Infor LN Performance Tracing and Tuning Guide

Publication Information

Release: Infor LN 10.x

Publication date: October 2, 2017

Document code: U9357G US

Contents

About this guide

This document provides guidelines to improve the Infor LN performance by tracing and tuning the environment. The chapters in this guide detail the processes to improve the operating system and the Infor LN application, not the database.

For database, specific tuning and tracing information refer to the database specific performance, tuning and tracing guides listed in the Related documents section.

These preconditions apply to this document:

- All information is based on the use of the Infor LN software. If you require information about other versions, you must refer to the relevant documentation.
- All information is based on the use of supported versions of the Infor LN, Operating System, and Database software.
- The database-related information in this document is described in detail in the database specific guides.

Note: This document is a comprehensive compilation. However, there may be instances wherein relevant information or procedures may have been omitted. Therefore, we recommend verifying the proposed changes in a test environment before moving to production. The information provided may not hold true for future versions of the operating system, database and application software.

Intended audience

This document is intended for intermediate to expert Infor LN and database Administrators and Technical Consultants to get optimal performance from an Infor LN system.

Organization

The chapters in this guide are:

Section	Description
Chapter 1, Introduction	Introduces the scope of this document.

Section	Description
Chapter 2, The Infor LN Architecture	Describes how the Infor LN application can be traced using the standard Infor LN tools.
Chapter 3, System Configuration and Tuning	How to configure a UNIX, Linux, or Windows system for optimal Infor LN Performance.
Chapter 4, Tuning Infor LN	Discusses Infor LN optimization parameters.
Chapter 5, Batch Performance Optimization	Explains the special settings to be considered for running Infor LN batches.
Chapter 6, Tracing Infor LN	How to identify performance problems within Infor LN.
Chapter 7, Performance Wizard	Helps to find the bottlenecks and how to solve these issues.

Related documents

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

- *Infor LN - Performance, Tracing and Tuning Guide for DB2 (B0077 US)*
- *Infor LN - Performance, Tracing and Tuning Guide for Oracle (B0078 US)*
- *Infor LN - Performance, Tracing and Tuning Guide for SQL Server (B0079 US)*
- *Infor LN - Data compression (B0050 US)*
- *Infor Enterprise Server - Technical Reference Guide for DB2 Database Driver (U7829 US)*
- *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US)*
- *Infor Enterprise Server - Technical Reference Guide for Microsoft SQL Server Database Driver (U8173 US)*
- *Infor Enterprise Server - Technical Manual (U8172 US)*
- *Infor Enterprise Server - Administrator's Guide (U8854 US)*
- *Infor LN – Deployment in a Virtualized Environment (B0073 US)*
- *Infor LN - Performance Guidelines (U9502 US)*
- *Infor LN - Sizing guide (B0045 US)*
- *Infor LN - Installation Guide (U9498 US)*
- *Infor ION - Sizing and Deployment Guide (B0074 US)*

You can find the documents in the product documentation section of the Infor Xtreme Support portal, as described in 'Contacting Infor' on page 12 or navigate to https://docs.infor.com/ln.

Related Knowledgebase articles

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

- *22881401 - Generic Infor LN Performance solution*
- *1183466 - Platform Support Matrix*
- *1576261 - Infor LN Hardware Selection - Important Notes*
- *22923520 - Latest Porting Sets Infor LN, Infor Baan5 and Infor Baan4*
- *1660883 - Investigate locking issues - error 107 / 201*
- *1463188 - Impact of 3-Tier on Infor LN performance, 2-Tier versus 3-Tier*
- *950226 - Examples of Call Graph Profiler commands*
- *1512083 - Infor LN Performance session ttsit0100m000*
- *1916664 - Infor Performance Analysis Tool (IPAT)*

You can find these KB articles in the InforXtreme Support portal, as described in ‘Contacting Infor’ on page 12.

Abbreviations and terminology

Abbreviation	Description
OLTP	Online Transaction Processing. A normal user behavior.
Batch	Applications that run to process data without user interaction.
CGP	Call Graph Profiler. The Infor LN call graph profiling tool.
DDL	Data Definition Language, such as CREATE and DROP TABLE commands.
DML	Data Manipulation Language, such as INSERTS, UPDATE, and DELETE queries.
DLL	Dynamic Load Library. A library is loaded in the application when required.
Named or licensed user	A user who can potentially log on to the Infor application
Connected or logged-on user	A user who is logged on to the Infor application
Active or concurrent user	A connected user who is actively using the Infor application
2-Tier Server	System that runs both the applications from the application server and the database server
3-Tier Application Server	System that runs the bshell (Virtual Machine) and the database driver
3-Tier Database Server	System running the database

Abbreviation	Description
DS	DS protocol, used for communication between the Infor Web UI Web server and the Infor LN server
JVM	Java Virtual Machine
ION	Integrated Open Network platform
DAS	Direct Attached Storage
SAN	Storage Attached Network
IOPS	Input/Output Operations Per Second

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 Introduction

1

After the installation of the Infor LN software, we recommend that you start a continuous process of monitoring and improving the environment. Infor LN strives for optimal default performance settings, but tuning is still required for optimal performance. It is still not possible to expect an optimal performing Infor LN application without any performance knowledge. Optimization starts during the installation, and continues when users are experienced with the processes, and expect more from the system when the database increases and functionality is added. This document describes how bottlenecks can be identified and the total performance can be improved.

Note: Most of the recommendations provided are proven by high-end benchmarks. Customers using only the default settings must consider the administrative overhead of deviating from these settings, as not all changes increase the performance in minor Infor LN implementations.

An application's performance is based on these factors:

- Performance

  The speed of the application, which is usually measured in:

  - Response time (OLTP)
  - Run time (batch)
  - Transactions per unit of time (batch).

  The performance aspect is important to end users as this aspect influences their day-to-day work.

- Effectiveness

  Effectiveness deals with the efficient use of the available recourses. To increase the speed, it is easier to add faster CPUs, additional memory, or faster disks, but the cost can be higher. Therefore, adding hardware does not always solve a performance problem. In case of a SQL query not performing as expected, adding more or faster CPUs might help, but it is more effective to change the query or query plan.

- Scalability

  This aspect used to measure the impact when:

  - More users are added: Will the increase in the number of users from 100 to 200, impact the performance? In this situation, locking can be an issue.
  - Data is added: Can the user process 200 orders per minute?
  - Scale-in: What happens when CPUs are added?
  - Scale-out: What happens when more systems are used in parallel?

This document covers these scenarios and helps to find a solution for better performance.

Chapter 2 The Infor LN Architecture

2

To create an optimum performing Infor LN environment, it is important to understand the Infor LN architecture. This chapter provides an overview of the important components.

The components in the architecture model:

- The user interface (UI) is Infor Ming.le and LN UI interface to connect to the Infor LN environment.
- The Infor LN application, which consists of:
  - A platform dependent layer called portingset. The main component of the portingset is the virtual machine, also known as bshell which runs the platform independent applications/sessions. A bshell process is started for each user. The bshell loads the database driver. The database driver translates the Infor LN queries to database native queries. The database driver can also run as a separate process.
  - A platform independent layer with the application sessions. This layer exists of the Enterprise Server with the 4GL engine and administration sessions. In addition, the Infor LN application also executes the business logic.
- The database is used for storage of data and must be an Infor supported database for Infor LN.

A model of the Infor LN architecture:

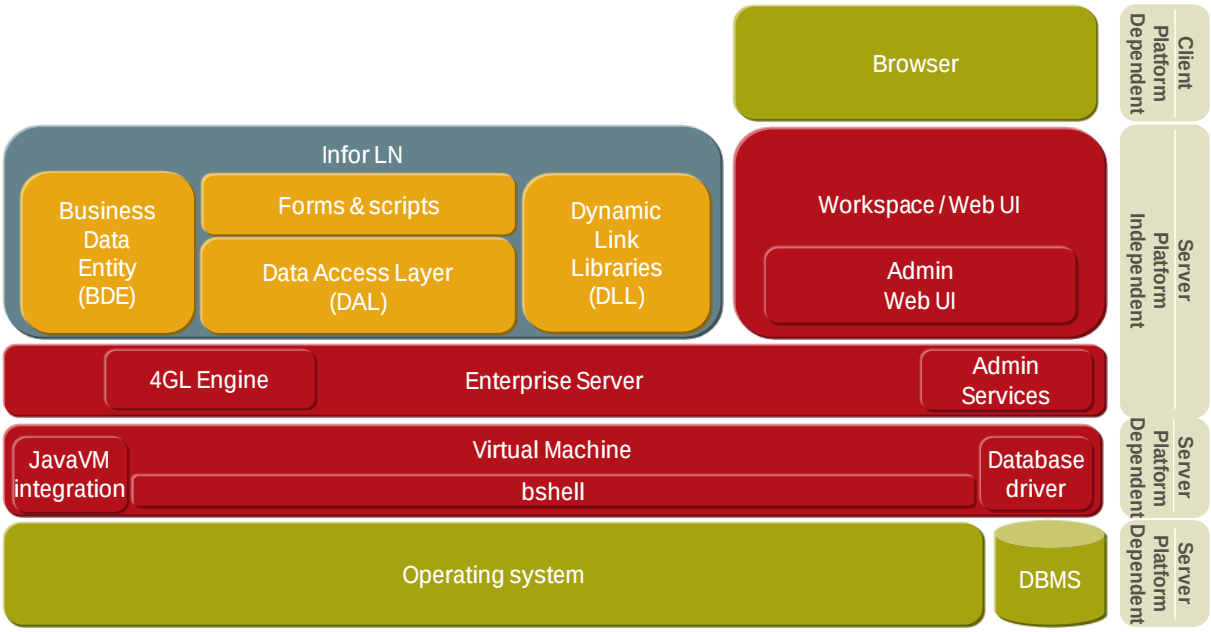


Figure 2-1 Infor LN architecture

If an application must interact with the database, an application command is sent to the database driver. The driver translates the command into a database-specific command and sends the command to the database. Results retrieved from the database are translated to data that can be read by the bshell.

This architecture has these characteristics:

- Multiple sessions can be started in one bshell, but only one session can be active. Time sharing is used to allocate the required processing power for each session.
- The communication between the different components is synchronous, which means that only the bshell, database driver, or database can be active at a given time.

The Infor LN bshell is based on a single threaded architecture. Therefore, only one session must be run simultaneously in one bshell. More than one session can be started simultaneously, but only one session at a time can be processed.

See *Infor LN - Sizing Guide (B0045 US)* for more detailed information about the Infor LN architecture.

Deployment

The components discussed in the Infor LN Architecture section are called tiers and can be deployed on separate servers. The deployment is selected based on the performance, security, and functional requirements. See *Infor Ming.le with Infor LN UI - Sizing and Deployment Guide (B0032 US)* for more information on deployment scenarios.

You can run the Infor LN application and database on the same server and the UI tier on separate servers (2-tier scenario). It is also possible to run every tier on separate servers (3-tier scenario).

Example of the 2-tier deployment scenario:

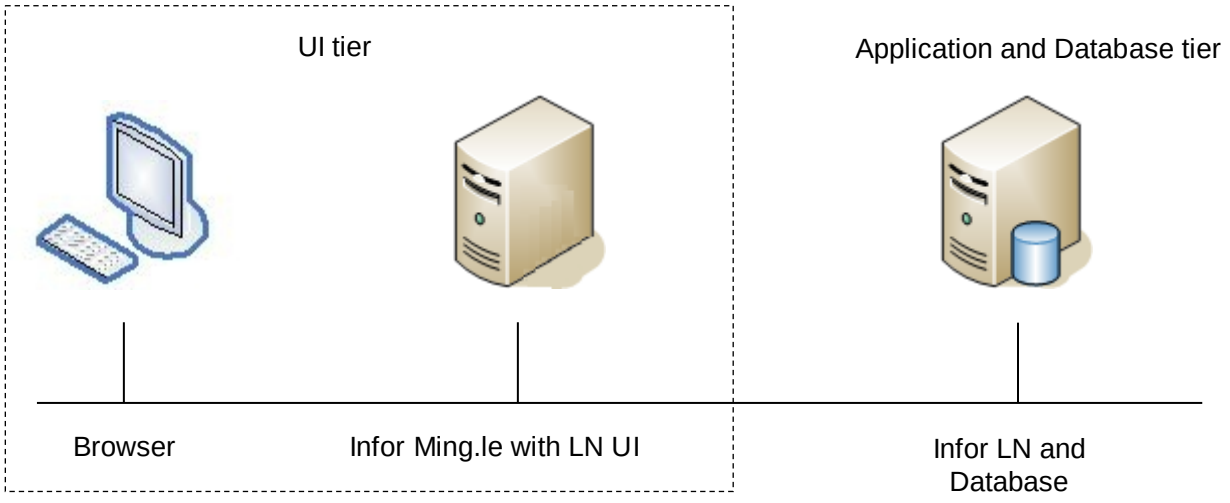


Figure 2-2 Infor LN 2-tier deployment scenario

Example of the 3-tier deployment scenario:

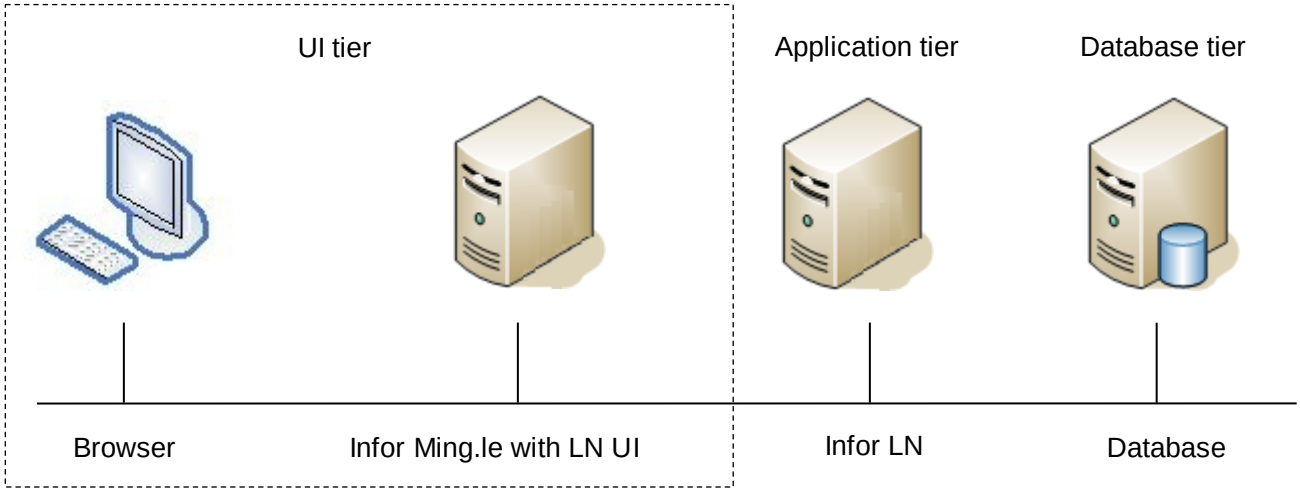


Figure 2-3 Infor LN 3-tier deployment

Chapter 3 System Configuration and Tuning

3

Before you install Infor LN, you must check the OS configuration and kernel and the required changes must be made to avoid performance issues. This chapter explains the OS configuration settings and kernel parameters for various platforms.

General

Some of the generic performance related information applicable for all operating systems and databases is provided in the following topics.

Hardware selection

For a high-performance Infor LN system, you must purchase hardware that suits your business requirements. Extensive rules for hardware selection are not described in this section. See *Infor LN - Sizing whitepaper (B0050 US)* and InforXtreme knowledge base article *1576261* for more information or consider sizing by Infor.

I/O Setup

The key to get an Infor LN system that performs at optimum; is the correct I/O setup, which is valid for any type of I/O setup, such as Direct Attached Storage (DAS) and Storage Attached Network (SAN) solutions. Currently, the performance of the storage subsystem not only depends on the number of disk spindles, but also advanced technologies such as all-flash, flash-based read caching, by combining SSD's and SAS drives into the same SAN, optimize data patterns to prevent random write access, and compression techniques.

For the Infor LN application, it is important that the total number of IOPS requested can be managed by the storage subsystem at lowest possible latency. Refer to the sizing for the required number of Input/Output Operations Per Second (IOPS) topic.

These guidelines help design and implement a correct I/O setup:

- For smaller implementations that use a DAS solution, you must configure sufficient drives to ensure that the transaction log volume is less than 225 IOPS per disk and the data and/or index

volumes is less than 85 IOPS per disk. Regularly monitor the number of IOPS per disk. If disks are nearing the capacity limit, add more disk drives.

- For optimal performance, the I/O subsystems must be configured in RAID 1, RAID 0+1, or RAID 10 volumes. When performance of the database data files is less important, RAID 5 or RAID 6 can also be used.
- For larger implementations (200+ Concurrent users) or systems with more than 10 disk spindles, we recommend isolating the database transaction log onto a separate RAID 10 volume. I/Os to the transaction log are mostly written sequentially. For optimal performance, configure transaction log I/O cache such as cache in SCSI Controller, to 100 percent write. Do not use this volume for any other purpose.
- The database temp table space is not used extensively in an Infor LN environment. However, for larger implementations, or for other applications using the Infor LN database, we recommend you allocate a specific volume to the temp table space.
- You must keep the disk fragmentation as low as possible on the database disks.
- Validate the performance of the storage subsystem before installing the software to ensure that the disk subsystem can deliver the required number of IOPS. There are third party utilities like DiskSpd and IOmeter available for this benchmark.

Storage setup

A good storage setup is important for performance. The following guidelines help in the designing and implementation of the storage setup, and are applicable for every database:

- The Infor LN database must be created with the appropriate size. If 150 GB is the requirement for a year, you create the database as per the required size. Do not let the database expand with small chunks and cause additional fragmentation in the database and the files on disk. If autogrowth size is required, use bigger chunks for the data, such as 2 GB, to limit the number of extents.
- Raw devices are recommended for best performance, but are difficult to manage. However, a correct implementation of Direct I/O and Concurrent I/O ensures a 97% performance of the raw devices.

For database specific storage recommendations, see the database specific Performance, Tracing, and Tuning Guide.

Mount options

To get the optimal performance on your UNIX system, it is important that the file systems are mounted with the correct mount options. The file systems and underlying volumes must be split based on their data access patterns:

- Segregate the file systems and volumes used for the database log writer (sequential writes) from the non-synchronous data.

- Segregate the file systems and volumes used for sequential read, write and random I/O as much as possible.

The recommended mount options for the different file types:

File type	Access type
Database Redo log and Archive log	Direct I/O
Database Data files	Concurrent I/O
Binaries	Cached I/O

Direct I/O

Direct I/O can be more efficient than using cached I/O and provides the benefit for writing the Database Redo log and archive log files.

Concurrent I/O

Concurrent I/O allows multiple processes to read from or write to the same file without blocking other read(2) or write(2) calls. This increases the performance, due to the huge decrease of JFS inode contention. Concurrent I/O performance range is 93-99% of raw logical volumes.

Cached I/O

Cached I/O uses the OS file system buffer cache, which is ideal for database and application binaries, as binaries do not change often and are kept in memory as much as possible. Cached I/O is enabled by default.

Binaries must not be mounted for direct I/O as these data blocks are accessed often. Using cached I/O on the database and application binaries eliminates I/Os that are not required.

Network setup

For a 3-tier setup, Infor LN uses a high number of relatively small network packets to communicate between the application server and database server. Therefore, the network performance between the different servers must be optimal. General guidelines for the network setup:

- The network connection between the application and database server must be a minimum of 1Gbps.
- Ensure that the physical distance between the application and database server is minimal with the least possible switches.
- Ensure there are no firewalls between the application and the database server.
- Network latency must be less than 1ms for every package.

- Package loss, disconnects, or retransmits must not occur on the network.

Large pages

Large pages are useful for large database shared memory (for example, Oracle SGA) sizes, and for systems with large amount of physical memory. Based on benchmark statistics, Infor LN runs at an optimum on a database that uses large pages. When using large pages, there are less virtual memory pages to address, which optimizes the use of Translation Lookaside Buffers (TLB), which results in less bookkeeping for the system.

Physical memory is partitioned into pages which are the basic units of memory management. When a process accesses a virtual address, the CPU must translate the same to a physical address. Therefore, for each process, the kernel maintains a page table that is used by the CPU to translate virtual addresses into physical addresses. Before the translation, the CPU performs several physical memory reads to retrieve the page table information. To speed up this translation process, the CPU saves information for recently accessed virtual addresses for future references, in the Translation Lookaside Buffers (TLB), a small but fast cache in the CPU. The use of this cache speeds up the virtual memory access. Since TLB misses are expensive, TLB hits can be improved by mapping large contiguous physical memory regions by a small number of pages. So, fewer TLB entries are required to cover larger virtual address ranges. A reduced page table size results in less memory management overhead.

Power-saving options

Power-saving technologies can have a huge impact on the overall system performance. So, it is recommended that you consider disabling these power-saving options, when the performance requirements outweigh the energy requirements. For Intel based systems, consider the following:

- Enable the Turbo Mode in the BIOS, if available.
- Ensure all CPU cores and sockets are enabled.
- When running a virtualized Infor LN environment, enable Virtualization support. This includes Intel VT-x and Intel EPT for Intel, and AMD-V and AMD RVI for AMD.
- Enable Hyper-threading, if available. This can increase overall performance by 20% approximately.
  Note: The database vendors take every virtual CPU as a real CPU core for licensing. In that case, it can be beneficial to disable Hyper-threading and only assign real cores to a virtual machine.
- Disable node interleaving to enable the NUMA architecture of the system, if available.
- Enable the *maximum performance* or the *high performance* power plan mode in the BIOS, on Hypervisor level and on Operating System level.
- Disable the C1E halt state and any other power saving mode.

See https://docs.microsoft.com/en-us/windows-server/administration/performance-tuning/hardware/power, for Server hardware power considerations.

Hypervisor

Running Infor LN in a virtual environment requires additional tuning which is described in *Infor LN Deployment in a Virtual Environment (B0073)*.

IBM AIX

These parameters can be changed to optimize the performance of the Infor LN application running on IBM AIX.

Large pages

AIX provides 4 KB and 16 MB virtual memory pages. Using the vmo command, you can specify the amount of 16 MB memory pages. Starting with AIX 5.3 version, the large page pool is dynamic, so the amount of physical memory specified does not require a system reboot. The additional memory is stored on the 4 KB pages.

A security access control mechanism prevents unauthorized applications from using large pages or large page physical memory. The security access control mechanism also prevents unauthorized users from using large pages for their applications. For non-root user ids, such as oracle or db2inst1 database owners, you must enable the **CAP_BYPASS_RAC_VMM** and **CAP_PROPAGATE** capability with the *chuser* command to use large pages.

Configure the AIX large page pool by calculating the number of large pages required for the database shared memory (for example Oracle SGA):

```
num_of_large_pages = INT(total_db_mem_size GB / Large page size) + 1
```

If the database shared memory is 20 GB, use 20 GB Large pages + 1 extra page. For example:

```
num_of_large_pages = INT(21474836480) / 16777216) + 1 = 1281
```

To allocate 20 GB of RAM as large pages (16 MB):

1. Run as root user and use these commands to reserve 20 GB of a large page:

   ```
   # vmo –p –o lgpg_size=16777216 –o lgpg_regions=1281
   # vmo -p -o v_pinshm=1
   ```

2. Reboot the system.

3. As root user, add these capabilities for the user:

   ```
   # chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE <user id>
   ```

4. Validate large page support using this command:

   ```
   # vmstat –P ALL

   System configuration: mem=112640MB
   pgsz            memory                           page
   ```

```
----- -------------------------- ----------------------------------
           siz       avm       fre    re    pi    po    fr     sr    cy
    4K 23515136 13955711   9307982     0     0     0     0      0     0
   16M     1281       785       496     0     0     0     0      0     0
```

For more information, see http://www-01.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.performance/large_page_ovw.htm?lang=en.

Mount options

The recommendation for the AIX mount option for the different file types:

File type	Access type	Mount options
Database Redo log and Archive log	Direct I/O	dio (JFS2 + agblksize=512) For Archive Log Files, the rbrw mount option is recommended.
Database Data files	Concurrent I/O	cio (JFS2 + agblksize=<DB block size>)
Binaries	Cached I/O	Optional: noatime

To check the mount options in use, use the following command:

```
# cat /etc/filesystems
```

Note 1: I/O requests made by Oracle must be aligned with the JFS2 block size to avoid a demoted I/O (Return to normal I/O after a Direct I/O Failure), when using DIO/CIO.

If the block size is >=4096, use a file system block size of 4096, else use 2048.

Oracle Redo logs are written in 512B blocks. Therefore, the JFS2 block size must be set to 512:

```
# mkfs –o agblksize=512 <filesystem>
```

Note 2: When using the **rbrw** mount option, the file system is mounted with both release-behind-when-reading and release-behind-when-writing capabilities. When the sequential reading of a file in this file system is detected, the real memory pages used by the file are released when the pages are copied to the internal buffers. When sequential writing of a file is detected in this file system, the real memory pages used by the file are released after the pages are written to the disk. This is recommended for database redo log and archive log file systems. For more information, see http://www-01.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.performance/multiple_page_size_support.htm

Note 3: Use the Concurrent I/O (cio), in combination with DB2, when running on AIX 5.3 or higher.

By default, the database manager prevents the caching of DB2 data, except for the temporary data and LOBs on AIX, by invalidating the pages from the cache. For more information, see: http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0051304.html

Note 4: Do not use the Concurrent I/O (cio) mount option, in combination with Oracle 11.2.0.2 or higher, when running on AIX 6.1 or higher.

Starting with Oracle 11.2.0.2, the O_CIOR flag option to open files on a JFS2 filesystem is used. Therefore, you must no longer mount the filesystems with mount–o cio option.

Direct I/O

To enable the direct I/O, use the following command:

```
# mount –o dio <mount_point>
```

Concurrent I/O

To enable concurrent I/O, use the following command:

```
# mount –o cio <mount_point>
```

Cached I/O

Specific mount options are not required for file systems using the cached I/O. However, it is possible to specify the *noatime* option, to turn off access-time updates. Use this option to improve performance on file systems, wherein many files are read frequently but are not updated often. If you use this option, the last access time for a file cannot be determined.

```
# mount –o noatime <mount_point>
```

Network

To optimize the network performance for Infor LN, especially in a 3-tier environment, the following parameters must be modified:

- The *tcp_recvspace* tunable specifies the number of bytes of data the receiving system can buffer in the kernel on the receiving sockets queue. The *tcp_sendspace* tunable specifies the amount of data the application can buffer in the kernel, before the application is blocked on a send call. Set *tcp_recvspace* and *tcp_sendspace* to larger values for a network of 1 Gb or higher:

  ```
  # no -p -o tcp_recvspace=262144
  # no -p -o tcp_sendspace=262144
  ```

- The *tcp_nodelayack* tunable is used to ensure that an acknowledgement is not delayed. This can reduce latency and allow the sender to receive the acknowledgement, and send the next partial segment sooner.
  To turn off delaying the acknowledgement, use this command:

  ```
  # no -p -o tcp_nodelayack=1
  ```

- The rfc1323 tunable enables the TCP window scaling option. Use this command:

  ```
  # no -p -o rfc1323=1
  ```

Restart the system to implement these settings.

Thread tuning

Setting these options influences the kernel thread behavior:

- The *AIXTHREAD_MNRATIO* variable controls the scaling factor of the library. Modifying this variable enhances the efficiency of the kernel thread support by setting one kernel thread for each user thread.

  This ratio is used when creating and terminating pthreads. This can be useful for applications with a very large number of threads. By default, one kernel thread serves 8 user threads. (8:1). Infor LN is a single threaded application; therefore, a ratio of 1:1 is beneficial.

  To permanently set this variable, add the following line to /etc/environment and restart the system:

  ```
  AIXTHREAD_MNRATIO=1:1
  ```

- The *AIXTHREAD_SCOPE* variable controls the AIX thread contention scope. The P option signifies a process-wide contention scope (M:N) while the S option signifies a system wide contention scope (1:1). One of these options must be specified; the default is P.

  If a user thread is created with a system-wide scope (S), the same is linked to a kernel thread and is also scheduled by the kernel. The kernel thread is not shared with another user thread.

  If a user thread is created with a process-wide scope (P), the same is controlled by the user scheduler. Therefore, the user thread:

  - does not have a dedicated kernel thread.
  - is set to the sleep mode when in the user mode.
  - is placed on the user run queue when waiting for a processor.
  - is subjected to time slicing by the user scheduler.

  Based on the benchmarks, user threads created with a system wide scope are beneficial for Infor LN. To permanently set this variable, add this line to /etc/environment and restart the system:

  ```
  AIXTHREAD_SCOPE=S
  ```

Fiber channel parameters

To support high disk I/O, some fiber channel adapter parameters can be modified. To obtain a list of the current hard disk parameters, enter this command:

```
lsattr -El fcsX
```

Modify the value of the following parameters:

- The *lg_term_dma* parameter controls the memory to store I/O commands and the related data. This can be increased to 8 KB using this command:

```
# chdev –l fcsX –a lg_term_dma=0x800000
```

- The *max_xfer_size* parameter controls the maximum I/O that can be issued. This can be increased to 2 KB using this command:

```
# chdev –l fcsX –a max_xfer_size=0x200000
```

- The *num_cmd_elem* parameter controls the maximum number of simultaneous I/Os the fiber channel adapter can process. This can be increased to 400 using this command:

```
# chdev –l fcsX –a num_cmd_elem=400
```

Hard disk parameters

On high-end systems, it is recommended to make certain changes to the hard disk parameters. To obtain a list of the current hard disk parameters, enter this command:

```
lsattr -El hdiskX
```

Modify the value of every hard disk in the same volume group. The volume group must to be varied off, before applying the changes:

- The *queue_depth* parameter controls the maximum number of I/O commands that can be queued on each hdisk (Default is 20). This can be increased to 40 using this command:

```
# chdev –l hdiskX –a queue_depth=40
```

- The *max_transfer* parameter controls the maximum disk I/O on each hdisk. This can be increased to 4 MB using this command:

```
# chdev –l hdiskX –a max_transfer=0x400000
```

- The *algorithm* and *reserve_policy* parameters are set to support the multipath I/O (MPIO) using the AIX MPIO capabilities. MPIO provides an efficient I/O and reduces path redundancy.

```
# chdev –l hdiskX –a algorithm=round_robin
# chdev –l hdiskX –a reserve_policy=no_reserve
```

Additional information about AIX performance

For more information, see these documents:

- The AIX 7.1 knowledge center
- http://www-01.ibm.com/support/knowledgecenter/ssw_aix_71
- Optimizing AIX 7 operating system performance https://www.ibm.com/developerworks/aix/library/au-aix7optimize1

Hewlett Packard – HPUX – IA64

These parameters can be modified to optimize the performance of the Infor LN application, running on HP-UX 11.31 and higher.

Enable hyper threading

By default, for HP-UX based systems, the hyper threading feature and the default Processor Set's LCPU (Logical processor) attribute is disabled. Hyper threading helps in improving the overall performance of Infor LN. It is recommended that you enable the feature.

To enable the LCPU attribute (HT) for the default Processor Set (PSET), use the following command:

```
# kctune lcpu_attr=1
```

File-system tuning

Performance of the application and database is I/O dependent; it is important that the file system is configured optimally. Following are some guidelines to tune the file system.

Mount options

The recommended HP-UX mount options for the different file types:

File type	Access type	Mount options
Database Redo log and Archive log	Direct I/O	delaylog,mincache=direct,convosync=direct
Database Data files	Concurrent I/O	delaylog,cio
Binaries	Cached I/O	delaylog,nodatainlog

To check which mount options are currently used, use this command:

```
# cat /etc/fstab
```

Direct I/O

To enable direct I/O, use this command:

```
# mount -F vxfs –o delaylog,mincache=direct,convosync=direct <device_special_file>
<mount_point>
```

Concurrent I/O

To enable concurrent I/O, use the licensed VxFS CIO mount option:

```
# mount -F vxfs -o delaylog,cio <device_special_file> <mount_point>
```

Note:

- The remount command/option must not be used, when using the ‘cio’ mount option.
- Do not use ‘cio’ and ‘mincache=direct, convosync=direct’ together as this can cause degradation. Use either the Direct I/O or the Concurrent I/O options.

Cached I/O

Use this mount options for cached I/O:

```
# mount -F vxfs –o delaylog,nodatainlog <device_special_file> <mount_point>
```

Kernel parameters

The following HP-UX kernel parameters are important for the performance of Infor LN. If the recommended value is lower than the default value, the default value must be retained.

Tune inode cache

Tune a static VxFS inode cache to decrease CPU usage:

```
# kctune vxfs_ifree_timelag=-1
# kctune -h vx_ninode=64000
```

FILECACHE_MIN / FILECACHE_MAX

Specifies the minimum and maximum percentage of memory used for file system buffer pages. These parameters are used for a lower and upper boundary for the buffer pages.

A minimum size of 2% and maximum of 10% of the total internal memory for the file system cache is recommended:

```
# kctune filecache_min=2%
# kctune filecache_max=10%
```

NPROC

The NPROC parameter specifies the maximum number of processes running simultaneously. This formula can be used to determine the correct value for this parameter:

2-tier	3-tier application server	3-tier database server
6 * #connected users + 512	4 * #connected users + 512	2 * #connected users + 512

The variable '#connected users' specifies the maximum number of connected users on the system. These users are not active, but utilize the system resources. It is assumed that each end user accesses the system, at least once. You must Increase the number of connected users with extra logins for each end user.

This formula is based on these assumptions:

- Each connection contains a bshell, (includes a database driver for the combination mode), an audit driver, a database backend process, and one additional process. In case of exceptional circumstances, the number of processes must be increased.
- Each user connects to Infor LN using the Infor LN UI.
- Each user runs one database driver.

Note: The 512 additional processes are based on a common environment with a database. In case of additional products, the total amount of processes must be increased.

When using additional database drivers to access multiple database instances simultaneously, the formula is:

2-tier	3-tier application server	3-tier database server
2 * #additional database drivers * # connected users	#additional database drivers * # connected users	#additional database drivers * # connected users

When a UNIX error 11 'No more processes' occurs, the value of NPROC must be increased. The current and maximum number of processes can be found in the column proc-sz, from the **sar -v** output.

Note: On some UNIX systems, the following additional command is started using the inetd:

```
sh –c /.../ipc_boot6.2/...
```

To prevent this, use a Korn shell **(/usr/bin/ksh)** as a login shell in **/etc/passwd**, instead of a Bourne shell **(/bin/sh)**.

MAXFILES

This parameter specifies the maximum number of files that can be opened per process. Infor LN does not use numerous files simultaneously. The recommended value is:

```
MAXFILES >= 256
```

The number of this parameter must be increased when an error 24 is displayed: *Too many open files.*

SHMMAX

SHMMAX specifies the maximum size of a shared memory segment that can be created in bytes. Several shared memory segments of this size can be created per process in an environment.

It is recommended that you set the SHMMAX value equal to the size of the internal memory. For example (32 GB):

```
SHMMAX = 34359738368
```

In case a database also runs on the same system, use the value recommended for the database vendor. Usually, this is in the range of 60% to 100% of the internal memory.

In case of an error 22, you must increase this value.

Messages

Messages can be used for communication between processes. However, communication between processes by message queues is not the optimal performing method. Therefore, skip configuring message queues in the kernel. If message queues (m) are used in the $BSE/lib/ipc_info file, change the messages to pipes (p).

MSGMNB

This parameter specifies the maximum length, in bytes, of a message queue. The maximum number of bytes in a message queue is 262144. Therefore, the minimum value of the *MSGMNB* must be:

```
MSGMNB >= 262144
```

Check the number of bytes in a message queue using the **ipcs –qo** command:

MSGMNI

Specifies the maximum number of message queues that can be used across the system.

The number of message queues required for Infor LN is relatively less. *MSGMNI* must be 64 or more.

```
MSGMNI = 64
```

Increase the number when an error 28 occurs.

MSGTQL

Specifies the maximum number of message headers that is the maximum number of open messages.

This parameter must be equal to *MSGMNI*:

```
MSGTQL = MSGMNI
```

MAXUPRC

This parameter describes the maximum number of concurrent processes per user id. The value depends on the configuration. Usually, the default value is recommended. However, there are some exceptions when this parameter must be increased:

- Multiple users run under the same user id.

- If **BSE_REM** is used, customers prefer to run all users as one remote user, such as the user **bsp**. In that case, the number of processes per user must be increased.
  Note: The use of BSE_REM is not recommended for performance reasons.
- On a 3-tier database server, all processes are run with the process ID of the database owner on several databases. Therefore, you must increase the number of processes per user.

If any of these scenarios occur, calculate the maximum number of required processes per user and tune the parameter.

When running out of processes, increase the value of *NPROC*, or lower the value of *MAXUPRC*, if possible.

Additional information about HP-UX performance

For additional information, search:

- HP-UX VxFS mount options for Oracle Database environments
- Common Misconfigured HP-UX Resources
  Veritas™ File System Administrator's Guide (HP-UX 11i v3)
- Performance Improvements using Concurrent I/O on HP-UX 11i v3 with OnlineJFS and the HP-UX Logical Volume Manager

Oracle Solaris

These parameters can be modified to optimize the performance of the Infor LN application running on Oracle Solaris. The mentioned settings are tested on Oracle Solaris 10.

Fixed-Priority (FX) Scheduling Class

The FX scheduler provides a scheduling policy for processes that require user or application control for scheduling priorities. The priorities of processes that run under FX are fixed. These priorities are not dynamically adjusted by the system. Based on the benchmarks, Infor LN benefits from fixed priorities on the Application server or 2-tier server.

Use the dispadmin command to set or retrieve the scheduling class.

Multiple Page Size Support

To gain the maximum CPU performance for your Infor LN server, use the Multiple Page Size Support (MPSS) for the database. To set up the MPSS with an optimal configuration of a 4 MB heap size, set the following in the profile for the Oracle users; for example:

```
set LD_PRELOAD=$LD_PRELOAD:mpss.so.1
set MPSSHEAP=4M
export LD_PRELOAD MPSSHEAP
```

Intimate Shared Memory

On Solaris, Oracle can use the (Dynamic) Intimate Shared Memory (DISM). Dynamic ISM allows a database to dynamically extend or reduce the size of the shared data segment. This feature eliminates the misconfiguration and denial-of-service security vulnerability of Intimate Shared Memory.

The ISM is a shared memory segment that consists of large locked memory pages. The ISM number of locked pages remains constant or unchanged. Dynamic ISM is pageable shared memory, where the number of locked pages is variable or can be modified. Therefore, the DISM supports releasing or adding additional physical memory to the system during dynamic reconfiguration. The size of the DISM can span available physical memory plus a disk swap.

For more information about DISM see: http://www.oracle.com/technetwork/articles/systems-hardware-architecture/using-dynamic-intimate-memory-sparc-168402.pdf

Linux

This section describes the parameters applicable for SuSE Linux and Redhat kernel 2.6.

Mount options

To increase the filesystem I/O performance, the *noatime* and *nodiratime* mount options can be used. When using these mount options, Linux prevents an update of the access time, unless the access involves a modification of a file's or a directory's metadata or content. Avoiding the write I/O associated with updating the access time can result in measurable performance gains. If you use the option, the last access time for a file cannot be determined.

To check which mount options are currently used, use the following command:

```
# cat /etc/fstab
```

To disable the access time use the following mount option:

```
# mount -o noatime,nodiratime <mount point>
```

Connections

By default, a maximum of 32 connections, on Linux, can be made to the xinet deamon. When connecting more than 32 users, rexec login failures occur. To increase the number of connections, edit the /etc/xinetd.conf file. The ‘instances = 32’ must be replaced with, for example, ‘instances = 1000’. You must restart the xinetd, using ‘/etc/init.d/xinetd restart’:

```
instances = 1000
```

Kernel parameters

Kernel parameters are important to improve performance. To list the kernel parameters, use this command:

```
# sysctl -a
```

To set a parameter, use the ‘sysctl –w’ command. For example:

```
# sysctl -w kernel.shmmax=2147483648
```

To ensure that a modification is permanent, add a line to the file /etc/sysctl.conf in the format:

```
<parameter>=<value>
```

The file /etc/sysctl.conf is used during the boot process.

Swappiness

This parameter impacts the priority paging mechanism. Lowering the value, gives a process memory page priority over a file system buffer page to remain in the memory.

The threshold, when processes must be swapped, can be tuned using /proc/sys/vm/swappiness. The default value is 60, which is recommended if idle daemons or programs must swap out before the file system buffer cache. A higher value provides additional buffer/page cache. A lower value waits longer to swap out idle processes. It is recommended that you modify the value of a swappiness to 10, to retain the bshell processes in memory:

```
vm.swappiness=10
```

Read ahead buffer

To speed up streaming reads, increase the size of the ‘block read ahead’ buffer for fast storage (SCSI disks or RAID). The default is 128. It is recommended to increase the block read ahead buffer, for database data disks, to 512:

```
# echo 512 > /sys/block/<sdX/hdX>/queue/read_ahead_kb
```

Number of requests

The length of a request queue impacts the I/O performance. The default is 128. To increase the throughput, it is recommended to increase the number of requests to 256 with CFQ scheduler for fast storage:

```
echo 256 > /sys/block/<sdX/hdX>/queue/nr_requests
```

Huge Pages

To use larger page sizes for shared memory, Huge Pages must be enabled on Linux which also locks these pages in the physical memory. The default page size in Linux for x86-64 is 2MB.

Configure the Linux Huge Page pool by calculating the number of huge pages required for the database shared memory (e.g. Oracle SGA):

```
num_of_huge_pages = INT(total_db_mem_size / Huge page size) + 1
```

If the database shared memory is 12 GB, use 12 GB Large pages + 1 extra page. For example:

```
num_of_huge_pages = INT(12884901888) / 2097152) + 1 = 6145
```

The following steps allocate 12 GB of RAM as huge pages (2 MB):

1. Set the *vm.nr_hugepages* kernel parameter to a value higher than the value specified for the database shared memory. In this case, the database shared memory is 12 GB, so use 12 GB and you must set the parameter to 6145.

   ```
   vm.nr_hugepages=6145
   ```

2. The database userid must be able to lock a greater amount of memory. So, the file '/etc/security/limits.conf' must be updated to increase the soft and hard *memlock* values for the database userid. In case of an oracle database, add the following lines:

   ```
   oracle          soft    memlock        12582912
   oracle          hard    memlock        12582912
   ```

3. After this setup, you must ensure that the database shared memory (e.g. Oracle SGA) is using the Huge Pages. The value of ((Huge Pages_Total – Huge Pages_Free) * 2MB) is larger than the size of the database shared memory (or equals the shared memory segment displayed in the output of ipcs -ma).

   ```
   # grep -i huge /proc/meminfo
   HugePages_Total:  6145            #The size of the pool of HugePages.
   HugePages_Free:   1656            #The HugePages that are not yet allocated.
   HugePages_Rsvd:      0            #The number of HugePages for which a commitment
                                     # to allocate from the pool has been made, but
                                     # no allocation has yet been made.
   Hugepagesize:     2048 kB         #HugePage size
   ```

For more information, see Oracle support notes 361323.1 and 361468.1.

From RedHat6 and SLES11 onwards, transparent HugePages are implemented and enabled (default) to improve the memory management. Transparent Huge Pages differ from regular Huge Pages as the Transparent Huge Pages are set up dynamically at run time by the khugepaged thread in the kernel while the regular Huge Pages had to be preallocated at the boot up time.

Oracle does not recommend using transparent Huge Pages. See Oracle support note 1557478.1 and https://www.redhat.com/en/resources/deploying-oracle-database-12c-red-enterprise-linux-7-recommended-practices.

Caution: Some experts do NOT recommend using Automatic Shared Memory Management (AMM, for example setting *memory_target*) with Linux HugePages. See Oracle support note 749851.1.

SHMMAX

Shared memory allows processes to access common structures and data by placing the same in shared memory segments. This is the optimum method of Interprocess Communication (IPC) available, as no kernel involvement is required when the data is passed between the processes. No data will be copied between the processes.

To view the shared memory settings, run:

```
# ipcs -lm
```

The *SHMMAX* parameter defines the maximum size, in bytes, for a single shared memory segment that a Linux process can allocate in the virtual address space.

As the database shared memory size (for example, Oracle SGA) is comprised of shared memory, *SHMMAX* can limit the size of the database shared memory. *SHMMAX* must be larger than the database shared memory size to prevent scattering data. It is recommended that you use a *SHMMAX* size equal to the internal memory.

The shared memory must be accommodated in the Huge Pages pool, see 'Huge Pages' section. To determine the current maximum size of a shared memory segment, run:

```
# cat /proc/sys/kernel/shmmax
2147483648
```

Change the *SHMMAX* size to the size of the internal memory. For example:

```
kernel.shmmax=34359738368
```

SHMALL

The *SHMALL* parameter is used to specify the total amount of shared memory pages that can be used across the system. Therefore, *SHMALL* must be set to *SHMMAX* / *PAGE_SIZE* at mimimum. The page size can be determined using:

```
getconf PAGE_SIZE
```

When the *SHMMAX* parameter is set to 34359738368 and the page size is 8192, the *SHMALL* parameter is 4194304:

```
kernel.shmall=4194304
```

Microsoft Windows

This section describes the parameters that can be modified to optimize the performance of the Infor LN application on Microsoft Windows.

Anti-virus software

Based on the customer experience study, anti-virus software and other process intrusive tooling (for example, Microsoft System Center Operations Management (SCOM)) can have major impact on the performance of an Infor LN system. Therefore, it is recommended to exclude the Infor LN directories and processes from a continuous scanning process. The following processes must be excluded at minimum:

- %BSE%\bin\ntbshell.exe
- %BSE%\bin\sort.exe
- %BSE%\bin\audit_srv.exe
- %BSE%\bin\shmserv.exe

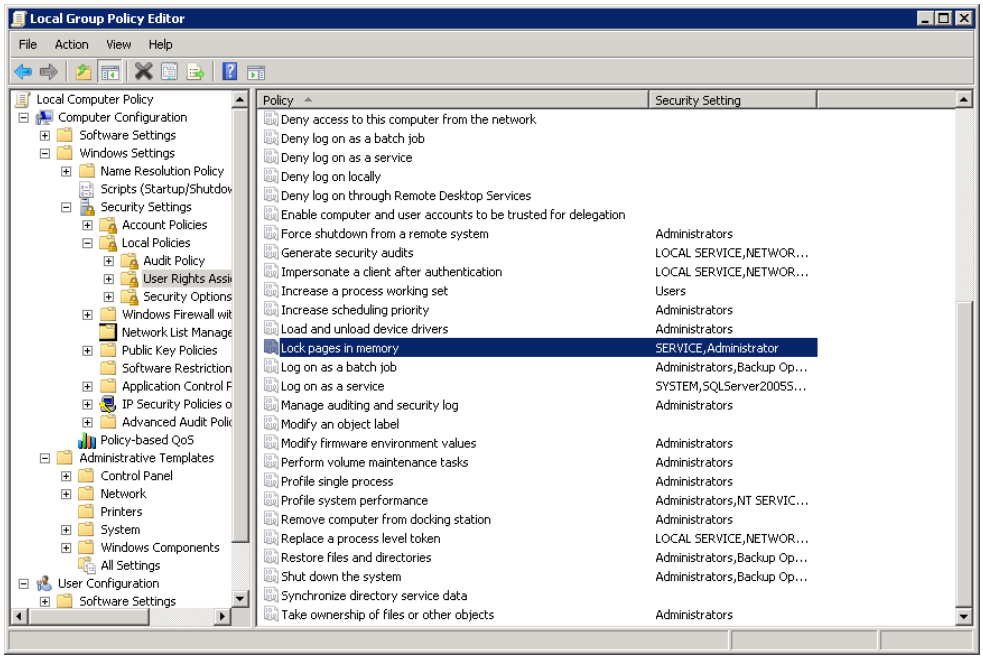
The Infor LN directory and files can be scanned during the scheduled maintenance window.

Large pages and Lock pages

To use large pages on Windows, assign the 'Lock pages in memory' privilege to a user that runs the database. This includes service accounts and administrators.

To assign privileges:

1. Select Control Panel -> Administrative Tools -> Local Security Policy or start gpedit.msc
2. Select Local Policies -> User Rights Assignment



3 Double-click Lock pages in memory

4 Add users and/or groups

5 Reboot the machine

Chapter 4 Tuning Infor LN

4

This chapter describes the most important settings and parameters for performance in Infor LN and the effect of various values. And, also includes the general deployment considerations.
The database dependent settings are described in the database specific Performance, Tracing, and Tuning Guides.

General

Before describing the specific Infor LN settings to optimize performance, it is recommended to validate general topics such as:

- Latest software
- Housekeeping
- Maintenance windows and batch windows

Latest software

Infor continually improves the related software by fixing bugs and implementing new features. Performance improvements are also made in the porting set, Enterprise Server, and application.

Porting set

The porting set consists of the executables, located in the $BSE/bin directory and shared libraries located in the $BSE/shlib. Apart from the bug fixes, performance enhancements are also made to the porting set, and new features are implemented. The latest porting set is compatible with the supported versions of the Infor LN application. Therefore, it is recommended to use a recent porting set. See Infor Xtreme solution 22923520 for the latest porting set. See Infor Xtreme solution 1183466 for the platform support matrix.

Enterprise Server

The Infor LN standard program maintains the functionality for standard 4GL functions; these functions are used in all Infor LN sessions. Performance improvements in the standard program have an impact on all sessions. Because the standard program is frequently updated, and the

performance improvements are regularly implemented it is recommended to use a recent version of the Enterprise Server, which includes the standard program.

Infor LN application

Based on the new developments and customer experience, Infor continuously solves performance bottlenecks in specific Infor LN application sessions. If you encounter a performance issue in a specific application session, it is recommended to check with Infor support if a fix is already available.

Housekeeping

For optimal performance of Infor LN, regular housekeeping is recommended. These activities must be performed.

Log files and Event viewer

The log files indicate the environment healthiness. Based on the number of problems that occur during a day, the performance can be improved by reducing these errors. The important files are the Windows event viewer, system log on Unix/Linux and bshell and database driver log files located in $BSE/log. A daily check of these log files, is recommended.

To redirect the log messages to file in $BSE/log instead of the Windows Eventviewer, set the following parameter in $BSE/lib/bse_vars.

```
USE_LOGFILE=1
```

Purge temporary files and log files

Infor LN generates temporary files in $BSE_TMP. This environment variable defaults to $BSE/tmp. It is recommended to delete all files older than 2 weeks in this directory every month using this script:

Windows:

```
set BSE_TMP=d:\inforln\bse\tmp
FORFILES /P %BSE_TMP% /D -15 /C "CMD /C del /q %BSE_TMP%\@FILE"
```

UNIX/Linux:

```
#!/bin/ksh
export BSE_TMP=/inforln/bse/tmp
find $BSE_TMP/* -mtime +15 -exec rm {} \;
```

It is also recommended to clean up these files regularly:

- User history saved in $BSE/lib/TIME.his
- Log files saved in $BSE/log

- Audit files saved in $BSE/audit

Purge print queue

It is recommended to clean the print queue weekly. Create an Infor LN job which runs the Purge Device Queue (ttaad3221m000) session and cleans the print jobs older than 2 weeks.

Remove application locks

It is recommended to remove application locks daily. Create an Infor LN job which runs the Remove Range of Application Locks (ttadv9298m000) session and removes the locks older than 1 day.

Check jobs

It is recommended to check daily whether the Infor LN jobs are completed successfully using the Job Data (ttaad5500m000) session. Running jobs can have significant impact on the performance experience of OLTP users. Monitor the job duration over a period of time and check for error messages.

Check first free numbers

During the lifetime of Infor LN, the first free number series increases. This is not unlimited; therefore, it is recommended to check regularly whether the first free number series are getting exhausted. The length of a series + first free number may not exceed 9 characters. For example, when the series length is 3, the maximum first free number is 999999. When the series length is 7, the maximum first free number is 99. Use this bsql query to display the number groups and the first free number series.

```
select a.nrgr,a.dsca,a.lngt,b.seri,b.dsca,b.ffno
from tcmcs051 a, tcmcs050 b
where a.iuse = 1
and b.blck = 2
and b.nrgr refers to a.nrgr
order by a.nrgr, b.seri
```

Maintenance, batch, and backup window

An optimized Infor LN environment requires maintenance. Data changes over time, patches are installed, and new functionality is added. Therefore, index maintenance is required, database statistics must be updated and integrity checks must be done.

To prevent locking issues and performance loss due to these tasks, it is recommended to schedule these tasks during the weekly maintenance window (schedule). You must ensure that the maintenance window does not overlap with the backup window and the batch window, wherein the

Infor LN jobs, such as MRP, are executed. User activity must be at a minimal during the maintenance, backup, and batch window.

Caution: Infor LN solutions which require table modifications must be installed only during a maintenance window instead of the normal production hours, because tables can be locked for longer time. This can lead to Infor LN sessions failing or performance issues during installation.

See the database specific Performance, Tracing, and Tuning Guide for more information on the database specific maintenance tasks.

Combo bshell

Formerly, the bshell and database driver ran as separate processes. Currently, when the combo bshell is used, the database driver is linked as a library in the bshell. The combo bshell is developed for performance reasons as this eliminates an additional process communication between these processes. In the earlier scenario, during the context switches between the bshell and database driver, a significant amount of CPU was spent on system activity. Running the combo bshell provides 10% percent CPU decrease for OLTP and more than 20% for the batches.

To enable the combo mode, an extra line must be added in the $BSE/lib/ipc_info file. The ipc_info file contains:

- The driver name (first argument)
- Reference to combo (second argument)
- The library name (third argument)

To enable combo mode for Oracle on UNIX the ipc_info file must contain:

```
oracle8        d  ora8_srv6.2
oracle8        s  ${BSE}/bin/ora8_srv6.2
```

To enable combo mode for Oracle on Windows the ipc_info file must contain:

```
oracle8        d  ora8_srv
oracle8        s  ${BSE}\bin\ora8_srv.exe
```

It is recommended to set the line that contains the ‘d’ option, before the line with the ‘s’ option. The first Oracle database driver ensures that the bshell starts in combo mode. If another database driver must be started, the non-combo mode is activated, because only one database driver can be started in the combo mode. In normal scenario, only one database driver is started, but when the table definitions are modified to support, for example multiple databases, casein alternate database driver is required.

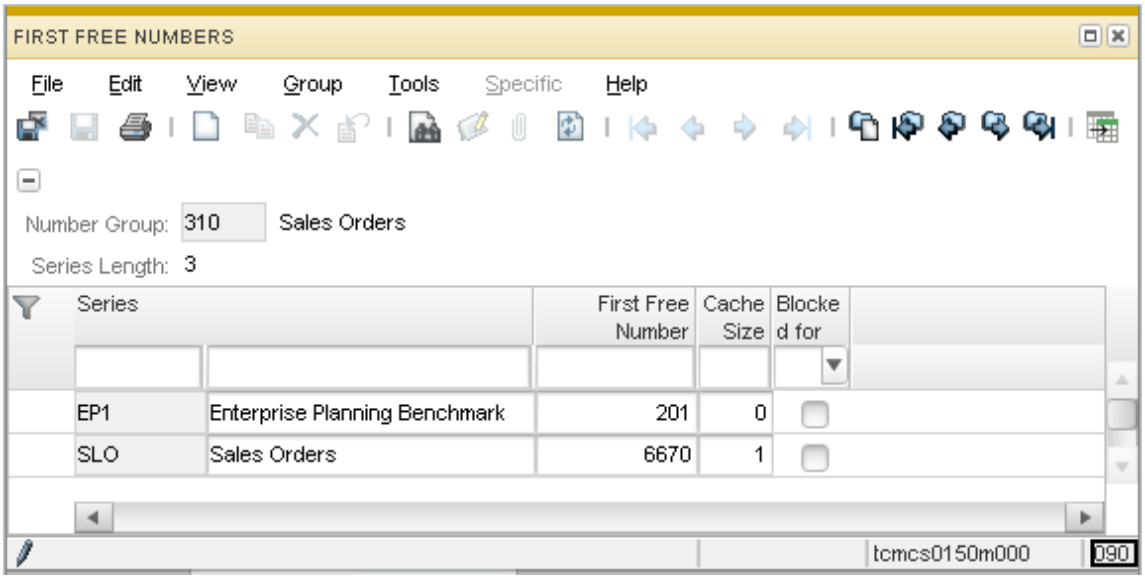
(Master) Application Server

Infor LN can scale over multiple application servers. One of the server's acts as Master Application Server (MAS) and the others run as Application Server (AS). The Application Servers only contain a minimal Infor LN setup (portingset only) and the Infor LN runtime is synchronized using the File System (FS) process. For performance reasons, it is recommended to run the database driver on the Application Server, connecting directly through the native database client libraries to the database server. This enquires the database client software installed on the Application Server.

For more information, see *Infor LN Installation Guide (U9498)*.

First Free Number Cache

First Free Numbers (FFN) are used for sequences in tables such as order numbers, invoice numbers, and contract numbers. These numbers are setup per Number Group and series length. The example explains the series EP1 and SLO for Number Group 310 and series length 3. The last used number for the EP1 series is 200 and the First Free Number is 201. The EP1 series do not use caching; the SLO series uses caching.



When a program requires a new first free number (FFN), the current value is considered, and the same is increased by 1, the record is updated in the database and the transaction is committed. The number can be used by the next program that requires a FFN. During the time between the update of the FFN and the commit, the record is locked for other users/programs. However, this can cause issues as some programs require additional time before the transaction can be committed.

From Baan 5 onwards, the session is enhanced with a cache. Setting the cache to 1 or higher enhances the program such that a FFN is read, updated, and directly committed in a separate process. This reduces the lock time on the FFN table and prevents locking issues on this table. The size of the cache determines the numbers that must be cached for the process.

If the cache size is set to 0 (zero) the application behaves as required. The FFN table is not locked till the commit. No gaps occur in the sequence numbers.

Setting the cache size to a value of 1 or higher, increases the FFN record with the cache size and immediately commits the value in a separate process. The calling program receives the number(s) that can be used as FFN. When the calling program uses numbers lesser than the cache size or if a transaction is aborted, gaps in the sequences can occur.

Cache size	Description
Cache size = 0	No caching is applied; user request a new number, the number is committed after the transaction is completed. Advantage: No gaps in the numbers. Disadvantage: Number series is locked during the entire transaction; other users cannot request a new number from the same series until the entire transaction is completed. Usage: Use this option if gaps between numbers are allowed. Request a new number as close as possible to the end of the transaction to reduce the lock time. This setting can cause performance and locking issues in high volume implementations.
Cache size = 1	User requests a single new number and the number is committed immediately. This occurs independently of the completion of the transaction. Advantage: Number series is locked for the least period, resulting in a reduced lock time and an improved performance. Disadvantage: Possibility of gaps in the number series, in case the entire transaction is not finished. Usage: This is the default value for number series to avoid performance and locking issues in high volume implementations.
Cache size > 1	User requests multiple new numbers and all the numbers are committed immediately. This occurs irrespective of the completion of the transaction(s). Advantage: Number series is locked for least period and the number of updates to the number series is reduced, resulting in lesser locks, better concurrency and an improved performance. Disadvantage: Possibility of larger gaps in the number series, in case the entire transaction is not finished. Usage: This is only recommended for batches, in case the cache size =1 setting does not solve the locking issues.

When gaps in FFN sequences are not allowed, the cache in FFN must not be used. When gaps in FFN sequences are allowed, it is recommended to set the cache to 1 when multiple users/processes use the same range. It is only recommended to set the FFN cache size higher than 1, when the cache size to 1 still shows locking issues on the FFN table. See *Infor LN - Performance Guidelines (U9502 US)* for more information.

Environment variables

To optimize and fine-tune the Infor LN application, environment variables can be set. For example:

- Set the environment string of the user interface.
- Specify the '-set' option in the user environment.

If certain variables are often used, the variables can also be specified on a more common location. Depending on the operating system, these can be placed in:

- The $BSE/lib/bse_vars file.
- The $BSE/lib/defaults/db_resource file.
- The $BSE/lib/defaults/all file
- The $BSE/lib/tabledef6.2 file
- The Infor Manager on Windows systems.
- In the Windows or UNIX environment (shell).

See the *Infor Enterprise Server Technical Manual (U8172)* for more information.

The bse_vars file

The bse_vars file can be placed in several locations on a system. The search order for reading this file:

1. The current directory; for BW interfaces, this is the home directory.
2. The $BSE/lib directory.
3. The /usr/bse directory. This is an absolute PATH and independent from the location of the $BSE variable.

If a variable is located in one of the files, the same is used with the specified value. The order of the search also specifies the priority for setting the variable. It is recommended to set variables for all users in the $BSE/lib/bse_vars file, and for specific users in the command field of the user interface.

The variables can be set in the $BSE/lib/bse_vars file as a normal environment variable:

```
<Variable>=<Value>
```

Example:

```
SLMHOME=/usr/slm
```

Note: For UNIX, as of porting set 8.8a.01, variables specified in the $BSE/lib/bse_vars file, and already existent in the environment, are overwritten by the value specified in the bse_vars file when connecting with a client using the rexec or blogin protocol. The value of the variable in the bse_vars file has precedence over the (previous) value of the environment variable. With earlier porting sets, the value in bse_vars is ignored, if a variable is already specified in the environment (of the rexecd or blogind process).

Variables in the bse_vars file, which are not already set as an environment variable, are added to the environment, similar to previous porting set versions. With this new behavior, it is possible to

override the PATH variable. Note: Values must be specified as fully qualified values, that is, no expansion of variables is done. For example, the following construct is invalid: PATH=$PATH:/home/user/mybin, but PATH=/home/user/mybin is valid.

Note: For Windows, as of porting set 9.0b, the variables in the bse_vars file take precedence over the Windows registry.

The db_resource file

Several parameters can be specified in the db_resource file. The db_resource file is located in <BSE>/lib/defaults. The variable name for the db_resource is identical to the environment variable but specified in lowercase.

To separate the variable and the value, a colon ':' must be used:

```
dbslog:1000
```

The all file

The <BSE>/lib/defaults/all file contain settings that can be used for every Infor LN process. For example:

```
log_size:4096
log_rotation_count:10
```

The tabledef6.2 file

The <BSE>/lib/tabledef6.2 file can contain environment variables that can be used for every bshell. Storing environment variables in this file is not recommended, however the environment variables set in this file are used.

Infor Manager Utility

If Infor LN is installed on a Windows system, the Infor Manager utility can be used to set the environment variables. The Infor Manager utility writes the variable in the registry path HKLM\Software\Baan\<BSE>\Environment.

Shell or environment

It is also possible to set the environment variables in the shell or the windows environment. Normally, these parameters are not read by the bshell. However, these parameters are read, for example, by the login deamon. You must ensure that the environment is clean when starting the login daemon. If the **blogind** is started with certain environment variables, this results in an unexpected behavior, such as failing connections or starting in a wrong company. The environment variables that are set when the blogind is started are inherited by the programs started (bshell, fs, and database drivers).

For example, if the BSE_COMPNR variable is set, the bshell starts in the specified company, and not in the users default company.

Examples of the variables:

- BSE_COMPNR
- BSE_USER, USER
- PACKAGE_COMB
- ORA_USE_VARCHAR

Environment parameters

These environment variables are important:

- BAAN_SQL_TRACE
  See the information on tracing with BAAN_SQL_TRACE. This variable can also be specified in the db_resource file.
- BDB_ALWAYS_FLUSH
  See the information on Call Graph Profiling.
- BDB_DRIVER
  When a database driver, other than the driver specified in the table definitions, is required, you can specify the new driver using the BDB_DRIVER variable. By specifying this variable, the table definitions are not read from the file. The tables to be audited are ignored. Use this variable only for test and tracing purposes.

  For example, use this parameter to test a new database driver other than driver installed and specified in the ipc_info file:

  ```
  BDB_DRIVER="oracle8_optim(SQL_TRACE=TRUE)"
  ```

  Example to start the driver on a remote system:

  ```
  BDB_DRIVER="db-remote"
  ```

- BSE_TMP
  Temporary Infor LN files can become large files. It is important that the performance critical sessions have the BSE_TMP set to a file system that is not slow, such as:

  ```
  BSE_TMP=$BSE/tmp
  ```

- BSE_SORT
  The sorting files are large and on a disk which is not slow, for example:
  ```
  BSE_SORT=/u2/sort
  ```
- DBSLOG
  See information on tracing with DBSLOG. This variable can also be specified in the db_resource file.
- DBSLOG_LOCK_PROF
  The DBSLOG_LOCK_PROF parameter specifies the minimum duration of a lock that must be logged. Any locks of shorter duration are not logged. This variable specifies the minimum number of milliseconds, which must elapse before a lock is logged. Lock time is calculated from the time when the first record in a transaction is locked to the time of the commit or abort. This is the longest time a record remains locked during a transaction. Note: The appropriate dbslog categories must be set, for example:
  ```
  DBSLOG_LOCK_PROF=2.0
  ```
  This variable can also be specified in the db_resource file.
- DBSLOG_NAME
  The DBSLOG_NAME parameter specifies the output of the DBSLOG file:
  ```
  DBSLOG_NAME=/logfiles/locking.log
  ```
  This variable can also be specified in the db_resource file.
- DISABLE_DEBUGGER
  This parameter prevents 4GL debug screens and continues when a session is compiled in debug mode. The parameter is similar to the '-nodebug' option that can be added in the BW configuration screen:
  ```
  DISABLE_DEBUGGER=1
  ```
- <DB>PROF
  See the information on tracing with <DB>PROF. This variable can also be specified in the db_resource file.
- PROFILE_ALL
  See the information on Call Graph Profiling.
- PROFILE_TRAMPOLINE
  See the information on Call Graph Profiling.
- PROF_CLIENT
  See the information on Call Graph Profiling.
- PROF_DIR
  See the information on Call Graph Profiling.
- PROF_RTIME
  See the information on Call Graph Profiling.
- SQL_TRACE
  See the information on SQL_TRACE. This variable can also be specified in the db_resource file.
- TEST_RETRY
  The variable TEST_RETRY specifies the frequency with which the system must return to the retry point during a commit. This variable must not be enabled in a live environment to avoid performance and locking problems, such as:

```
TEST_RETRY=3
```

In this example, when a commit is executed, the driver returns to the retry point without committing the data. When the commit is executed again, the process is repeated. The third time the commit is executed, the actual commit is done.

- MAX_RETRY
  The MAX_RETRY parameter sets the maximum number of retries for a transaction. Each time the transaction fails and returns to the db.retry point, the number of retries is increased by 1. This continues until the value of the MAX_RETRY parameter is reached or the transaction succeeds. If the number of MAX_RETRY is reached, the session is stopped and an error message is displayed. The default value is 10. If an error displays that 10 retries are insufficient, the value for the MAX_RETRY parameter can be increased. For example:

  ```
  MAX_RETRY=20
  ```

  Instead of increasing the value for the MAX_RETRY parameter, it is recommended to check the application logic. You can also increase the lock wait time using lock timeout settings. In that case, the transaction waits longer for acquiring the lock rather than re-executing the query from its retry point. See the database specific guides for more information about lock timeout.
- RETRY_DELAY
  If the bshell must retry a transaction, the RETRY_DELAY parameter specifies the delay time between the retries. The first retry does not have a delay. The formula for this delay:

  ```
  Delay time = (# of retries – 1) * retry_delay
  ```

  The default value for the RETRY_DELAY is 200 (ms). This default value can be overwritten by specifying the RETRY_DELAY in the all file. The value must contain a valid number in milliseconds.
- USR_DBC_RES / USR_DBS_RES
  The variable USR_DBS_RES specifies the location of a db_resource file. This contains a relative path starting from the $BSE environment. For example:

  ```
  USR_DBS_RES=lib/defaults/batch_resource
  ```

  This variable is used to specify an alternative db_resource file for the specific sessions/jobs. The USR_DBC_RES variable sets the location for the client, and USR_DBS_RES sets the location for the server.
  Note: Any driver resource set in the alternative resource file overrides the setting of the same driver resource in db_resource.

See the *Infor Enterprise Server Technical Manual (U8172)* or the database specific reference guides for more information.

Dynamic Scrollbar

Infor LN uses a dynamic scrollbar. When a multi-occurrence session is started, a query is executed to determine the size of the scrollbar. This query can take time, which can increase the time to start the session.

The application query used for determining the dynamic scrollbar can be found by searching for 'SELECT count (*):1'. If this query is time consuming, the dynamic scrollbar can be disabled. From Infor Enterprise Server 8.6 onwards, a specific session is available containing a list of sessions for which the dynamic scrollbar is disabled. This session can be started using the Maintain parameters (ttaad0100m000) session. By default, the dynamic scrollbar is disabled, when a session count is 10,000 records or more.

When running Enterprise Server 8.5 or lower, the environment variable SCROLL_LIMIT=-1 must be used to disable the dynamic scrollbar. This environment variable can be set in the bse_vars file, which disables the dynamic scrollbar for all users and sessions.

Database driver parameters in the db_resource

This section describes the database independent performance-related db_resource variables. The database dependent variables are discussed in the database specific performance, tracing, and tuning guides.

- rds_full
  The resource rds_full can only be used when run in the non-combo mode (separate bshell and database driver process) and sets the maximum number of rows transferred to one block from the driver to the bshell. Many batch sessions are using set-oriented queries, but most of the OLTP sessions fetch 1 row at a time; therefore, the best starting value is approximately 2. In most scenarios, the difference is negligible when set to 1 or 3. However, if many users access the system, the database is overloaded with SQL statements, or too much data is read when this value is not set as per the requirements. From porting set 8.4b onwards, the default is 2. In the application the value of rds_full can be overwritten for a specific query by adding this code:

  ```
  hint array fetching and array size 100
  ```

  When setting rds_full to a specific value, you must also check the value of array fetch; these must be the same for optimal data transport from the database to the database driver.

  In case the database driver is linked as a dll to the bshell (known as combo mode), this setting is ignored. This is the default setting.
- bdb_max_sessions
  The resource bdb_max_sessions define the number of sessions per driver. If any driver reaches the threshold, a new driver is started to handle new sessions. It is not recommended to use this parameter because of the additional memory consumption.
- bdb_max_session_schedule
  The resource bdb_max_session_schedule indicates the minimum number of sessions that are open in the database-driver. Every time an Infor LN session is closed, the driver checks the value of this parameter to determine if a session must be kept open in the driver. Specifying a value that is low causes additional overhead during close and start of a session, but specifying a value that is too high consumes the memory. Depending on the configuration, use, and requirements, this variable can be set to a specific value. Usually, the value of 3 is sufficient, but for batch sessions that switch between jobs, it is recommended to increase this value. For example:

```
bdb_max_session_schedule:250
```

- mle_join_type
  The resource mle_join_type determines the type of join used, in case of a Multi Language Enabled (MLE) environment between the data table and the corresponding shadow table (with translations). The default join type (INNER) is recommended for normal circumstances. However, the database does not always generate an optimal execution plan. This can be the case when 5 data languages or more are used. This behavior can be changed by using a LEFT join type between the tables. This resource is implemented for MLE tables for all databases; however, it must be applied after extensive testing. When upgrading to a new database version, the use of this resource must be validated again. The following values are supported:
  - 0: INNER (default)
  - 1: LEFT

Driver parameters in the database definition

You can set different parameters in the Database Definitions (ttaad4510m000) session. These parameters can also be set in the db_resource.

It is recommended to set parameters as much as possible in the db_resource file for maintenance reasons and a better overview of the parameters.

After the required modifications in the database definition, you must convert the data to runtime.

There are two parameter files that influence the behavior of the database driver:

- The storage parameter file.
- The driver parameter file.

Database storage files contain parameters that influence the creation of tables and indexes. The contents of these files differ per database. See the database specific reference guides.

Auditing

Infor LN specifies some tables for auditing, which are mostly parameter tables; however, in most situations other tables also must be audited. Auditing impacts performance, but in certain situations this can become a performance bottleneck. Therefore, it is recommended to minimize the number of tables to be audited. Auditing can be specified in Tables by Database (ttaad4111m000) session.

Besides Infor LN auditing, it is possible to use database auditing. The number of tables audited by the database must be kept to a minimum to optimize performance.

Shared tables

It is possible to share tables across companies, which can be specified in Logical Tables (ttaad4120m000) session. During the conversion of the runtime of this session, the data is stored in the $BSE/lib/compnr6.2 file. When starting a new database connection, the driver reads the content of the file to determine the location to store the tables. Based on the size of the file, the time to start the driver is impacted and the preparation of queries takes additional time. For administrative and performance usage, it is recommended to load the shared tables in modules, instead of using specific tables. For example:

```
tcemm:500:501,502,503,504,505
tcfin:500:501,502,503,504,505
```

Database authorizations

In Infor LN, it is possible to use these types of database authorizations:

- Company authorizations (ttams3144m000)
- Table authorizations (ttams3142m000)
- Field authorizations (ttams3143m000)
- Table data authorizations (ttams3145m000)

The company, table, and field authorizations are checked in the bshell/driver, but the table data authorizations are checked in the database. Table data authorizations are added to the query and have an impact on the performance; usually, the performance is not optimal. Table data authorizations can be split into 2 categories:

- Table data authorizations are a part of an index. The performance can be acceptable, if the table data authorizations are only on the first field of the table and the table data frequently searched using a specific index.
- Table data authorizations are not part of an index. These authorizations are potential performance bottlenecks. If the table is small (less than 100 rows), this usually does not affect the performance, but if the table is large, the performance can decrease. Note: Tables are relatively small when created, but grow over time. Therefore, it is important to prevent table data authorizations, even when the table size is small.

The table data authorizations must be kept to a minimum to prevent performance problems.

Shared memory usage

Infor LN uses shared memory for sharing frequently used data definitions, objects, and reports. This improves performance and saves memory because all the shared objects are not required to be loaded to private memory of each user again.

To reduce disk I/O it is recommended to set a linger time for which objects are considered valid. During this period, the shared memory manager does not check for validity of the object. In case Infor LN updates are applied, it takes more time before the updates are loaded into shared memory.

Add this to the <BSE>/lib/bse_vars file to set the shared memory linger time to 5 minutes (300 sec):

```
SHMLINGER=300
```

Note: Restart Infor LN shared memory to activate this setting. It is recommended to use a lower linger time when debug objects are in use.

For more information about shared memory see the *Infor Enterprise Server Technical Manual (U8172)*.

Shared memory block size

It is recommended to create a single block for all shared memory data on UNIX. All data is loaded to the shared memory, after the rc.start command (by the srdd_init –i command). The amount of shared memory in use is displayed using this command.

```
$ shmmanager6.2 -s
SEMID 945818141
Bshell common ptrs :
Reserved                0x0
Objects                 0xa0000000001e088 version 0x08070104 (caller is using dynamic
shm directory service)
not used                0x0
not used                0x0

not used                0x0
Security Files          0xa00000000000930
Compnrs                 0xa00000003b1ecb8
Db driver DD            0xa00000003b2ef70

Shm Flags               0xa0000000001b258 version 0x08070104 flags 0x00000001
Date Currency           0xa000000000019d0 mtime: Tue Apr 20 11:03:05 2010
not used                0x0
Bshell Cmd              0x0

Audit SeqNr             0xa00000003b56f00
Options                 0xa000000000004b8
not used                0x0
Shm Timer               0xa000000000004a8

      DESCRIPTION OF SEGMENT TABLE
USED BYTES 62222120 FREE BYTES 4886744 SHMID 31460072 NO ATTCH 2
NODE a00000000000000 SEG [0] ATCH_ADDR a00000000000000 FREE_ADDR a00000003b56f28
```

The example shows 1 segment of 64 MB. This is the default and recommended value. The segment size can be set by specifying shm_segment_size in the $BSE/lib/shm_config file. This file can be used to override programmed defaults for shared memory. It is recommended that you use the specified defaults. Setting values is only required when attaching data to the shared memory fails. The size of a block is equal to the sum of USED BYTES and FREEBYTES.

Package combinations in shared memory

Shared memory is used to store the database definitions of a package combination. The package combinations that must be loaded in the shared memory are specified in Package Combinations (ttaad1520m000) session. It is recommended to load all package combinations linked to a user and company to the shared memory.

Objects and reports in shared memory

To reduce memory usage and improve performance, it is possible to load objects into shared memory. Infor LN offers the option to load frequently used objects and reports to shared memory by measuring the used objects during a period of time. See the *Infor LN Administration Guide (U8854 US)*.

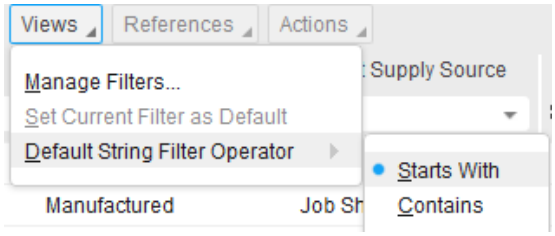
Easy Filtering

When using Infor LN UI, users have the possibility to filter on a lot of columns. This is a powerful feature, but can have huge performance impact. Recommendations:

- Consider whether you want users to filter on non-index fields because this consumes time and leads to resource intensive full table scans. You can change the easy filtering behavior using the Maintain Parameters (ttaad0100m000) session => Configuration Easy Filter => Allow Easy Filter.

  It is recommended to set this parameter to 'Table Index Fields' or 'All Fields till Runtime Count Limit'. The filtering depends on the count limit specified in the Count field, in the Maintain Parameters (ttaad0100m000) session. If the number of records in the table is greater than the count limit, easy filtering is only enabled for index fields in the grid.
- When using segmented fields and not using project codes, select 'Is empty' for the first segment (project code) and use the 'Starts with' filtering option for the second segment (item code):

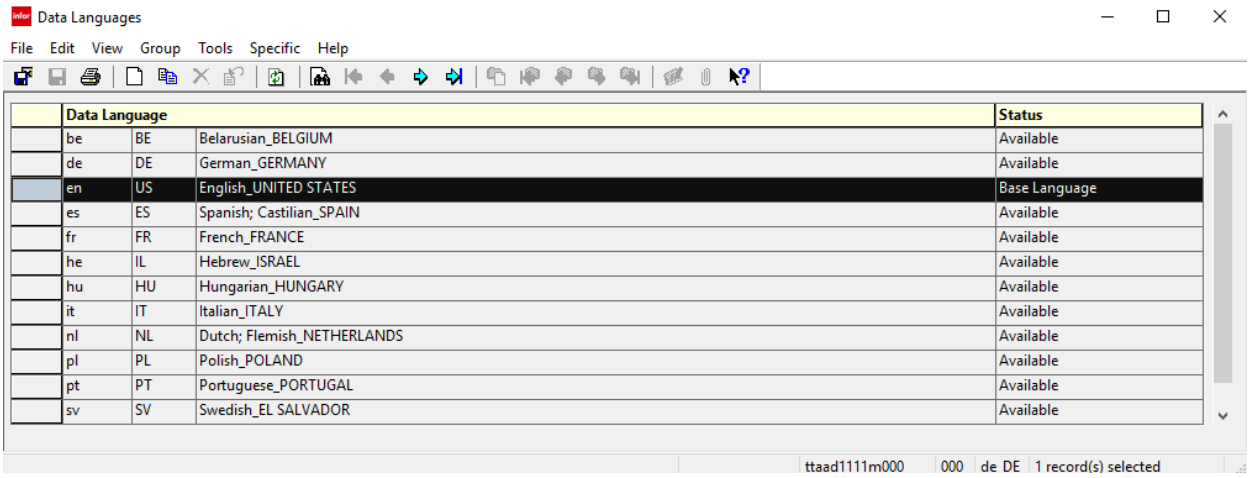
- If you know the first part of an item code, use the option 'Starts with' rather than 'Contains'. When using 'Starts with', an index can be used if a matching index exists, preventing time consuming and resource intensive full table scans.
- It is recommended to change the default filtering method per session to 'Starts With' using Views => Default String Filter Operator -=> Starts With



For detailed filtering options, see *Infor Ming.le-LN Plug-in User Guide (LN UI) (U9791)*.

Multi language enabling (MLE)

It is possible to save data for many fields in multiple languages. For example, you can get the item description of the same item, in your language. The data translation is saved in shadow tables. To read and/or change the translation, the query joins the data of the shadow table. For optimum performance, it is recommended to set the base language in the Data Languages (ttaad1111m000) session to the language used by most Infor LN users.



Note: Select additional Data Languages carefully. Adding a Data Language has overhead and decreases performance, especially on reporting sessions when data of all languages is read.

Validate Data Integrity in parallel

From portingset 9.1c, it is possible to run bdbvalidate (Validate Data Integrity session) in parallel. Running bdbvalidate is a typical step in a migration or upgrade scenario, to ensure that the data is consistent before and after the migration or upgrade. The number of parallel bdbvalidate processes is configurable using a resource variable. Running an older Infor LN version with the new portingset can take advantage of this new feature.

Use Infor LN resource bdbvalidate_parallel to specify the degree of parallelism. Include option '-J N' to configure degree of parallelism when running bdbvalidate on the commandline.

Chapter 5 Batch Performance Optimization

5

This chapter describes the settings that can be modified to increase the speed of the Infor LN batch processes. The OLTP users must not experience performance issues based on the settings required for the batch processes. Therefore, the system can be optimized for OLTP users and for batch sessions, but the customized settings must be implemented separately.

This chapter also describes the process to improve the batch performance using performance boosters and the method to tune the same. Currently, the following performance boosters exist:

- Parallel processing
- Table boosters
- Prevent progress indicator

Besides these boosters, other configuration options exist to improve the batch performance.

Parallel processing

Parallel processing is a mechanism in the application to speed up a batch session by spreading the load across multiple bshells.

Using parallel processing, the duration of a batch process can be lowered and the throughput can be increased. The disadvantages of parallel processing are:

- Tracing and debugging is difficult.
- Contention on resources increase.
- Increased possibility for failures.
- Additional consumption of resources, such as CPU and memory.
- Additional Infor LN licenses.

It is recommended to implement and use parallel processing discerningly. These sections describe the aspects of implementing and tuning parallel processing.

Preparations

Before testing parallel processing, it is recommended to test the batch sessions without parallel processing and check for any bottlenecks. For example, if disk I/O is already a bottleneck, parallel processing does not improve the system's performance.

Configuration

Parallel processing is implemented in a couple of Infor LN sessions. To check if a session includes the parallel processing capability, you must execute the session once. If the session includes the processing capabilities, running the session generates a record in table ttaad720, in the application company. You can define the settings for parallel processing in the **Parallel Processing Configuration** (ttaad7520m000) session. Use this session to search for the session with the parallel processing capability. The value of the **Mode** field is set to **Not Active**, which is the default value for the performance configuration. The available options:

- **User:** If a user is specified, the values are valid only for the specified user. If a value is not specified in the field, the values are applicable for all users. The user-dependent settings take precedence over the common settings. To specify settings for a user, a record must be added to the session. This is the only valid reason to add a record to this session; any other reason can confuse the end user, because the settings are not validated by the sessions.
- **Servers:** In all other scenarios, the session uses parallel processing only when the value of the **Servers** field is **2** or more. This value specifies the number of parallel bshells that are started to run the session.
- **Mode:** Parallel processing can be activated by specifying one of the following values:
  - **Not Active**: Parallel processing is not active for this session.
  - **Manual**: If a session is not started from a job, the session uses parallel processing.
  - **Job**: If a session is started from a job, the session uses parallel processing.
  - **Manual & Job**: The session always uses parallel processing.

Tuning

In general, due to the single thread architecture of Infor LN, each batch session consumes a processor core. Therefore, if a system contains 8 cores and at least 2 cores are required for OLTP users, you can set the number of parallel Servers to 6. When running in a 3-tier configuration, it is recommended to set a higher value because the load is distributed across the application server and database server. To find the optimal number of servers, you can run the same batch several times, with several values, for **Servers**. In general, 50% of the batch processing is done by the bshell (including database driver) and 50% is processed by the database and network.

After using parallel processing, for a session, for over a year, it is recommended that you check if the number of servers is still optimal for the configuration, because the dataset and other settings might

change. If you change the system, for example, decrease or increase the number of cores, you must reconsider the settings for parallel processing.

Troubleshooting

An error message is displayed, in case a batch session with parallel processing fails. More information can be found in <BSE>/log/log.parbshell. For additional information in the log.parbshell file, the environment variable TRACEPARBSHELL can be used. The following values can be used:

- 0: no tracing
- 1: trace API calls
- 2: trace communication
- 3: trace API calls and communication

The construction of a trace line:

```
[date time] <user>:<session>(pid), <component> (bshellpid): <message>
```

A <component> is one of the following:

- ClientAPI
- ServerAPI
- SupportAPI
- ServerCOM
- ClientCOM

Debugging

The sources must be debugged when an in-depth analysis is required. These steps must be completed:

1. Compile the required sources in the debug mode.
2. Start a maximum of two servers to prevent an overload of work on the debug screens.
3. Set this commandline option:

   ```
   -set APP_DS=bw
   ```

   Use this option to display the BW interface and a debug screen for new connections.
4. Start the client bshell to initiate the debugging process.

Documentation

Additional documentation on parallel processing is available in the Infor LN programmer's guide.

Table boosters

Table boosters are programmed to cache frequently used table data. Reading data from a bshell memory is faster than reading data from the database or disk. In many sessions, lots of redundant data is read. For example, when items with the units are read, in many cases, the unit is the same and is not required to be read again. Using the tccomdll0201 table, a tool set is created to cache table data.

Configuration

Table boosters are implemented in a couple of Infor LN sessions. After running a session, the configurable table boosters are displayed in the Table Boosters (tcmcs0598m000) session (tcmcs0158m000 session for Baan IV), in the application company. By default, the value of the **Booster Valid** field is set to **Not Active**. The available options:

- **User:** If a user is specified, the values are valid only for the specified user. If a value is not specified in the field, the values are applicable for all users. The user-dependent settings take precedence over the common settings. To specify settings for a user, a record must be added to the session. This is the only valid reason to add a record to this session. Other reasons can confuse the end user, because the settings are not validated by the sessions.
- **Load option:** Specifies the method to read table data:
  - **Incremental**: A record is first read from memory; if not available, the record is read from the database and stored in memory. No index is generated to locate a record, and each record is scanned sequentially in memory.
  - **Full**: When the first record is required, the complete table is stored in memory. An index is also generated to locate a specific record quickly.
- **Maximum Number of Rows:** The Maximum Number of Rows option specifies the maximum number of rows that must be read in memory. The maximum size for a bshell is 30 MB per table (20 MB for Baan IV). The default maximum memory is 5 MB per table booster. To increase this value, the environment variable TCCOMDLL0201_MAX_ALLOC can be used. The value of this environment variable must be specified in bytes. The default value is 5.120.000 bytes per table booster. To set the maximum memory to 6 MB:

  ```
  -set TCCOMDLL0201_MAX_ALLOC=6144000
  ```

- **Booster Valid:** Table boosters can be activated by specifying one of the following values:
  - **Not Active**: Table booster not active for this session.
  - **Manual**: If a session is **not** started from a job, the table booster is used.

- **Job**: If a session is started from a job, the table booster is used.
- **Manual & Job**: The session always uses the table booster.

Tuning

To tune the table boosters, the size of the table and the usage of that table must be known to identify if an incremental or a full load must be used. The advantage of an incremental load is that only the required records are added to memory. The disadvantage of an incremental load is that no index is generated. This can impact performance, if more than 400 records are read approximately. Based on this input, these rules can be used to identify the settings for a table booster:

- **Only a few records from a table are read**

  If only a few records (less than 50% with a total of less than 400) are read, the Incremental option must be used.
- **More than 400 records must be read**

  If the table is large, it is recommended to calculate the average number of times the table is read when compared to the number of records in the table.

  If the number of times the table is read is higher than the number of records in the table, it is recommended to load the complete table into memory, by specifying the Full load option.
- **Most records from a large table are read multiple times**

  If more than 50% of the records are read multiple times and the table is larger than 400 rows, the Full load option must be used.

Note: Some additional memory is consumed by the bshell for the table booster, which is not shared between users.

If none of the above options are applicable, it is recommended to disable the table booster as there is no performance improvement.

You can use the **Maximum Number of Rows** option to define a maximum number of rows to be used. If additional rows are required, the rows are read from the database. It is recommended to disable a table booster instead of reducing the value of the Maximum Number of Rows to be stored into memory.

Troubleshooting

Use these environment variables to trace table boosters:

-set LOG_COMDLL0201=1	Logs memory allocation and other technical information.
-set LOG_COMDLL0201=2	Logs the ‘idb’ calls including the arguments. So, when enabling one or more table boosters, the trace option displays the functions used and identifies the related arguments and table.

Prevent progress indicator

Most of the batch sessions within Infor LN are equipped with a progress indicator that displays the workload that must be processed and the workload part that is completed. The progress indicator is not applicable for the batches that run automatically and for the batches without an interface. Therefore, it is recommended to disable the progress indicator to prevent consuming additional CPU power:

- Before a progress indicator can be started, the number of records to be processed is counted.
- The interaction between the bshell and UI consumes time for every update of the progress indicator.

You can disable the progress indicator using Performance Boosters (tcmcs0597m000) session. If a session includes the **Prevent Progress Indicator** option, you can enable and disable the progress indicator. These fields can be set:

- **User:** If a user is specified, the values are applicable only for that user. Else, the values are applicable for all users. The user-dependent settings take precedence over the common settings. To specify settings for a user, a record must be added to the session. This is the only valid reason to add a record to this session; any other reason can confuse the end user, because the settings are not validated by the sessions.
- **Booster valid:** By changing the Booster Valid option, you can activate or prevent using the progress indicator:
  - **Not Active:** The performance booster is disabled. The progress indicator is used.
  - **Manual:** If a session is not started from a job, the progress indicator is not displayed.

Environment and driver parameters

You may be required to check the following parameters for optimal batch performance:

- bdb_max_session_schedule
- cursors
- array size (fetch and insert)

Consult the database specific performance, tracing, and tuning guide for more information about the parameters and the value to be specified for the batch sessions.

Batches can be started with different values by specifying the variables in the command line, or redirecting the batches to a separate db_resource file with the USR_DBC_RES and USR_DBS_RES environment variables.

Network

When running batches in a 3-tier configuration, the network is an important factor, which can delay the batches by a factor of 2 to 3 when compared to running the batch in the hostmode.

To improve the batch performance, consider these:

- Run the batch on the database server. The disadvantage of running a batch on the database server is that a (partial) BSE environment on the database server must also be created and maintained.
- Disable interrupt moderation on the network card. To reduce the number of interrupts, many NICs use interrupt moderation. When using interrupt moderation, the NIC hardware does not generate an interrupt immediately after a packet is received. Instead, the hardware pauses for more packets to receive or a time-out to expire occurs before generating an interrupt. The hardware vendor determines the specific algorithm for the packet size and time-outs.

  Consider to set **Latency Sensitivity** to **High** when running Infor LN in a VMware environment. Once the value is set to high, 100% of the VM configured memory must be reserved. It is also recommended to reserve 100% of its CPU.

  More information:

  - Deploying Extremely Latency-Sensitive Applications in VMware vSphere 5.5
  - Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs

The measured roundtrip time for a packet is a commonly used technique to determine the network bandwidth between two endpoints. However, when interrupt moderation is enabled, on receiving a packet, an immediate interrupt is not generated. The perceived roundtrip time for a packet is higher than the average time.

A performance decrease is expected for the Infor LN batches when using interrupt moderation, while it is beneficial for OLTP. Consider to disable interrupt moderation, when there is an issue with the batch duration. Based on Benchmarks, the batch duration can be increased up to 50% when you disable interrupt moderation on the network card. See the network card vendor documentation for information to disable interrupt moderation.

Note: CPU usage can increase. It is recommended to verify the proposed changes in a test environment before moving to production.

Chapter 6 Tracing Infor LN

6

When performance problems occur, the exact cause of the problem must be identified. If the performance problem is within an Infor LN session, tracing can be used to identify the reason. Tracing can be performed on several levels.

The following list contains the frequently used tracing methods for performance:

- Call Graph Profiling

  The Call Graph Profiler (CGP) can be used to identify functions and application queries that consume a lot of time. The CGP includes a call graph functionality that displays the relations between parent and child functions.

- BAAN_SQL_TRACE

  The output of the trace lists the Infor LN queries. The standard program queries are also traced. It is difficult to identify the queries that consume most of the time without additional tools. Also, all DML and DDL queries are not displayed in this output.

- ORAPROF, MSQLPROF and DB2PROF

  Depending on the database the ORAPROF, MSQLPROF or DB2PROF environment variable can used to display all queries from the driver to the database, including the timing information. A formatting tool is required to analyze the output, because each query is placed in one line. Note that the output can be large.

- bsql

  You can use the bsql tool to trace and test application queries. Using bsql prevents you from running a large session to identify if a query is optimized.

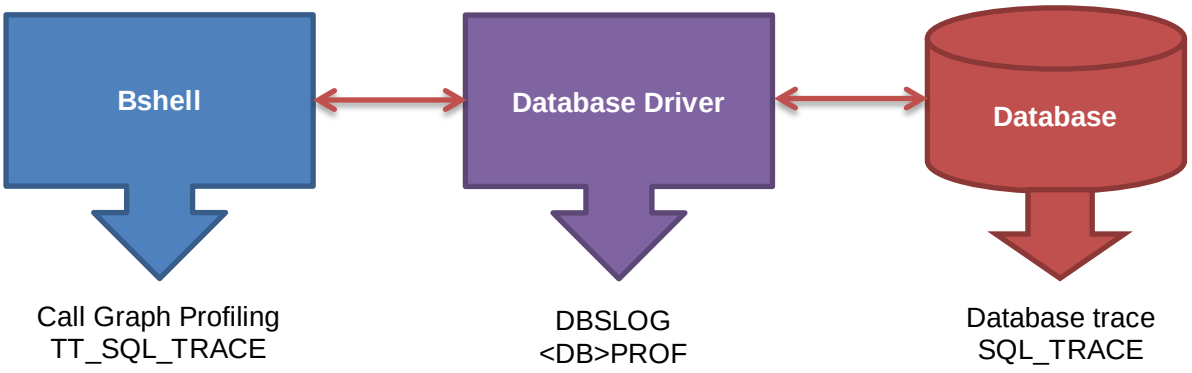
- DBSLOG

  This tracing method includes many possibilities. In this document, this method is only used to find the input variables for a query that must be reproduced by bsql.

- SQL_TRACE

  This option is an Oracle method to trace all Oracle queries and is described in the *Performance, Tracing and Tuning Guide for Oracle (B0078 US)*. The SQL_TRACE option is recommended when compared to ORAPROF, because the output is shorter and the standard format possibilities are better. This option displays the Oracle execution plan and resources consumed in Oracle.

Each tracing method is integrated in the application. The diagram displays a high-level overview of the components with trace possibilities. The trace possibilities affect each of these components.



Call Graph Profiling

This tool helps you monitor the time spent, from the bshell perspective.

The Call Graph Profiler (CGP) displays the 3GL and 4GL functions and application queries within the functions of the session. CGP also displays the relationship between the functions:

- The number of times the functions are called from other functions.
- The method used to call the functions using the other functions.

The amount of time consumed is also added to the data.

Compiling objects in the profile mode is not required when using CGP, although compiling objects help retrieve function names, instead of function numbers when using a 6.1x or 7.1x porting set.

The output for CGP is HTML, which makes it easier to navigate and identify the performance problems.

Using CGP or any other tools consumes additional CPU power, spent on timing and other resource functions. Usually, a 5 percent to 10 percent additional CPU is considered.

The profile output is generated when a session ends as per the process.

Terminology

The terminology used in the sections:

- **Utime**: The user time; this is the CPU time that the bshell spent for the code.
- **Stime**: The system time; this is the CPU time that the operating system is utilizing for the bshell.
- **CPU time**: The amount of processing time spent to execute the code in the bshell process. CPU time is the utime + stime. This calculation does not include the time spent waiting for the database or the disk to respond, and does not include the time spent waiting for user interaction. When running in the combo mode, the CPU time also includes the CPU time used by the database driver.

- **rtime**: The amount of run time, sometimes referred to as wall clock time or elapsed time. The rtime is the amount of time that the process requires to execute the code, including the waiting time for databases, disks, user interaction, or other processes.
- **children/parents***:* The term ‘child,’ or sometimes ‘descendants’, is used for the functions that are called from a function. ‘Parent’ is also used for the functions that are used to call a function.

Features

All sessions are logged, except for these Enterprise Server (or tools) sessions that provide no performance information:

- ttstpmsg. Popup message.
- ttstpdisplay. Print output for device ‘D’.
- ttstppollmess. Display system message.
- ttstpsplopen. Select print device.
- ttstpsplclose. Print spooler close.
- ttstpoledaemon. OLE Daemon.
- All ttdsk* programs. Menu browser and so on.

The profile output is in the HTML format:

```
profile.<pid>.<session name>.<bshell_pid>.html
```

Explanation:

- pid: The operating system process id.
- session_name: The code of the session, such as tccom0501m000.
- bshell_pid: The process id of the session within the bshell. So, if you start the session twice, two trace files are generated, each of which has a unique bshell_pid.

By default, the file is placed in the BSE_TMP directory.

Usage

To enable CGP, these parameters can be used:

PROFILE_ALL

This variable activates the CGP. If you set the value to 1, all used sessions are traced. If you set the value to ‘tccom’ all session codes that contain this string are traced. Multiple strings can be specified, such as ‘tccom,ti’ So, all session codes that contain tccom or ti are traced. For example:

```
PROFILE_ALL=1
```

Or

```
PROFILE_ALL=tccom
```

If you set the value to 2, all used sessions are profiled including above Enterprise Server sessions. Note: This causes additional overhead.

PROFILE_TRAMPOLINE

From portingset 8.9a.05 onwards, a new feature is implemented in the Call Graph Profiler to profile specific bshell functions which are normally not traced. The output will be added to the normal Call Graph Profile output.

A comma separated list of functions can be provided to profile multiple specific bshell functions. An asterisk at the end of the string can be used to profile multiple functions. Example:

```
PROFILE_TRAMPOLINE=bshell.fun1[,bshell.fun2,bshell.fun*]
```

Use this setting to profile the alloc.mem function, for example:

```
PROFILE_TRAMPOLINE=alloc.mem
```

Caution: Use this option only when a bshell function is suspected for a performance problem.

PROF_DIR

This variable can be set to redirect the output to another directory instead of the default BSE_TMP. The supplied directory must exist. Example:

```
PROF_DIR=/home/user/profiles
```

PROF_CLIENT

This parameter must be set to an action and a directory. The action can be ‘start’ or ‘save’ to store the output on a client’s PC and to start the default internet browser to open the profile.

The directory must be a fully qualified directory on the local PC. Example:

```
PROF_CLIENT=start:C:\temp
```

BDB_ALWAYS_FLUSH

In the Infor LN environment, a Data Manipulation Language (DML) such as an insert, update, or delete action is normally delayed until the next query. Just before the next query is initiated, the DML action is performed. This ensures that the lock time caused by the DML operation is limited to a minimum amount of time. When debugging, testing, or tracing with the Call Graph Profiler, unexpected results can occur due to these:

- When a DML action hits a locked record, the action runs the actual statements on the next query. If next query is a normal query, the debugger must not return to a retry point on the specified point.
- The time consumed to execute a query can be long because a delete operation is executed before the query. This delete operation is forced to check a reference in another table. The time for this check of the reference and the delete is added to the time of the query.
- To force an immediate action on a DML action, the action can be extended with a third argument (eflag). However, extending the DML action using this process can influence the behavior of the session and requires additional coding.

Therefore, the variable BDB_ALWAYS_FLUSH is introduced. When setting the value of the variable to 1, all DML actions are immediately flushed instead of being buffered.

By default, the value of this variable is set 1 when the CGP is enabled.

Caution: BDB_ALWAYS_FLUSH is only intended for testing, tracing and debugging purposes.

bshcmd

You can activate the profiler using the bshcmd command. After triggering the bshcmd, only recently started sessions are profiled.

Example:

```
bshcmd6.2 -T "PROFILE_ALL=tdpur" <bshell-pid>
```

Output

The output of the CGP contains these parts:

- General trace information.
- CPU, wait and run time.
- Hyperlink to the query summary.
- Hyperlinks to sections sorted on CPU time (user time plus system time).
- Hyperlinks to sections sorted on real time (if PROF_RTIME is enabled.)

General trace information

This section contains information about the session, used system, BSE, company, package combination, 4 GL set, bshell name, porting set, trace date, and a link to the system information.

See the general information on the Call Graph Profile diagram.

When you open the system information file, the settings of various variables are displayed:

System information

Content

The system information is useful for situations in which the behavior of the bshell or database driver is dependent on any of these parameters.

Legend

This section describes the use of the parameters.

Legend:

```
rtime/run time: time it takes to run, also known as 'wall clock' or 'elapsed time'
cpu time      : processor time used
wait time     : included in run time when a process is waiting on UI, processes, etc.
utime         : cpu time executed in user space
stime         : cpu time executed in system space (Operating system time)
note 1. Due to rounding not all times might sum up exactly.
     2. cpu time has a granularity of 0.01s, run times have a granularity of 0.001s.
```

CPU, wait, and run time

The diagram displays the CPU time and duration of the session. The wait time is split into several sections.

Summary:

```
cpu time   :   0.770s
wait time  : (included in run tume)
               3.783s UI wait time  Evitable wait time
               0.447s Interprocess wait times  Inevitable wait time for other processes
               0.558s Interprocess run  time  Inevitable processing time executed by other processes
           + -------------
total          4.790s
run time   :   6.102s
```

The wait time includes:

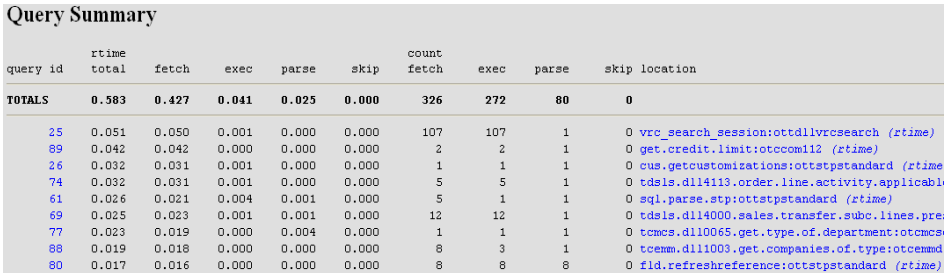
- Bshell waiting for UI events.
- Database flush operations such as insert, update, and commit actions.
- 3GL inter-process communication, such as bms commands.
- 3GL process zoom actions, such as zoom.to$ commands.
- 3GL process sleeps, such as brp commands

The 3GL sections also contain hyperlinks to the CGP of the related sessions.

Query summary

The query summary, as shown in the diagram, contains a link to the queries. These queries are sorted on runtime in descending order.

As the total time of the queries is included, you can quickly identify if queries must be improved. Example:



Query Summary

query id	rtime total	fetch	exec	parse	skip	count fetch	exec	parse	skip	location
TOTALS	**0.583**	**0.427**	**0.041**	**0.025**	**0.000**	**326**	**272**	**80**	**0**	
25	0.051	0.050	0.001	0.000	0.000	107	107	1	0	vrc_search_session:ottdllvrcsearch *(rtime)*
89	0.042	0.042	0.000	0.000	0.000	2	2	1	0	get.credit.limit:otccom112 *(rtime)*
26	0.032	0.031	0.001	0.000	0.000	1	1	1	0	cus.getcustomizations:ottstpstandard *(rtime*
74	0.032	0.031	0.001	0.000	0.000	5	5	1	0	tdsls.dll4113.order.line.activity.applicabl
61	0.026	0.021	0.004	0.001	0.000	5	1	1	0	sql.parse.stp:ottstpstandard *(rtime)*
69	0.025	0.023	0.001	0.001	0.000	12	12	1	0	tdsls.dll4000.sales.transfer.subc.lines.pre
77	0.023	0.019	0.000	0.004	0.000	1	1	1	0	tcmcs.dll0065.get.type.of.department:otcmcs
88	0.019	0.018	0.000	0.000	0.000	8	3	1	0	tcemm.dll1003.get.companies.of.type:otcemmd
80	0.017	0.016	0.000	0.000	0.000	8	8	8	0	fld.refreshreference:ottstpstandard *(rtime)*

To get more information about the time spent, you can use these links:

- Use the link on the query id to display the location wherein the query is parsed, in the section sorted on CPU (utime + stime). Example:

```
select   tccom112.crlr:o.credit.limit
from     tccom112
where    tccom112._index1 = {:i.itbp}
and      tccom112.cofc = ""
as set with 1 rows
```

- Use the link to the function:object, for example get.credit.line:otccom112, to display the time spent in the function, its caller and the called functions, in the section sorted on CPU (utime + stime).
- Use the link to the rtime section to display the time spent in the function, its caller and the called functions, in the section sorted on run time.

If you search for the query id 89, a sql.exec and a sql.fetch section are displayed. The sql.parse displays the application query. The sql.parse, sql.exec, and sql.fetch sections, are not the 3GL or 4GL sections, but are the query states in the bshell. During the parse, the query syntax is validated. During the execution phase, the query execution plan is generated in the database. During the fetch, the result of a select query is retrieved. A query can be opened in function 'A' and records can be

fetched using function ‘B’. For example, when used for dynamic queries. To locate these queries, use the Find option in your browser.

A query cache is available in the bshell and results in a different query representation. That is, a query is reused when hard closed, but this is only possible with dynamic queries.

Object summary

The object summary, shown in the diagram, displays all the used objects and the amount of time spent on the object. The object summary is sorted on the CPU time in the utime + stime section, and sorted on run time in the rtime section.

run time per object

object	run time	cpu time	perc	perc_t
ottstpstandard	**4.308**	0.320	70.6	70.6
ottstp_stddll	**1.038**	0.040	17.0	87.6
otcmcsdll0005	**0.264**	0.240	4.3	91.9
ottdllvrcsearch	**0.056**	0.010	0.9	92.9
otccom112	**0.043**	0.000	0.7	93.6
otdslsdll4113	**0.032**	0.000	0.5	94.1
otccomdll0201	**0.029**	0.000	0.5	94.6
otdsls4100	**0.029**	0.020	0.5	95.0
otdslsdll4000	**0.027**	0.010	0.4	95.5
ottstpamdll	**0.025**	0.030	0.4	95.9
otcmcsdll0065	**0.023**	0.000	0.4	96.3

The ‘perc’ column lists the percentage of time spent for the object. The ‘perc_t’ column lists the percentage used by the object, including all objects. This helps identify the objects that can contribute to a performance improvement.

Object function summary

Click the object’s link to display the functions in the object:

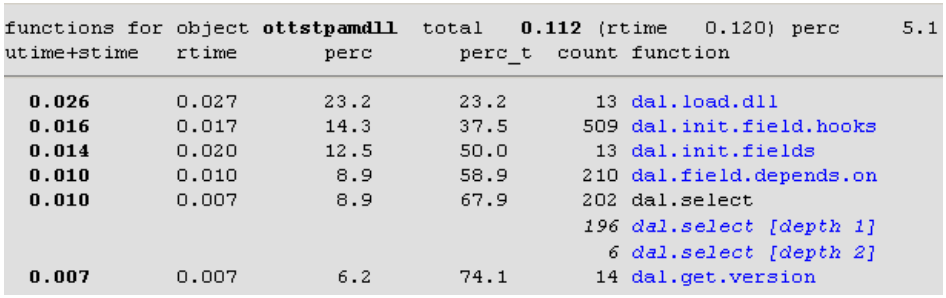
functions for object **otccomdll0201** total **0.029** (cpu time 0.000) perc 0.5

run time	cpu time	perc	perc_t	count	function
0.022	0.000	73.4	73.4	19	idb.eq
0.003	0.000	10.1	83.4	24	define.real.leng
0.001	0.000	2.3	85.7	13	init.buf.main.index
0.001	0.000	1.8	87.6	36	idb.alloc.mem
0.000	0.000	1.7	89.3	18	idb.init.buf
0.000	0.000	1.5	90.8	64	search.table.name

In the object function summary, the list ends when the perc_t value is 90 percent or more.

A function is displayed with a depth layer, when the function is called directly or indirectly.

For example, when a DAL calls another DAL, the dependency is identified by the depth:



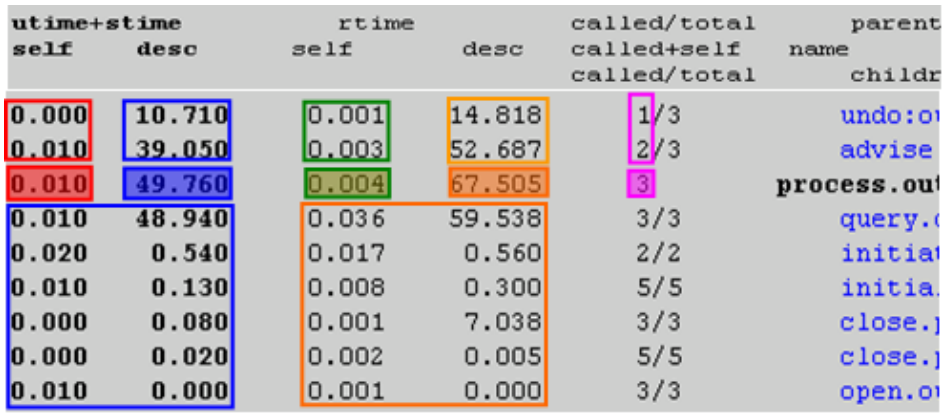
functions for object **ottstpamdll** total **0.112** (rtime 0.120) perc 5.1

utime+stime	rtime	perc	perc_t	count	function
0.026	0.027	23.2	23.2	13	dal.load.dll
0.016	0.017	14.3	37.5	509	dal.init.field.hooks
0.014	0.020	12.5	50.0	13	dal.init.fields
0.010	0.010	8.9	58.9	210	dal.field.depends.on
0.010	0.007	8.9	67.9	202	dal.select
				196	*dal.select [depth 1]*
				6	*dal.select [depth 2]*
0.007	0.007	6.2	74.1	14	dal.get.version

If the ‘dal.select [depth 1]’ function has no direct function call to depth 2, the function call is indirect. This is tracked by the profiler.

Call Graph

This section contains all the functions and the related parent-child functions. An example of a part of the output:



utime+stime self	utime+stime desc	rtime self	rtime desc	called/total called+self called/total	parent name childr
0.000	10.710	0.001	14.818	1/3	undo:o
0.010	39.050	0.003	52.687	2/3	advise
0.010	49.760	0.004	67.505	3	**process.ou**
0.010	48.940	0.036	59.538	3/3	query.(
0.020	0.540	0.017	0.560	2/2	initia
0.010	0.130	0.008	0.300	5/5	initia.
0.000	0.080	0.001	7.038	3/3	close.)
0.000	0.020	0.002	0.005	5/5	close.)
0.010	0.000	0.001	0.000	3/3	open.o

In the Call Graph profile, the process.outbound function is called three times, (marked in Purple). The function is called once using the ‘undo’ function and twice using the ‘advise’ function (marked in Purple).

The shaded red section displays the CPU time consumed by each function (0.010 sec). The red square indicates how the time is generated using the calling functions: The call from the ‘advise’ function consumed all CPU time required for the ‘process.outbound’ function.

The blue square displays the timing of all functions called from the ‘process.outbound’ function. The timing information of the functions and the children of the ‘child’ functions are also displayed. The functions are ordered on the total CPU time, wherein the function that consumes the most time, is the closest to the ‘process.outbound’ function.

The shaded blue section is the summary of the shaded red section (the CPU time) and the blue section (time consumed by functions). This is the total CPU time of all functions started using the ‘process.outbound’ function, including the time consumed by the ‘process.outbound’ function.

The blue section at the top displays the distribution of the time of the parent functions (shaded blue section). The undo' function is responsible for 10.710 seconds in the 'process.outbound' function and the children.

The function that contributes the most to the process.outbound function is placed closest to the 'process.outbound' function. This applies for the parents and the children.

The same process is repeated on the runtime columns, except for the ordering of data. To sort data on runtime, select the section based on 'rtime'.

A percentage column is also added. The information in this column can be used to calculate the time spent on the functions and the called functions.

Flat Profile

This section contains the functions ordered by CPU time or run time and also contains the number of times a function is called. This section describes the relationship between functions or the queries.

Analyzing the output

You can use the profiler to search for improvements. However, some of the improvements require an application insight while the other improvements require help from database specialists. The guidelines to analyze a profile:

Run time vs. CPU time

If the bshell, driver, and database are located on the same system, the run time is usually twice the CPU time when running in a non-combo mode. If the run time is larger, you must check the queries. If the CPU time is almost as same as the runtime, it is recommended that you search for CPU intensive tasks.

Queries

Bad performing queries are listed in the queries section. Always check the time gained by improving a query. For example, a query that takes 20 seconds may initially be on the higher side. However, compared to the total run time of 1 hour, it is most probably not the cause of the performance issue.

Expensive functions

The flat profile displays the expensive functions. The functions that consume most of the CPU time are displayed on top in the 'utime + stime section'. However, if the rtime section is also included, functions ordered on run time can help to identify the performance critical code.

Note: Functions can be costly when called often, which increases the expense per call.

Function tree

To identify performance overheads in the application, use the function tree (next to the call graph). When using the function tree, it is useful to have application knowledge. When investigating performance issues, check for these functions:

- Functions that are called often, such as functions to read parameters or other static data.
- Functions that are called often because of a badly written application. For example, if 10 orders (each with five lines) must be updated and certain functions are called 500 times, check to identify the reason these functions are called often.

Note: Every function that is not called improves the performance, because the child functions are not called.

- It is recommended that you identify areas where performance can be improved. The 'percentage' column helps to determine the time that can be gained by not calling a function.

BAAN_SQL_TRACE

This trace variable gathers information from the bshell. All the application queries sent to the database can be logged, except DML and DLL queries. Queries that are generated by the standard program can be recognized because the SQL statements are printed in uppercase. Queries that are built partly by the standard program are printed in both uppercase and lowercase; the uppercase statements are usually generated by the standard program/query extensions. The lowercase statements in the queries are added by the application developer.

If the BAAN_SQL_TRACE is used without specifying a file, the output is sent to the bshell.<pid> file in the BSE_TMP directory. This file is removed when the bshell is closed. Therefore, it is recommended to write the trace in a separate file, by specifying a file name in the command line, using the following option:

```
-keeplog -logfile <filename> -set BAAN_SQL_TRACE=<value>
```

The 'keeplog' option prevents the log file from being removed from the system when the bshell is closed.

Apart from setting the BAAN_SQL_TRACE as an environment variable, you can also set the variable from the option dialog. Using this option, you can switch on tracing just before the actual tracing is required; this can be stopped after the trace. The settings can be configured in the 'Debug Bshell' section of the option dialog on the **BDB/SQL Tracing** tab.

These BAAN_SQL_TRACE options are currently valid:

- 0000001 Show parse tree
- 0000002 Show exec tree
- 0000004 Show filter details of exec tree
- 0002000 Show calls of internal SQL functions and bind variables

Combinations of these options are allowed by adding the options using an octal system. For example:

```
-set BAAN_SQL_TRACE=2007
```

For the first trace, the value 2000 provides sufficient information. It is recommended that you use the Call Graph Profiler because the output of BAAN_SQL_TRACE is difficult to read. An example of the output with BAAN_SQL_TRACE=2000:

```
================================================
[1] SQLStatement::Constructor
[1] SQLStatement::Prepare( SessionId=1 Mode=4 )
StatementText:
select  ttaad001.*
        from    ttaad001
        where   ttaad001.seqn = 1
        and     ttaad001._compnr = 0
        as set with 1 rows

[1] SQLStatement::Prepare() returns 0
[1] SQLStatement::Exec( Compnr=0 )
[1] SQLStatement::Exec() returns 0 nrows 0
[1] SQLStatement::Fetch()
[1] SQLStatement::Fetch() returns 100
[1] SQLStatement::Break()
[1] SQLStatement::SoftClose()
[1] SQLStatement::SoftClose() returns 0
[1] SQLStatement::Break() returns 0
)================================================
```

BAAN_SQL_TRACE with -dbgflow

If it is not possible (for example, the object is closed with a kill command) or difficult to use profiling, it becomes difficult to identify the location of application queries that perform badly. One possible option is to use the BAAN_SQL_TRACE=2000, in combination with the bshell trace command -dbgflow. The ‘dbgflow’ option writes every change of an active object or DLL in the trace file. Therefore, when a bad performing query is found in this trace, such as with BAAN_SQL_TRACE=2000, you can find the object that contains the query.

It is recommended that you log the changes in a separate file by specifying a file name in the command line using this option:

```
 -dbgflow -keeplog -logfile <filename> -set BAAN_SQL_TRACE=2000
```

Example of the output of BAAN_SQL_TRACE:

```
B:0000022:::(00001):Flow: -->>  (depth 01):    main() (object ottstpstdlib)
B:0000023:::(00001):Flow: -->>  (depth 02):     get.multibyte.factor() (object
ottstpstdlib)
B:0000024:::(00001):Flow: <<--  (depth 02):     get.multibyte.factor() (object
ottstpstdlib)
B:0000025:::(00001):Flow: -->>  (depth 02):     init.logfin.comp() (object
ottstpstdlib)
[1] SQLStatement::Constructor
```

```
[1] SQLStatement::Prepare( SessionId=1 Mode=4 )
StatementText:
select  ttaad001.*
        from    ttaad001
        where   ttaad001.seqn = 1
        and     ttaad001._compnr = 0
        as set with 1 rows

[1] SQLStatement::Prepare() returns 0
[1] SQLStatement::Exec( Compnr=0 )
[1] SQLStatement::Exec() returns 0 nrows 0
[1] SQLStatement::Fetch()
[1] SQLStatement::Fetch() returns 100
[1] SQLStatement::Break()
[1] SQLStatement::SoftClose()
[1] SQLStatement::SoftClose() returns 0
[1] SQLStatement::Break() returns 0
```

When a new function is called, the same is written in the trace file:

```
B:0000023:::(00001):Flow: -->>  (depth 02):     get.multibyte.factor() (object
ottstpstdlib)
```

This get.multibyte.factor() is the new function to be called in object ottstpstdlib.

When a function is not used, this is written to the trace file:

```
B:0000024:::(00001):Flow: <<--  (depth 02):     get.multibyte.factor() (object
ottstpstdlib)
```

All trace information that is performed in-between these traces is implemented in the DAL object or DLL object. In the example, the query is located in the code of ottstpstdlib.

<DB>PROF

When using the <DB>PROF variable, the queries from the driver are sent to the database and the timestamps can be traced. The string '<DB>' must be replaced with the used abbreviation for the used database.

Depending on the used database driver, these traces are available: DB2PROF, MSQLPROF, and ORAPROF.

The value specified for the <DB>PROF variable is the threshold (in seconds). Each query takes additional time, or equals the value, is reported, for example, when the setting is:

```
ORAPROF=4.0
```

All queries that take longer than four seconds are reported. To obtain all the queries, use this setting:

```
ORAPROF=0.0
```

The <DB>PROF variable can be set similar to other environment variables in the command line:

```
-set ORAPROF=0.0
```

The <DB>PROF variable can also be used in the db_resource file. The variable name is the same, but in lowercase:

```
oraprof:0.0
```

The output of <DB>PROF is the same as the variable used but the output file is in lowercase. Using ORAPROF generates the file ‘oraprof’ The output file is created in the directory where the bshell is started. The new data is added to an existing file.

DB2PROF

The DB2 UDB driver writes the timing information for each application action (parse/exec/fetch) that takes longer than the time (in seconds) defined by the DB2PROF.

This db2prof output is obtained by running these commands:

```
export DB2PROF=0.0
bsql6.2 –c090 –q "select item,dsca from tcibd001 where seak='COST ITEM'"
```

The output displays the various steps in the execution of the query and the time required for each step.

No performance problems are detected when this query is executed:

```
2011-06-10[12:53:24]: Profiling value = 0.0000 sec
------ Profiling value exceeded -----
<root><bsql_00>:2011-06-10[12:53:25.819]:
Time (parse) : 0.000022 seconds
SELECT a0.t_item,a0.t_dsca FROM FP6BKIT.ttcibd001090 a0  WHERE a0.t_seak = CAST(? AS
VARCHAR(9))   OPTIMIZE FOR 5 ROWS
-------------------------------------
```

There are comprehensive methods to obtain tracing and to explain information from DB2 UDB: CLI trace, Dynamic SQL Snapshot, and Explain Output. Usually, these tracing facilities produce detailed information of Infor LN sessions that are running, including the time spent in the application and the processing time on the DB2 UDB server. Overall, these DB2 tracing tools require a higher level of analysis to deliver a detailed and accurate analysis of the running system.

ORAPROF

The Oracle driver writes timing information for every application action (parse/exec/fetch).
The differences between tracing with ORAPROF and tracing with SQL_TRACE:

- SQL_TRACE output is smaller.
- SQL_TRACE output can be formatted with the included tools.
- Oracle tracing works with a precision of 1/100 sec, and ORAPROF is precise.
- Difficult to trace parallel processing with ORAPROF because all output is directed to the same file.

- Timestamps with ORAPROF include the round-trip time between the driver and database. In case of expected network problems, a comparison between the two output files is recommended.

Example of the ORAPROF output:

```
2011-06-10[12:53:24]: Profiling value = 0.0000 sec
------ Profiling value exceeded -----
<root><bsql_00>:2011-06-10[12:59:10.342]:
Time (parse) : 0.000000 seconds
SQL statement:
SELECT /*+ FIRST_ROWS index(a0 ttcibd001603$idx2) */ a0.t$item,a0.t$dsca FROM
baan.ttcibd001603 a0 WHERE a0.t$seak = :1
-------------------------------------
```

This output is generated using this command:

```
ORAPROF=0.0 bsql6.2 -q "select item, dsca from tcibd001 where seak =   'COST ITEM'"
```

The output displays the various steps of the query: parse, execute, and fetch. For each step, the time is measured.

MSQLPROF

The Microsoft SQL Server driver writes timing information for each application action (parse/exec/fetch) that takes longer than the time (in seconds) defined by the MSQLPROF.

The MSQLPROF output is obtained by running these commands:

```
set MSQLPROF=0.0
bsql.exe –q "select item,dsca from tcibd001 where seak='COST ITEM'"
```

The output displays the various steps in the execution of the query and the time required for each step.

```
2011-06-10[12:53:24]: Profiling value = 0.0000 sec
------ Profiling value exceeded -----
<baan><?>:2011-06-10[12:53:25.819]:
Time (parse) : 0.000000 seconds
SQL statement:
SELECT a0.t_item,a0.t_dsca FROM dbo.ttcibd001090 a0 WITH ( INDEX ( Ittcibd001090_2a ))
WHERE a0.t_seak = ?  OPTION (FAST 10)
-------------------------------------
```

There are comprehensive methods to obtain tracing and to explain information from the Microsoft SQL Server such as MSQL Query Analyzer (SQL Server Management Studio in Microsoft SQL Server), SQL Profiler, Extended Events, Server-Side Traces, Trace Flags and others. Usually, these tracing facilities produce detailed information of Infor LN sessions that are running, including the time spent in the application and the processing time on the Microsoft SQL Server Database. Typically, these Microsoft SQL Server tracing tools require a higher level of analysis to deliver a detailed and accurate analysis of the running system.

DBSLOG

The DBSLOG variable provides detailed debugging information about the online processing of the driver. The information is logged in the dbs.log file in the driver's current directory.

This trace option is usually not used for performance tracing, but there are two important features for tracing the Infor LN application for performance:

- You can fetch the value of used variables. To obtain the queries and the used variables, you can use DBSLOG=500.
- The number of retries and measure the duration of a lock with the value of 100000 and 200000 can be logged. Tracing on locks and retries can be useful in environments in which processes do not scale as expected or other locking issues are bound to occur.

The output of DBSLOG is generated in the current directory; for BW users, this is the home directory. The default file name is dbs.log unless specified otherwise by the DBSLOG_NAME variable.

Other values of the DBSLOG are:

- 0000001 Data Dictionary information of tables within the driver.
- 0000002 Query info (SQL Level 1).
- 0000004 Query plan info (SQL Level 2).
- 0000010 Row action information.
- 0000020 Table action information.
- 0000040 Transaction action information.
- 0000100 DBMS input/output data (SQL Level 2).
- 0000200 Administration file info (SQL drivers).
- 0000400 DBMS SQL statements.
- 0001000 General debug statements.
- 0002000 Query processing info (for BAAN_SQL_TRACE info).
- 0004000 Data buffering info (communication).
- 0100000 Lock retries logged (includes session name).
- 0200000 Logs duration and tablename of the first lock set in a transaction. Use DBSLOG_LOCK_PROF=n to set a threshold.

To define multiple categories, add the octal values. These values are checked bit wise to determine if a specified category must be logged.

To trace all locks longer than a specific time, specify the DBSLOG_LOCK_PROF variable. To prevent additional logging, it is recommended that you set the value of the variable to 0.5 seconds. Additionally, to redirect the output of the trace, specify the DBSLOG_NAME variable.

You can set the DBSLOG variable in the BW command field, similar to all other environment variables. For example, to log all SQL statements and DBMS input variables and output data, use:

```
-set DBSLOG=500
```

The diagram shows an example output of DBSLOG=500:

```
<63439014> 2011-06-28[11:06:53]: Logging started mode 0500
        ---- LOG DBMS  INFO [0000100] ----
        ---- LOG SQL   INFO [0000400] ----

SQL> SET CURRENT QUERY OPTIMIZATION = 0

SQL> SELECT a.t_pacc,a.t_keyr,a.t_desc,a.t_Refcntd,a.t_Refcntu FROM
FP6BKIT.tttadv999000 a WHERE a.t_pacc=? AND a.t
_keyr=?  OPTIMIZE FOR 5 ROWS   [1105734f0]  t_pacc [1105734f8]  t_keyr [110229c9a]
[110229c90]  [110229c89]  [1102
29c82]  [110229c7b]
----- DBMS Where Input ----
Bind nr 1 : pacc       : [1105734f0] string  : db_ind   8 : db_size   8 : '        '
Bind nr 2 : keyr       : [1105734f8] string  : db_ind  22 : db_size  22 : 'moduttdsk
'
Fetch : end of set.
----- DBMS Where Input ----
Bind nr 1 : pacc       : [1105734f0] string  : db_ind   8 : db_size   8 : 'b61au   '
Bind nr 2 : keyr       : [1105734f8] string  : db_ind  22 : db_size  22 :
'progttdskbbrowser   01'
----- DBMS Output Row ----
Bind nr 1 : pacc       : [110573520] string  : 'b61au   '
Bind nr 2 : keyr       : [110573528] string  : 'progttdskbbrowser   01'
Bind nr 3 : desc       : [11057353e] string  : '0ottdskbmbrowser|a1|||1||||||
             '
Bind nr 4 : Refcntd    : [110573590] long    : <0>
Bind nr 5 : Refcntu    : [110573598] long    : <0>
```

Caution: It is not recommended to set trace variables such as DBSLOG in the db_resource file, as this generates additional data for every user, which impacts performance.

Tracing with bsql

A tool in the porting set is called bsql and can be used to start queries. When a bad performing Infor LN query is identified, the query can be tested using this tool. This tool can only be started from the command prompt.

The options for performance tracing are:

- -f <QueryFile>: The bsql processes the query from the specified file.
- -o <OutputFile>: Redirects the output to the specified file.
- -c <Company>: Performs the query in the specified company.

If a query with a problem is detected with the Call Graph Profiler, the query can be copied to a file. The variables must be replaced with real values; A DBSLOG trace can provide the values of the variables for a query. The value DBSLOG=500 can be used to provide queries and variable values. BAAN_SQL_TRACE=2000 can also be used to obtain information about the bind variables.

If multiple rows are expected as output, usually a dummy file must be used. For Windows systems, the NUL file has no disk bottlenecks; for UNIX, the /dev/null device has no disk bottlenecks. If the output is required, it is recommended that you use a disk that operates faster on which to store the output file.

An example of bsql usage:

```
bsql –f bad_query –o NUL –c 100
```

The bsql command can be used with environment variables or system tools:

```
$ time ORAPROF=0.0 SQL_TRACE=true bsql6.2 –f bad_query –o /dev/null –c 100
```

In the example, bsql processes the query from the bad_query file and stores the output in /dev/null. The query is started in company 100. The UNIX ‘time’ command is used to calculate the CPU usage and wall clock time for the bsql command. To locate the Oracle SQL statements, the ORAPROF and SQL_TRACE variables are added.

Choosing the correct tracing method

The Call Graph Profiler is easy to use and therefore, it is recommended that you use the same for performance traces. This tool can analyze most of the performance problems for specific sessions.

If the Call Graph Profiler is not sufficient and it is not clear why a query takes a certain amount of time, database tracing can be added. See the database specific Performance Tracing and Tuning Guide for details.

See InforXtreme knowledgebase article 1660883 to Investigate locking issues.

<DB>PROF and BAAN_SQL_TRACE output is usually too large and difficult to interpret; therefore, these must only be used when required. For example, if there is a problem in the database driver or network.

Before you run traces for all tests, it is recommended that you isolate the problem as much as possible. If a session that calculates projects performs badly during one project, it is recommended that you trace only that project instead of all projects. Isolating the problem help you identify the bottleneck faster.

Chapter 7 Performance Wizard

7

This chapter helps you analyze and solve performance problems quicker.

Session related

Performance problems can be session related. In this section, situations that can occur are described.

Session takes long to start for the first time

Normally, when a session is started for the first time, the objects must be loaded in the memory and the database cache must be filled. This is the reason a session takes longer to start, when compared to starting the session the second time. Possible reasons when a session takes more than 20 seconds to start the first time:

- It takes a lot of time to start the database connection.
  - Check if the login when using database specific tools such as Oracle sqlplus is fast.
  - Check if the first Infor LN query sent to the database takes more time than expected. The easiest method to do this is using the Call Graph Profile in combination with a database trace. Check the first application and database query generated in the starting session.
- Objects/data definitions are not loaded in shared memory. It is recommended to store frequently used objects and reports in the shared memory.

You must also:

- Check the CPU, memory and disk I/O usage of the system.
- Check the amount of memory available for the database.

All sessions are slow even when used by few users

If all sessions are slow even when used by a few users, check these:

- Check if the system is not heavily loaded. It is possible that the system includes ‘run away’ processes or a backup. To view the processes that run on the system and the amount of CPU

power consumed, use system tools. Also check the disk utilization. If any of these problems are found, you must first resolve these issues.

- Check if the connection to the database is slow. Check the Call Graph Profiler output to identify where the time is consumed. If the CGP output shows that simple queries are slow as well, it is recommended to check the database connection and verify if the database is tuned to optimal.
- To identify a connection or database problem, it is recommended to run (besides the CGP) the database trace such as SQL_TRACE for Oracle. If the timing between the application and the database tracing differs, the problem may be caused by the network connection.

Some sessions are slow

Slow sessions can be caused by locking, bad performance of the session, wrong expectations of the session, or misconfigured application settings. Check these scenarios:

- If a session is slow even when no other session is running on the system, the session must be traced as described in the **Particular (batch) session is slow** section.
- If the session runs faster only when no other session is running on the system, this can indicate that locking problems are causing the performance degradation. It is recommended to check the database for queries waiting for a releasing lock. See InforXtreme knowledge base article 1660883 to Investigate locking issues.

Particular (batch) session is slow

Depending on the possibilities, if a specific session is slow, the following traces can be made:

- Call Graph Profiling
- Database trace

For more information about tracing, see chapter 6, ‘Tracing Infor LN’.

Note: Application settings and the data that must be processed have an important effect on performance.

System related

Performance problems can be system related. In this section, situations that can occur are described.

System uses 100 percent CPU

If the system uses 100 percent of the CPU, this is not necessarily a problem. This is just the optimal use of the available resources. However, this can become a problem when the load on the system is high, and the response time to the user increases to an unacceptable level. System utilities such as perfmon on Windows and sar or vmstat on UNIX can be used to analyze the system.

System is not CPU bound but sessions are slow

If the system uses less than 80% of the CPU but sessions are still slow, it is important to check which sessions are slow and the related circumstances. Also check memory and disk I/O.

Note: Memory pressure can cause high disk I/O on a specific disk, as the system starts paging out memory pages to the disk.

See ‘Some sessions are slow’ in Chapter 7, Performance Wizard.

Process related

Performance problems can also be process related. In this section, situations that can occur are described.

Bshell is consuming most of the CPU time

If an OLTP bshell is consuming > 70% CPU of a processor core for more than 20 seconds, there can be a problem. Possible reasons for a bshell process consuming most of the CPU time are:

- Table boosters are used. These are the tables loaded in bshell memory. Reading from these tables can lead to usage of additional CPU time on the bshell, particularly when table boosters are activated with the ‘Incremental Load’ feature. Check if table boosters are used to the optimum, see ‘Table boosters’ section.
- A lot of sorting is performed. Some functions must sort data using one method or another. With profiled objects, you must check which functions consume most of the time and the reason for sorting. Usually, sorting is a CPU/disk intensive task.
- Sorting is performed in the database driver. In certain situations, it is possible that data must be sorted in the driver. If sorting requires more than a certain amount of memory, disk space is allocated for the sort process. Check if huge/many sort files, which start with qp, exist in the BSE_SORT directory. If sorting causes performance degradation, the problem function/query can be found with profiled objects. Try to eliminate driver sorting as much as possible.
- Re-parsing of queries.

Database is consuming most of the CPU time

The possible reasons can be:

- Database is not tuned optimal. See the database specific *Performance, Tracing and Tuning Guide*.
- Check if parsing is consuming time in the database. If parsing is done often, check if the time can be reduced by tuning the database or changing driver settings.
- Sorting can lead to additional usage time by the database. Check if the queries force the database to perform additional sorting.
- Bad queries. This can be checked by running the Call Graph Profiling or database tracing.

Appendix A Performance Checklist

A

The following checklist contains the topics that must be checked on every Infor LN configuration before analyzing a performance problem:

1 Tunables

- Which environment variables are set and do they have the correct value?
- Which variables are set in the db_resource file and do they have the correct value?

2 Table definitions

- Do all tables use the same driver?
- Do all tables use the same environment variables?

3 Auditing

- Are all tables specified for auditing required?

4 Shared tables

- Are the shared tables grouped optimal?

5 Authorizations

- Have Infor LN authorizations been used efficiently?
- Have database authorizations been used efficiently? Are field authorizations used?

6 Shared memory usage

- Are the correct package combinations, objects, and reports loaded in shared memory?

7 Software

- Is a recent porting set used?
- Is a recent version of Enterprise Server used?
- Are there any specific fixes available for a slow performing session?

8 Log files

- Do the (system) (error) log files grow rapidly? Any messages in the Windows' event viewer?
- Do the Infor LN database driver or bshell log files show (performance) problems?

9 Configuration

- Is the Infor LN environment configured optimal for the usage of OLTP vs. Batches?
- Is the maintenance window separated from the backup and batch window?
- Are the database maintenance (indexes/statistics) running regularly?

Appendix B Format Trace Output

B

Using the scripts listed in this appendix, the output of a BAAN_SQL_TRACE=0200 trace can be formatted to be read better. The output generates the following columns:

- **QID**: the Query ID number corresponds to a query at a later stage in the output.
- **Count**: The number of fetches done on this query.
- **Time**: The time required for the total amount of fetches.
- **Avg**: Average Time per Fetch (= Time/Count).
- **First**: Displays the time required for the first fetch.

An example of the output:

```
QID  Count Records    Time  First
  19      3      18    3.02   0.02
  28      2       4    2.01   0.01
  33      1      13    1.86   0.86
  26      1       6    1.03   0.03
  27      1       3    1.02   0.02
```

After the QIDs and the times, the queries are generated similar to the original output:

```
Query : 33
SELECT tcibd001.*, tcmcs001.tccu, tcmcs003.nwrh, tcmcs048.cref,tcmcs052.plnk
FROM tcibd001,  tcmcs001, tcmcs003, tcmcs048, tcmcs052
WHERE  {tcibd001.item} >=  {:tcibd001.item} AND (tcibd001.cwun refers to tcmcs001
and tcibd001.cwar refers to tcmcs003
and tcibd001.cpcp refers to tcmcs048
and tcibd001.cprj refers to tcmcs052)
ORDER BY tcibd001._index1

Query : 34
SELECT ttadv999.*
WHERE ttadv999._index1 > { :1, :2 }
ORDER BY  ttadv999.pacc ASC,  ttadv999.keyr ASC
AS SET WITH 3 ROWS
```

The script:

```
if [ $# -ne 2 ];then
   echo "Usage $0: <inputfile> <outputfile>"
   exit
fi

Awk=awk
[ -x /usr/xpg4/bin/awk ] && Awk=/usr/xpg4/bin/awk
[ -x /usr/bin/nawk ] && Awk=/usr/bin/nawk

$Awk '
```

```
/^Fetch times of Query/ {
   getline
   getline
   i=0
   while (substr($1,1,5)!="-----") {
      if ($1!="")
         InSQL[++i]=$0
      getline
   }
   for (j=MaxQID;j>0;j--) {
      for (k=i;k>0;k--)
         if (InSQL[k]!=SQL[j,k])
            k=-1
      if (k==0)
         j=-j
   }
   if (j==0) {
      MaxQID++
      QID=MaxQID
      for (j=1;j<=i;j++)
         SQL[QID,j]=InSQL[j]
      Lines[QID]=i
   }
   else
      QID=-j-1
   getline
}
/^Nr Rows Fetched/ {
   LastFetch=$5
   Fetch[QID]+=$5
   Count[QID]++
   getline
}
/^Fetch Time for 1st Row/ {
   First[QID]+=$7
   getline
}
/^Max Fetch Time/ {
   Max=$5
   getline
}
/^Average Fetch Time/ {
   AvgTot=LastFetch*$5
   getline
}
/^Average Fetch Time/ {
   if ($7+Max>AvgTot) {
      Tot[QID]+=$7+Max
   }
   else
      Tot[QID]+=AvgTot
   Tot[QID]+=1
   getline
}
END {
   for (i=1;i<=MaxQID;i++) {
      Q[i]=i
```

```
      for(j=i;j>1 && Tot[j-1]<Tot[j];j--) {
         swap(Count, j)
         swap(Fetch, j)
         swap(First, j)
         swap(Tot, j)
         swap(Q, j)
      }
   }
   print " QIDCount Records    Time  First"
   for (i=1;i<=MaxQID;i++)
      printf("%4d %6d %7d %7.2f %6.2f\n",
         Q[i], Count[i], Fetch[i], Tot[i], First[i])
   print
   for (i=1;i<=MaxQID;i++) {
      print "Query : " i
      for (j=1;j<=Lines[i];j++)
         print SQL[i,j]
      print
   }
}
function swap(A, i) {
   t=A[i-1]
   A[i-1]=A[i]
   A[i]=t
}' $1 > $2
```

Appendix C Application Response Time Measurement

C

The appendix describes the process to measure the transactions using Application Response Time Measurement (ARTM). Infor LN ARTM is an implementation of the ARM interface.

Examples of ARTM are:

- Response times when processing an order.
- Response times when searching for an account.

For example, an ARTM is useful when Service Level Agreements are defined for response times of critical Infor LN sessions. The ARTM data can be sent to an external tool that supports standard ARM calls such as HP Glance or IBM Tivoli. These tools are capable of presenting the data in several ways to the end-user. There are agents for both products for several platforms, so customers also running on platforms other than HP and IBM can be supported.

ARTM is designed for system administrators to monitor the overall user response time of the Infor LN system and the specific functionality within a specific application. The data can be analyzed at a later stage.

ARTM helps the IT manager to meet service level agreements and helps the system administrator to detect slow responses to users and identify the bottlenecks in an installed Infor LN system.

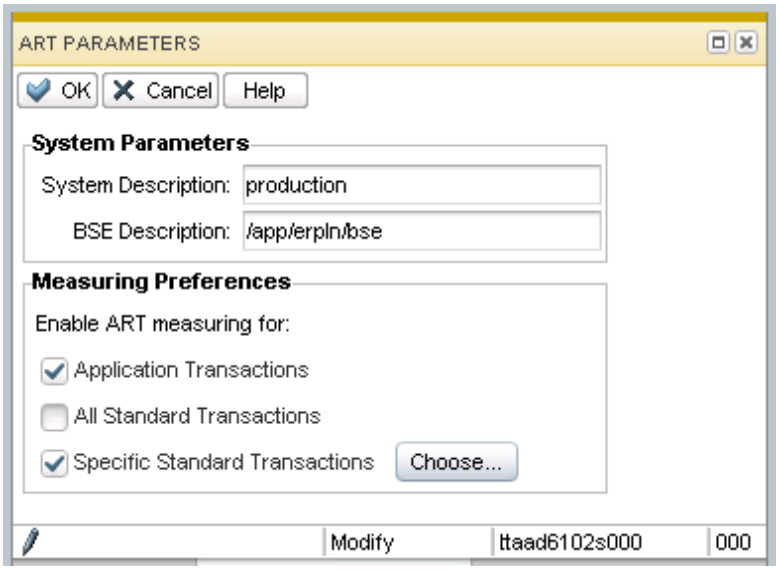
ARTM is not a tracing system that generates huge amounts of data. The information provided by ARTM must be kept to a minimum, so the system administrator has an overview of the responsiveness of the system. Based on tests, the overhead is minimal, less than 0.5%.

This implies that transactions must be implemented on a high level, starting when a user action begins, and ending when the user action is finished. The default transactions can be logged and you can also create a logging by 4GL code.

See the Infor LN online help for more information.

Setup

The parameters for ARTM can be specified, in the ART Parameters (ttaad6102s000) session:



The System parameters:

- System description: Name of the system, such as Production.
- BSE description: BSE, such as /app/inforln/bse.

Additionally, you can measure:

- Application transactions: These are the custom programmed transactions. You must select this option.
- All Standard transactions: Traces all standard transactions in all sessions. This measurement generates a lot of data and this option is usually cleared.
- Specific Standard transactions: This option can be used instead of ‘All Standard Transactions’. This allows the user to specify transactions that must be measured. To select a specification click **Choose**.

By default, the ARTM calls are not sent to the ARM agent. The art_enable parameter must be set to activate the ARTM. These parameters can be used to enable ARTM:

- BAAN_ART_ENABLE / art_enable

Valid values for BAAN_ART_ENABLE/ art_enable are 1 (enable) or 0 (disable = default). Specifying 'art_enable:1' in $BSE/lib/defaults/all will enable the ARTM calls for all users. Specifying this resource in the $BSE/lib/users/u<user> file enables the ARTM calls only for this user. The environment variable is useful for testing.

- BAAN_ART_USER / art_user

The BAAN_ART_USER/ art_user variable specifies the name of the user to which the transactions are registered. By default, this name of the user is 'BAAN'; for example, all transactions are registered under the same name. If specific users must be monitored separately, this can be configured by specifying this variable in the BW configuration file or the $BSE/lib/user/u<user> file.

Programming own transactions

Sometimes, more than the default information is required. For example, by default, a processing session displays the amount of processed orders to ARM. This additional information is also required when the SLA defines that each order takes 10 seconds to be processed. Additionally, an order line with 100 order lines takes more time than an order with 10 lines. This functionality can be included using the following functions:

- artm.define.transaction.class
- artm.redefine.transaction.class
- artm.begin.transaction
- artm.update.transaction
- artm.end.transaction

An example of the code used for these functions:

```
long   nr.orders, nr.lines
long   defined.tra.class.id, tra.id, i, ret
       |* Redefine the transaction belonging to
       |* form command "Process orders"
       defined.tra.class.id = artm.define.transaction.class(
               "Process Orders", "Order Processing Session",
               ART.COUNTER, "Order Count",
               ART.COUNTER, "Order Line Count")
        |* User has started order processing.
        |* Start the transaction
        tra.id = artm.begin.transaction(
                defined.tra.class.id,
                ART.COUNTER, 0,
                ART.COUNTER, 0)
        for i = 1 to 10
                |* process.order sets nr.orders and nr.lines
                process.order()
                |* Give feedback about the order processed
                ret = artm.update.transaction(
                        tra.id,
                        ART.COUNTER, nr.orders,
                        ART.COUNTER, nr.lines)
        endfor
        |* All orders are processed. End transaction.
        ret = artm.end.transaction(
                tra.id,
                ART.TRANSACTION.SUCCESS,
                ART.COUNTER, nr.orders,
                ART.COUNTER, nr.lines)
}
```

See the *Programmers Manual*.

Tracing and debugging

Tracing and debugging can be enabled using the BAAN_ART_TRACE / art_trace variable.

This variable can be used to trace ARM calls. By default, the tracing is off (0) and can be started by specifying a value. The valid values are:

- 1: Basic tracing of each call to the ARM library.
- 2: Basic tracing and the binary blocks sent to the ARM library.
- 3: Does not bind the ARM library, but uses internal stub functions. Application developers who do not have the ARM library installed can use this to for basic tracing of their ARM calls.
- 4: Same as 3, but includes the binary blocks.

By default, the output of the trace is sent to the $BSE/tmp/bshell.<pid> file. The trace made by BAAN_ART_TRACE=5 is useful to test ARTM. An example of the output:

```
B:0000000:::(00001):arm loaded: Baan internal ARM tracing stubs binded
B:0000001:::(00001):arm_init called (time stamp 0)
B:0000002:::(00001):    application name   = BaanERP
B:0000003:::(00001):    user name          = baan
B:0000004:::(00001):    flags              = 00000000
B:0000005:::(00001):    data pointer       = 0
B:0000006:::(00001):    data size          = 0
B:0000007:::(00001):    returned global id = 10000000

B:0000016:::(00007):arm_getid called (time stamp 13990)
B:0000017:::(00007):    application id     = 10000000 (268435456)
B:0000018:::(00007):    transaction name   = cprrp0520m000/Read
B:0000019:::(00007):    transaction detail = Read record(s)
B:0000020:::(00007):    flags              = 00000000
B:0000021:::(00007):    data pointer       = 8044A1C
B:0000022:::(00007):    data size          = 344
B:0000023:::(00007):    returned class id  = 10020000

B:0000030:::(00007):arm_start called (time stamp 55476)
B:0000031:::(00007):    class id           = 10020000 (268566528)
B:0000032:::(00007):    flags              = 00000000
B:0000033:::(00007):    data pointer       = 8044A80
B:0000034:::(00007):    data size          = 256
B:0000035:::(00007):    returned trans id  = 10020002

B:0000036:::(00007):arm_stop called (time stamp 55486)
B:0000037:::(00007):    application id     = 10020002 (268566530)
B:0000038:::(00007):    flags              = 00000000
B:0000039:::(00007):    data pointer       = 8044A80
B:0000040:::(00007):    data size          = 256
B:0000041:::(00007):    returned           = 10020002
```

The output shows:

- Block 1: Initialization of ARM.
- Block 2: The Item Order Plan – Find (cprrp0520m000) session is started and the Read call is initialized.
- Block 3: The Read option is activated (start time = 55476). Time is in milliseconds.

- Block 4: The Read option is finished (end time = 55486). Therefore, the total duration of the find is 55486 – 55476 = 10 ms.

The relation between the different calls can be monitored by the application, transaction, and class ids.