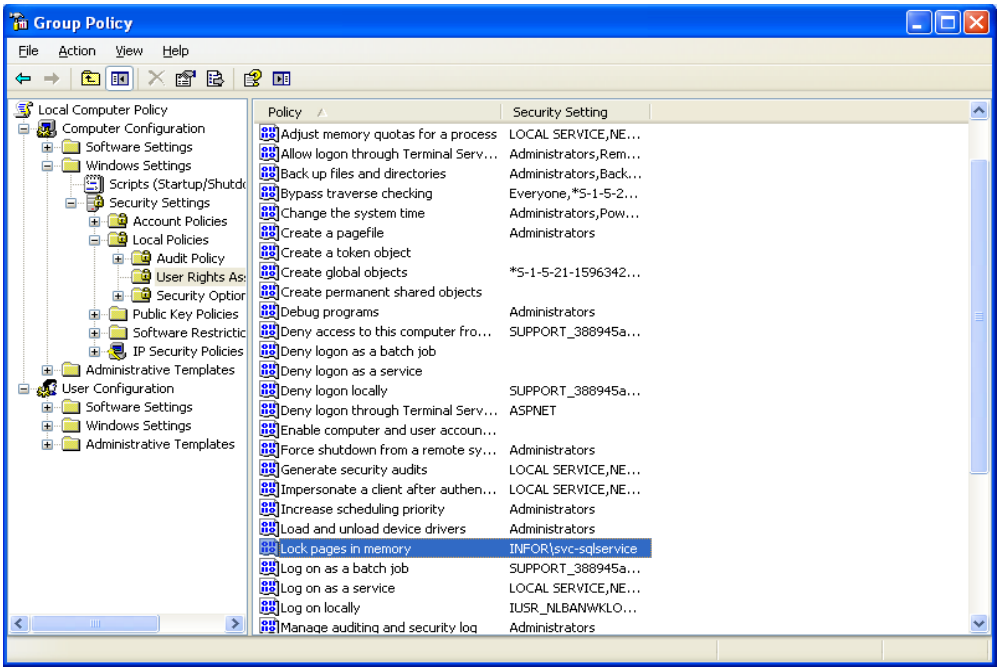
```
exec sp_configure 'backup compression default', 1
go
reconfigure with override
go
```

Note: Compression increases CPU usage, and the additional CPU consumed by the compression process might adversely impact concurrent operations. Schedule backup activities as much as possible during the period of low user activity. Changing default backup compression does not necessarily affect a 3rd party backup utility.

Lock pages in memory

Memory pages are locked in memory to enable faster allocations. SQL Server memory pages are non-pageable when locked in memory. Windows cannot use these pages when aggressively trimming the memory. To assign the 'Lock pages in memory' privilege, you must modify the Group Policy on the server with group policy editor (start with gpedit.msc).

Navigate to the User-Rights Assignment folder and add the (service) account to the ' Lock pages in memory' privilege.



Large pages

When using a larger page size for memory as organized by the kernel, the process of virtual address translation is faster. The normal page size for Windows memory is 4 KB on x64 systems. When using large pages, the page size is 2 MB.

SQL Server supports the concept of Large Pages when allocating memory for some internal structures and the buffer pool. But the use of large pages for the buffer pool is not recommended for every customer situation and the usage must be tested and planned.

When the SQL Server instance is started, a decision on using a large page support from Windows is taken, based on these three conditions:

- using SQL Server Enterprise Edition
- the system has at least 8 GB physical RAM available after the boot process
- the 'Lock Pages in Memory' privilege is set for the SQL Server service account

If these conditions are true, the SQL Server starts the 'initialize' process for the LargePageAllocator for each memory node on the computer. The ERRORLOG displays these messages:

```
2009-06-04 12:21:08.16 Server      Large Page Extensions enabled.
2009-06-04 12:21:08.16 Server      Large Page Granularity: 2097152
2009-06-04 12:21:08.21 Server      Large Page Allocated: 32MB
```

The Large Page Granularity is the minimum size of a 'large page' on a specific Windows Platform.

Large Pages are enabled by the trace flag 834. From SQL Server 2012 onwards, large pages are allocated for all SQL Server memory areas. In previous versions, SQL Server allocated large pages only for the buffer pool.

If the engine detects that the trace flag 834 is enabled when the database is started, all memory defined by the **Max server memory** setting is allocated using the LargePageAllocator. The memory allocation is a one-time activity.

Use trace flag 834 in combination with Infor LN, when using:

- Windows 2012, or higher.
- SQL Server on a dedicated database server or ensure that sufficient memory is available for the SQL Server
- a heavily loaded Infor LN SQL Server database instance with memory usage above 16 GB
- a NUMA based system with 2 or more CPU sockets

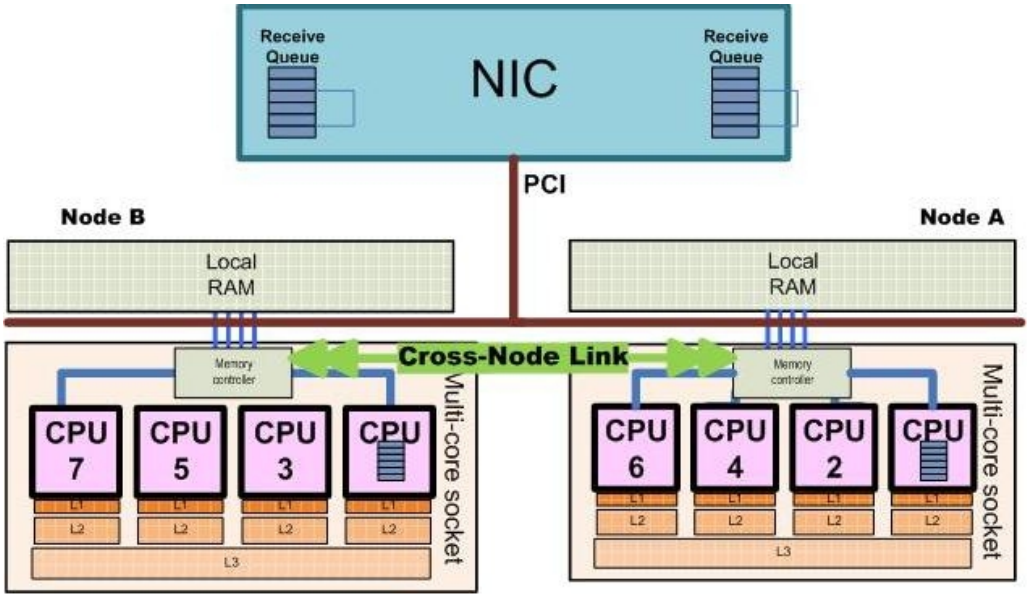
It is difficult to provide accurate figures and estimates for the increased performance levels when trace flag 834 is used. On an average, a performance increase in the range of 10 – 30 percent can be expected for heavily loaded SQL Server instances with Infor LN.

Caution: Do not over allocate memory for the SQL Server. When using the trace flag 834, ensure that the memory allocated by the SQL Server is at least 10 percent lower than the available system RAM. When additional memory is allocated by the SQL Server, the system can change to a 'low memory' state, which triggers housekeeping activities such as trashing the plan cache. Therefore, it is recommended to carefully monitor the SQL Server plan cache hit ratio in 'perfmon', which must be > 95 percent.

SQL Server and NUMA

SQL Server is designed to take advantage of NUMA-based computers without requiring any application changes. Earlier, NUMA was used by hardware vendors for 'big iron' to resolve SMP-based (Symmetric Multi-Processor) scaling issues. Currently, all the latest CPUs include the interconnect technology which makes all systems NUMA based computers. Modern processors have multiple cores per socket. Each socket is represented, usually, as a single NUMA node.

The diagram shows the NUMA architecture in detail:



In principle, a NUMA based system is split into nodes, with a NUMA-node for each Multi-core socket. This means that the system memory is also divided across the nodes. Local memory can be quickly accessed but memory on other node(s) must be accessed using the cross-node link with more latency.

There are various tuning recommendations for the SQL Server and NUMA, especially for the high-end usage. However, these are the basic recommendations:

- If the customer requirement is small, do not tune the database to prevent additional complexity.
- For Infor LN running in a 2-Tier configuration, assign the SQL Server instance to a dedicated NUMA node using the Affinity mask setting. SQL Server memory access on the other node(s) is always more expensive, and if the CPU capacity of all NUMA-nodes is not required, you can prevent expensive cross-node access.
- SQL Server memory (max server memory parameter) is divided across the available NUMA nodes. On most systems, there is no (partial) memory sharing between NUMA nodes. This means additional total memory is required for the SQL Server to meet the per NUMA node memory requirements, when compared with a non-NUMA system.

Software NUMA

The SQL Server database engine partitions various internal structures and service threads per NUMA node. With processors containing 10 or more cores per socket, using software NUMA (soft-NUMA), to split hardware NUMA nodes, generally increases scalability and performance.

With SQL Server 2016, whenever the database engine server detects more than 8 logical processors at startup, software NUMA nodes are created automatically by default. Physical and hyper-threaded processor cores are not differentiated when counting logical processors. When the number of logical processors detected are more than 8 per socket, the database engine service creates soft-NUMA nodes that ideally contain 8 cores, but can go down to 5 or up to 9 logical

processors per node. The size of the hardware node can be limited by a CPU affinity mask. See the SQL Server ALTER SERVER CONFIGURATION option. The number of NUMA nodes never exceed the maximum number of supported NUMA nodes.

Infor LN benchmarks running over 1000 concurrent users shows slightly better performance running SQL Server 2016 with Soft-NUMA off. Consider disabling the software NUMA after extensive testing. Execute these steps to disable soft-NUMA:

1. Stop the instance of SQL Server Agent.
2. Execute: ALTER SERRVER CONFIGURATION SET SOFTNUMA OFF
3. Re-start the SQL Server instance.
4. Start the instance of SQL Server Agent.

You can use these DMVs to view the current state and configuration of soft_NUMA:

- sp_configure: Displays the current value (0 or 1) for SOFTNUMA
- sys.dm_os_nodes: The node_state_desc column for each NUMA node shows whether the node is created by soft-NUMA.
- sys.dm_os_sys_info: The softnuma and softnuma_desc columns show the configuration values.

For more information, see https://msdn.microsoft.com/en-us/library/ms345357.aspx.

Plan cache size

Infor LN benchmarks on SQL Server 2012 showed that SQL Server continually compiled query plans. This normally occurs in a period after the cold start of the database. Compiling query plans is an expensive CPU operation. In the benchmark, this occurs when more than 500 users access the database. If you encounter this issue, test whether the Optimize for ad hoc workloads configuration option can resolve the issue. The Optimize for ad hoc workloads option is used to improve the efficiency of the plan cache for workloads that contain many single use ad hoc batches. When this option is set to 1, the Database Engine stores a small compiled plan stub in the plan cache when a batch is compiled for the first time, instead of the full compiled plan. This helps to relieve memory pressure by not allowing the plan cache to be filled with compiled plans that are not reused.

You can use these commands to enable the Optimize for adhoc workloads option:

```
--Make sure you don't have any pending changes
IF EXISTS(SELECT * FROM sys.configurations WHERE value <> value_in_use)
        RAISERROR ('You have pending changes.', 0, 1)
ELSE
        BEGIN
                    exec sp_configure 'show advanced options', 1
                    reconfigure
                    exec sp_configure 'optimize for ad hoc workloads', 1
                    reconfigure
        END
```

If not sufficient, you can implement SQL Server trace flag 8032, to use the SQL Server 2005 plan cache behavior and in general allow the plan cache size to be larger. When implementing trace flag 8032, ensure that your SQL Server has configured using sufficient memory for the buffer cache. See http://msdn.microsoft.com/en-us/library/ms188396.aspx.

Chapter 2 Tuning Infor LN for SQL Server

2

This chapter describes the important resource settings and performance parameters in Infor LN running on a SQL Server database.

Important db_resource parameters

The default db_resource values provide a balance between the performance and memory usage. The following parameters are important for performance, and in certain circumstances it may be necessary to change the same.

Recommended OLTP settings

It is recommended to set this parameter in the <BSE>/lib/db_resource file:

```
msql_serverhost:<dbserver>\inforln
```

For 3-tier OLTP it is recommended to add these parameters in the <BSE>/lib/db_resource file:

```
msql_max_arrsz:10
msql_opt_rows:10
```

Note: the array interface (msql_array_fetch / msql_array_insert) is enabled by default.

For parameter details see *Infor Enterprise Server Technical Reference Guide for SQL Server Database Driver (U8173 US).*

Recommended batch settings

For batches, 2-tier and 3-tier, an alternative db_resource file can be used to increase the performance of batches, without affecting the OLTP performance. This can be done using the USR_DBS_RES environment variable that must be set only for the users executing the batch jobs.

Example:

```
-set USR_DBS_RES=lib/defaults/db_resource.batch
```

See the ‘Tuning Infor LN’ topic in the *Infor LN Performance, Tracing and Tuning guide (U9357 US).*

These database driver settings for batches, are recommended:

```
msql_max_arrsz:50
msql_opt_rows:50
msql_retained_cursors:200
bdb_max_session_schedule:250
```

Using these parameters, increases the memory usage of the bshell (and including database driver), but can save the CPU cycles, to execute the batch faster.

Note: Settings in <BSE>/lib/defaults/db_resource are loaded and the driver resources set in the alternative resource file overrides the setting of the same driver resource in db_resource.

msql_opt_rows

The msql_opt_rows resource allows you to specify a hint that the first ‘n’ rows must be retrieved faster from the database. This allows the SQL Server to optimize the fetch request. Based on this value, the SQL Server determines a suitable communication buffer size to improve the performance. By default, the value of this parameter is equal to the default array fetch size.

msql_retained_cursors

The msql_retained_cursors resource sets the number of inactive (breaked) cursors that must be retained in the list, for reuse. These cursors can be reused and prevent a prepare/bind overhead. Additional resources, such as, memory until the cursors are closed and released, are required.

The default value is 20, which is a good starting point for most customers in a 2-tier OLTP. For batches, it is recommended to increase this parameter to 200.

msql_serverhost

Use the msql_serverhost resource to specify a host name for the driver to locate the SQL Server instance that must be used. You can specify a network protocol and a SQL Server Instance:

```
msql_serverhost:<dbserver>\inforln
```

In this example, ‘dbserver\inforln’ specifies the use of ‘inforln’ SQL Server Instance on the server ‘dbserver’. See the ‘Tuning SQL Server for Infor LN’ topic for optimal SQL Server network settings.

msql_array_fetch

The msql_array_fetch resource is used to enable or disable the array fetch interface. The valid values are 0 and 1. When set to 0, the array fetch interface is disabled; when set to 1, the interface is enabled. The default value for this option is 1.

msql_array_insert

The msql_array_insert resource is used to enable or disable the array insert interface; the valid values are 0 and 1. When set to 0, the array insert interface is disabled; when set to 1, the interface is enabled.

Note: The database driver disables this option under specific circumstances. For example, if references must be checked or updated, or the application requires an immediate response from the driver, no array insert can be executed.

msql_max_arrsz

If the msql_array_insert resource is enabled, the msql_max_arrsz parameter is used to define the maximum number of rows inserted at once in the RDBMS. If the msql_array_fetch resource is enabled, the msql_max_arrsz parameter is used to define the maximum number of rows fetched at once from the RDBMS.

msql_lock_timeout

The msql_lock_timeout resource parameter is used to determine the timeout value (in seconds) for queries, blocked by locks in the database server. The default value is 10, which means 'one waits for 10 seconds for the lock to be released'. When set to -1, the driver waits indefinitely for the locks to be released; when set to 0, the driver does not wait for the locks to be released. It is recommended to use this resource carefully.

When batches are executed during normal working hours, and in some high concurrent OLTP workloads, it can be beneficial to increase the timeout to 60 seconds. The transactions hold longer to acquire the lock rather than re-executing the query from the retry point.

msql_no_index_hint

From portingset 91b.01, the Infor LN database driver suppresses generating query index hints. This allows the SQL Server optimizer to choose the optimal index to be used. This is the optimal and recommended setting. Customers running a portingset prior to 91b.01 are recommended to upgrade the portingset or set the msql_no_index_hint resource to 1.

Chapter 3 Troubleshooting SQL Server

3

Besides the Infor LN troubleshooting possibilities, SQL Server in combination with Windows includes powerful tools to troubleshoot. It is recommended that you start with a global tool (as Windows Task Manager) and start the detailed tracing based on symptoms identified using the global tools. See http://msdn.microsoft.com/en-us/library/ms179428.aspx, for more information on SQL Server performance tools.

Windows Task Manager

The Windows Task Manager is recommended as a starting point for any tracing and tuning exercise. Do not use other advanced tools as the root cause is displayed in the Task manager.

SQL Server Management Studio Activity monitor

The Activity Monitor available in the SQL Server Management Studio can be used by database developers and administrators for an overview of the SQL Server system performance.

See http://msdn.microsoft.com/en-us/library/cc879320.aspx for more information.

SQL Server Management Studio Reports

SQL Server Management Studio generates built-in performance reports at the server level and the database level. Right-click the Infor LN database icon in the Object Explorer pane and select Reports > Standard Reports to access the reports. Most reports are based on dynamic management views (DMVs). Apart from these standard reports, Microsoft provides performance dashboard reports, which can be downloaded from the Microsoft website. The performance dashboard reports allow a database administrator to quickly identify a bottleneck on the system, and capture additional diagnostic data that is required to resolve the problem.

SQL Server Dynamic Management Views (DMV)

The DMVs are used to display the internal memory structures within the SQL Server address space. DMVs and dynamic management view functions (DMFs) reflect the server processes or all the sessions on the server. DMVs can be used for diagnostics, memory and process tuning, and monitoring the sessions on the server. DMVs alleviate the requirement to monitor the server with tools, such as Profiler and Performance Monitor (perfmon), for the diagnosis of the performance issues. Additionally, DMVs can provide Information about performance issues, after the issues occur. See http://msdn.microsoft.com/en-us/library/ms188754.aspx.

Tracing with sqlcmd

Usually the graphical plan is easy to use but in some scenarios, it is beneficial to use the command prompt utilities. The SQL query can be executed on the command prompt with the 'sqlcmd' utility. sqlcmd -? is used to display the options available. See http://msdn.microsoft.com/en-us/library/ms162773.aspx and http://msdn.microsoft.com/en-us/library/hh213540.aspx.

These options provide useful information when analyzing query performance:

- SET_SHOWPLAN_TEXT {ON|OFF}. When set to ON, the query execution plan information is displayed. The query is not executed. Example:

```
sqlcmd -S <dbserver>\inforln -d inforlndb -U sa -P <password>
1> SET SHOWPLAN_TEXT ON
2> GO
1> select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
2> GO
StmtText
------------------------------------------------------------------------
 select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"

(1 rows affected)
StmtText
------------------------------------------------------------------------
 |--Nested Loops(Inner Join, OUTER REFERENCES:([inforlndb].[dbo].[ttcibd001090
   |--Index Seek(OBJECT:([inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_2a]),
   |--Clustered Index Seek(OBJECT:([inforlndb].[dbo].[ttcibd001090].[Ittcibd00

(3 rows affected)
```

  The example does not display the complete result set due to wrapping. To prevent wrapping, it is recommended that you expand the command prompt window.

- SET_SHOWPLAN_ALL {ON|OFF}. When set to ON, the query execution plan information is displayed. The option is used to return information as a set of rows that form a hierarchical tree representing the steps executed by the SQL Server query processor, for the processing of each statement. Each statement displayed in the output contains a single row with the text of the statement, followed by several rows with the details of the execution steps. The query is not executed. Example:

```
sqlcmd -S <dbserver>\inforln -d inforlndb -U sa -P <password>
```

```
1> set SHOWPLAN_ALL ON
2> go
1>  select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
2> go
StmtText
                                                                                DefinedValues
-------------------------------------------------------------------------------------------
--------
------------------------------------------------------------------ -----------------
--------
 select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
                                                                                NULL
  |--Nested Loops(Inner Join, OUTER REFERENCES:([
inforlndb].[dbo].[ttcibd001090].[t_item], [Ex
                                                                                NULL
       |--Index Seek(OBJECT:([ inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_2a]),
SEEK:([inforlndb
]=N'COST SET 1') ORDERED FORWARD
[inforlndb].[dbo].[ttcibd001
       |--Clustered Index Seek(OBJECT:([
inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_1a]), S
]=[ inforlndb].[dbo].[ttcibd001090].[t_item]) LOOKUP ORDERED FORWARD
[inforlndb].[dbo].[ttcibd001

(4 rows affected)
```

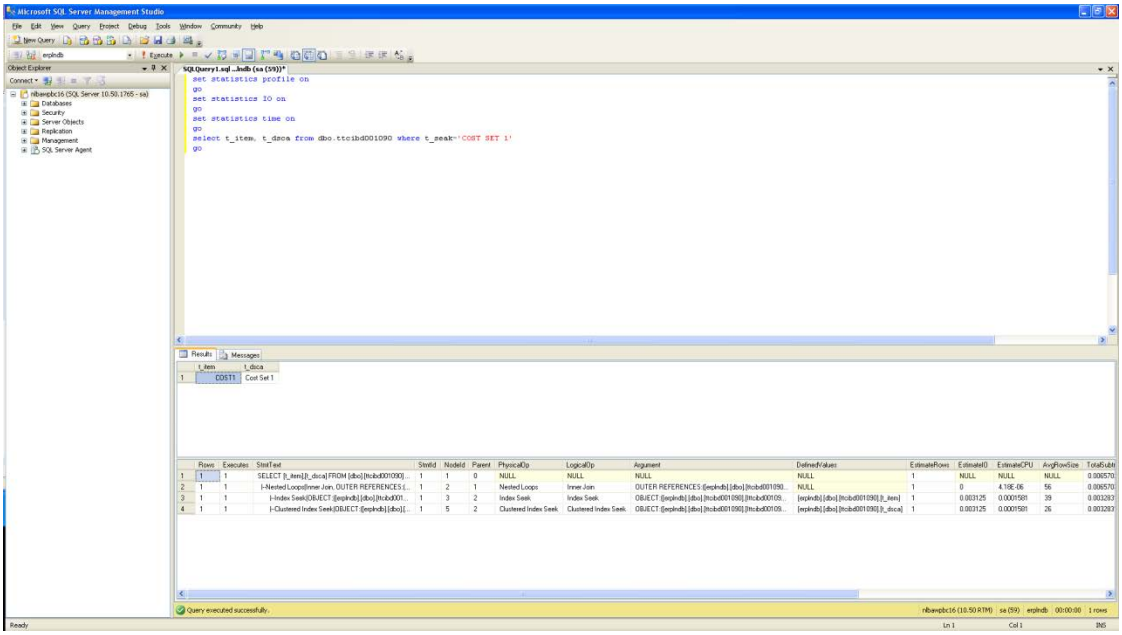
- SET STATISTICS PROFILE {ON|OFF} does not prevent the execution of the query. Apart from returning the same Information as SET SHOWPLAN_ALL, the query also displays two additional columns: Rows and Executed. Rows contains the numbers of rows returned, and Executed contains the actual number of times the operator executed the query.
- SET STATISTICS IO {ON|OFF} displays the count of table accesses, logical and physical reads, and read ahead reads for each T-SQL statement.
- SET STATISTICS TIME {ON|OFF} displays the time in milliseconds to parse the statement, compile the query, and execute the query.

See http://msdn.microsoft.com/en-us/library/ms190356.aspx for more information about profiling options.

SQL Server management studio

One of the best methods to trace the SQL Server is to use the SQL Server Management Studio.

Running a query in ‘sqlcmd’ mode In the SQL Server management studio displays the same output as the ‘sqlcmd utility’. This ensures that the information is formatted in a better way:



The tab messages display the additional statistics information:

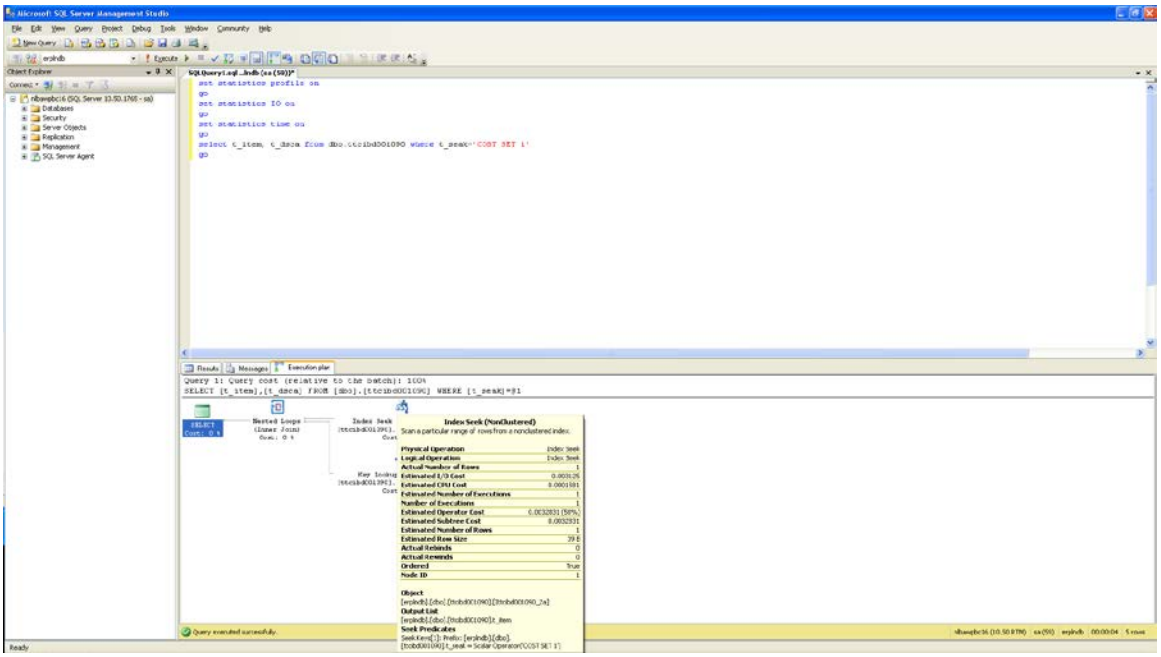
```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

(1 row(s) affected)
Table 'ttcibd001090'. Scan count 1, logical reads 7, physical reads 0, read-ahead
reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(4 row(s) affected)

SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 1 ms.
```

When you enable ‘Include Actual Execution Plan’, on the ‘query’ tab, displays the graphical query execution plan. The diagram displays an example of the trace output in SQL Server Management Studio.



Management Data Warehouse (MDW)

The Performance Data Collection and Warehouse can be used by database administrators to gather performance related data using the built-in data collectors which are also known as data collection containers. Using data collectors, you can gather performance data from multiple SQL Servers and store the data in a Management Data Warehouse (MDW). When you successfully configure the Management Data Warehouse (MDW) with the Disk Space**,** Query Statistics, and Server Activity Data Collection Sets, multiple built in reports can be generated for analysis. These reports can be executed from within the SQL Server Management Studio (SSMS). See http://msdn.microsoft.com/en-us/library/bb677179.aspx.

SQL Server Profiler

The SQL Server Profiler is a powerful SQL Server tracing tool. Creating a SQL Server Profiler snapshot of a badly performing query or multiple statements, helps you identify the reason for the performance issue. You can use the SQL Server Profiler feature to correlate a Profiler trace with Performance monitor (perfmon) data. The impact of the performance on the database server correlates directly to the Profiler trace and vice versa. See http://msdn.microsoft.com/en-us/library/ms191152.aspx.

SQL Server Extended Events

When compared to the SQL Server profiler, Extended Events is a light weight server-side performance monitoring system. Events in SQL Server can be captured using the SQL Server Management Studio (SSMS) management node. The events for CPU, Memory, disk, database, cursor usage, locks, latches and so on. Results can be stored in a file or table.

See InforXtreme knowledgebase article 1915943 for a script to quickly setup an Extended Event trace.

See http://msdn.microsoft.com/en-us/library/bb630282.aspx for more information about Extended Events.

Note: Some events can have significant performance impact. Enable only the events needed to trace a specific issue and ensure that you filter as much as possible.

Troubleshooting with SQLDiag utility

SQLDiag is a diagnostics collection utility that can run either from the command prompt or as a service. SQLDiag can be used to gather:

- Windows Performance logs
- Windows Event logs
- SQL Server Profiler traces
- SQL Server blocking Information
- SQL Server Configuration Information

To analyze the SQLDIAG output, you can also use the PAL utility. See http://msdn.microsoft.com/en-us/library/ms162833.aspx for more information about SQLDiag.

Troubleshooting with PAL tool

When you have a performance issue and are not sure about the performance counters to be gathered or how to analyze the same, you must use the PAL tool (Performance Analysis of Logs). The tool reads and analyzes a performance monitor counter log using the known thresholds. The features of the tool:

- Thresholds for most of the major Microsoft products such as IIS, SQL Server, and so on.
- An easy to use GUI interface which enables creating batch files for the PAL.ps1 script.
- A GUI editor for creating or editing your own threshold files.
- An HTML based report which is easy to copy to other applications.

- Analyzes performance counter logs for thresholds using thresholds that change the criteria based on the computer's role or hardware specification.

See https://github.com/clinthuffman/PAL.

Performance monitor

To monitor the utilization of system resources, use the Performance Monitor (perfmon). Collect and view real-time performance data in the form of counters, for server resources such as processor and memory use, and for many SQL Server resources such as locks and transactions.

The most useful SQL Server resources in the Performance monitor:

- SQLServer: Access Methods.
- SQLServer: Buffer Manager.
- SQLServer: Cursor Manager by Type.
- SQLServer: Cursor Manager Total.
- SQLServer: Databases.
- SQLServer: General Statistics.
- SQLServer: Latches.
- SQLServer: Locks.
- SQLServer: Memory Manager.
- SQLServer: SQL Statistics.
- SQLServer: Wait Statistics.

Server-Side traces

Server-Side tracing is performed using stored procedures. The SQL Profiler is a powerful tool, but the graphical UI that displays all the captured events, utilizes most of the CPU resources. For long running traces on loaded SQL Servers systems, Server-Side traces are a good solution.

The stored procedures used for these tasks are documented. However, the easiest way to create scripts that utilize these traces is to create a trace using the SQL Profiler graphical UI, and selecting Export on the File menu..

The system stored procedures to be used when creating and managing a trace:

- sp_trace_create
- sp_trace_generateevent
- sp_trace_setevent
- sp_trace_setfilter

- sp_trace_setstatus

SQL Nexus

SQL Nexus is a tool that helps you identify the root cause of SQL Server performance issues. The tool loads and analyzes performance data collected by SQLDiag. The amount of time you spend to analyze the data manually is reduced significantly.

Before working with the SQL Nexus tool, it is recommended that you gain knowledge of other performance topics and tools of the SQL Server.

SQL Nexus features:

- Fast, easy data loading: You can quickly and easily load SQL Trace files; T-SQL script output, including SQL DMV queries; and Performance Monitor logs into a SQL Server database for analysis. All the three use bulk load APIs to insert data quickly. You can also create your own importer for a custom file type.
- View loaded data using reports: After the data is loaded, you can generate several different charts and reports to analyze the data.
- Trace aggregation to display the TOP N most expensive queries (using ReadTrace)
- Wait stats analysis to view blocking and other resource contention issues (based on SQL Perf Stats)
- Full-featured reporting engine: SQL Nexus uses the SQL Server Reporting Services (RS) client-side report viewer (this does not require an RS instance). You can create reports for Nexus from either the RS report designer or the Visual Studio report designer. You can also modify the reports that are included with Nexus using either of the report designers. You can Zoom in/Zoom out to view server performance during a specific time window. Expand/collapse report regions (subreports) for easier navigation of complex data. Export or e-mail reports directly from SQL Nexus. Nexus supports exporting to Excel, PDF, and several other formats.
- Extensibility: You can use the existing importers to load the output from any DMV query to a table, and the RS reports you add to the Reports folder are automatically displayed in the reports task pane. If required, you can add a new data importer for a new data type. SQL Nexus can automatically fix the database references in your reports to reference the current server and database, and provide generic parameter prompt for the parameters that your reports support.

See http://sqlnexus.codeplex.com.

Third party utilities

- SentryOne SQL Sentry
  https://www.sentryone.com

- Red-gate SQL monitor
  http://www.red-gate.com/products/dba/sql-monitor
- Idera SQL diagnostic manager:
  https://www.idera.com/productssolutions/sqlserver/sqldiagnosticmanager
- Solarwinds Server & Application monitoring
  http://go.solarwinds.com/en/sam/sem/sql-server-performance-monitor
- Quest Foglight for SQL Server
  https://www.quest.com/products/foglight-for-sql-server
- EG SQL Server monitoring
  https://www.eginnovations.com/product/sql-server-monitoring
- Ola Hallengren SQL Server Maintenance Solution
  https://ola.hallengren.com
  https://github.com/olahallengren/sql-server-maintenance-solution
- Brent Ozar First Responder Kit
  https://www.brentozar.com/first-aid
  https://github.com/BrentOzarULTD/SQL-Server-First-Responder-Kit