Deployment Considerations

QAD .NET User Interface

May 2006
Revision 1

QAD Inc.
6450 Via Real
Carpinteria, California 93013
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents

CHAPTER 1

Introduction

This document outlines general concepts relating to QAD .NET User Interface deployment options and specific steps that system administrators can take to enhance the performance of the product. Although much of the material here can guide any administrator to making system improvements, some of the solutions mentioned are best implemented by experienced personnel.

This document is a good starting point for understanding deployment and performance issues. For complex systems in need of performance optimization, it is strongly recommended that QAD Global Technical Services be employed to assess how best to improve system performance.

What is in this Guide?

Audience

Symbols and Conventions

What is in this Guide?

Use this guide for general information on the various setup and deployment options with the QAD .NET User Interface (UI). This guide is also a useful tool for understanding sizing and hardware selection options.

This document is intended as a supplement to *Installation Guide: QAD User Interfaces*.

Although the nature of the information contained here may be extended to commercial UNIX releases, testing and documentation have initially been carried out using the following environment:

- Red Hat Enterprise Linux (Advanced Server AS 4.0) Server
- Microsoft Windows XP Clients

All testing was conducted using the Progress database engine, and using a real customer database.

Audience

These instructions are intended for a system administrator with experience installing MFG/PRO and QAD Desktop, as well as configuring and managing hardware and operating system software. This person also should have a good understanding of networking concepts and administration, as well as a basic understanding of performance concepts.

If you do not have this expertise within your company, you should contact your QAD Support representative for information on the installation and capacity requirements planning offerings supplied by QAD's Global Technical Services.

Related Documentation

Progress Documentation

Find the complete Progress Open Edge documentation set online at the Progress Web site:

http://www.progress.com

MFG/PRO Documentation

For information on installing MFG/PRO or converting to a more recent release, refer to the appropriate installation or conversion guides for your system. MFG/PRO installation guides are not included on a CD. Printed copies are packaged with your software. Electronic copies of the latest versions are available on the QAD Web site.

For information on using MFG/PRO functions, refer to the User Guides.

For instructions on navigating the QAD .NET UI, refer to *User Guide: QAD .NET User Interface*.

For QAD customers with a Web account, the complete MFG/PRO documentation is available for review or downloading at: http://support.qad.com/

Register for a QAD Web account by accessing the Web site and clicking the Accounts link at the top of the screen. Your customer ID number is required. Access to certain areas is dependent on the type of agreement you have with QAD.

Features of the Web site include an online solution database to help you answer questions about setting up and using the product. Additionally, the QAD Web site has information about training classes.

Symbols and Conventions

This document uses the text or typographic conventions listed in the following table:

If you see:	It means:
`monospaced text`	A command or file name; for example `compile.lst`.
`italicized monospaced text`	A variable name for a value you enter as part of an operating system command; for example, *`QAD UISystemName`*.
`indented command line`	A long command that you enter as one line, although it appears in the text as two lines.
Note	Alerts the reader to exceptions or special conditions.
Important	Alerts the reader to critical information.
Warning	Used in situations where you can overwrite or corrupt data, unless you follow the instructions.

CHAPTER 2

Performance Concepts

This section covers basic performance tuning concepts and system performance management.

Definitions

What is Good Performance?

Managing Performance

Definitions

Many of the concepts discussed in this guide rely on a basic understanding of common performance-related definitions. Some of the more widely used definitions are listed here.

Average response time

The average time to return a response to the client.

90% response time

The time within which 90% of all responses are returned to the client.

Load

The degree to which a computing resource is used. In this document, load is expressed as a decimal value. When the load on a server starts to increase beyond acceptable thresholds, noticeable generalized performance degradation begins to occur. This typically manifests itself to the users of the server in the form of decreased responsiveness, and decreased throughput (transactions take longer to finish). The load is a derivation from the number of processes blocked because they are waiting for CPU resources to become available.

Paging

A process requires real-memory pages to execute. When a process references a virtual-memory page that is on disk, because it either has been paged-out or has never been read, the referenced page must be paged-in and, on average, one or more pages must be paged out (if replaced pages had been modified), creating I/O traffic and *delaying the response of the process*.

The operating system attempts to borrow real memory from pages that are unlikely to be referenced in the near future, through the page-replacement algorithm. A successful page-replacement algorithm allows the operating system to keep enough processes active in memory to keep the CPU busy. But at some level of competition for memory, no pages are good candidates for paging out to disk because they will all be reused in the near future by the active set of processes. This situation depends on the total amount of memory in the system and the number of processes.

Scalability

A measure of how well a system performs as additional users and processes are added.

Throughput

The amount of transactions that can be processed in a given time period.

What is Good Performance?

User Expectations

Software users want to complete software tasks (for example, creating a sales order in an ERP system) with as much efficiency as possible, and, typically, as quickly as possible. A user perceives software as performing well when the software responds at least as quickly as the user expects it to.

There is a large variance in what users accept as good system response. When entering data, most users find performance acceptable if field submits (that is, any action within a program screen that requires pressing the Enter key) take no longer than a second or so. Reports and programs that display large amounts of data can take several seconds to display, and this is generally acceptable for users.

Performance tuning begins with setting proper goals and understanding user expectations. The QAD .NET UI, when properly configured and when powered by adequate hardware, can meet most user performance expectations.

Another aspect of system performance is how well the system handles additional users; that is, how well the system scales. The QAD .NET UI architecture has been designed to allow its components to be distributed (tiered) in order to promote exceptional scalability.

Technical Expectations

Performance measurement of software is a complex and broad subject. A user interacting with an application such as the QAD .NET UI in real time has very different performance needs than a system administrator running batch processes.

The performance requirements for QAD .NET UI are likely to be determined by response time and scalability. This document focuses on these areas: optimizing response times while maintaining system stability and providing good scalability.

On a technical level, criteria such as response time, system load, and other resource utilization can be measured.

Managing Performance

Establish a Baseline

It is also important to establish baseline acceptable application performance. Once the QAD .NET UI has been configured and is performing acceptably, statistics should be gathered and kept as a reference baseline. This baseline is needed in order to identify abnormal conditions and conduct capacity requirements planning. A statement such as “90% of the responses on these screens are exceeding the maximum allowable time of 1.5 seconds” is much more meaningful in this context than “The system runs slow.”

In addition to keeping baseline data, it is important to track performance information over time to graph trends and examine system patterns to help plan for performance problems.

The types of basic performance information that should be kept are:

- System configuration (hardware information, operating system information, database startup parameters, QAD .NET UI parameters, file system layout and setup)
- Operating system performance (CPU utilization, CPU run queue lengths, disk activity, memory)
- Mitigating factors
- Application performance data (response time)

Balancing Resources

Maximizing the throughput in specific component parts (such as the database server) can result in that part consuming the majority of the server resources. In a simple deployment environment where all components exist on the same server, it could well be the case that maximizing resources to one area can minimize QAD .NET UI response times. Throughput and response time can be mutually exclusive goals. In situations like this, the issue becomes one of balancing system resources against the various application requirements.

If performance requirements cannot be met in a single-tier environment, moving some components to a separate dedicated server may be an option, or even moving the entire environment to a different server with a higher resource capacity.

Performance Monitoring Tools

On UNIX and Linux systems, it is useful to install the `sysstat` RPM (from the manufacturer's installation media or from http://perso.wanadoo.fr/sebastien.godard/) and use `sar` (system activity reporter) to monitor system performance.

CHAPTER 3

System Configuration

This section covers basic performance tuning concepts and system performance management. It also provides information on the various configuration and setup options available with the QAD .NET User Interface.

Disclaimer

Operating System

Tomcat

Database and Progress Software

QAD .NET User Interface Options

Hyperthreading Enabled CPUS

Disclaimer

The information presented in this section is informational only, and is presented as a guide. Every system is unique and may have unique configuration and setup requirements. For the initial release of this guide, only Red Hat Linux Server editions are specifically covered. Most commercial UNIX systems can be configured in a similar (but not identical) way.

Operating System

UNIX and Linux installations must typically be configured to support multi-user database systems. When installing from media, the operating system resources available are usually not sufficient to support applications such as those that QAD provides. Most of the actions in this section require superuser (root) access.

Telnet

The telnet subsystem (`in.telnetd`) may not be configured and working by default, or only Kerberos secure telnet may be available. If telnet logins are not enabled, consult your operating system installation guides on how to enable telnet.

Kernel

The kernel will probably need to have several limits raised to support QAD .NET UI.

```
/etc/sysctl.conf
```

The following changes made to `/etc/systctl.conf` will support at least a 128-user system on Red Hat Linux:

```
kernel.core_uses_pid = 1
kernel.shmmax = 2147483648
kernel.shmmni = 128
kernel.shmall = 2097152
kernel.sem = 500 32000 100 128
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
```

The kernel can be updated in real time to make these settings take effect by executing the following command:

```
# sysctl –e –p /etc/sysctl.conf
```

File Handle limits

The total number of allowable processes and open files may need to be raised. This is controlled in the file `/etc/security/limits.conf`:

```
#<domain>     <type>     <item>     <value>
#

*             soft       nproc      2047
*             hard       nproc      16384
*             soft       nofile     1024
*             hard       nofile     65536
```

Tomcat

Tomcat is the Java application server used by the QAD .NET UI. Typically the memory allocation of Tomcat must be increased to support the interface requirements.

In the Tomcat installation directory, the following file can be modified to increase the memory settings:

TomcatInstallDir`/bin/setenv.sh`

The following changes are useful for all user counts:

```
-Xmn256m
-Xss256k
-XX:-UseBoundThreads
-XX:MaxLiveObjectEvacuationRatio=20
-Dfile.encoding=utf-8
```

These are the recommended settings for various user counts:

Up to 20 users:

```
-Xms256m
-Xmx256m
```

Up to 50 users:

```
-Xms512m
-Xmx512m
```

Up to 100 users:

```
-Xms1024m
-Xmx1024m
```

Database and Progress Software

Progress Patches

You should always ensure the latest Progress patches are installed on any installation running QAD software.

Database

There are no special requirements for tuning the Progress or Oracle databases when running the QAD .NET UI. The normal database tuning techniques should be applied.

In particular:

- Ensure enough database memory is allocated (`-B` parameter).
- Ensure spin locks are used on multi-CPU systems.
- Use before-image writers, after-image writers, application page writers, and the watchdog.
- Choose an optimal database block size.

Consult the MFG/PRO installation guides for more information.

QAD .NET User Interface Options

System Design and Technical Components

For information on how the QAD .NET UI is designed and how the various technical components relate to each other, please consult *Installation Guide: QAD User Interfaces*.

Maximum Count on Browses

The QAD .NET UI provides functionality that shows the user the total number of records available in a specific browse. This is a useful feature; however, for large tables and record counts, it can be very computationally expensive. Because even 10,000 records is an extremely large number to work with, QAD has set 10,000 as the default maximum records to be counted. If the record count in the browse is above the maximum record threshold, the information displayed in the top right corner of the browse (the record count area) is amended.

This value can be made higher or lower, depending on both the user requirements and the effect the counts are having on server load. The value is set on the server, and can be overridden on individual clients. Be aware that raising the setting to (much) higher values can have an adverse effect on overall system performance.

Server Side

The file that contains the default setting is either:

```
TomcatInstallDir/webapps/qadhome/client/configs/default.xml
```

Or:

```
TomcatInstallDir/webapps/qadhome/client/configs/SystemName.xml
```

With the *SystemName* taking precedence.

The value to change is:

```
<add key="MaximumBrowseRecordsToCount" value="10000" />
```

Client-Side Override

On the individual clients, this value can be overridden. The file to modify is:

```
[QAD .NET User Interface Install]\plugins\mfgpro\plugin-config.xml
```

Maximum Record Download/Print/Export Limit

In addition to the record count maximum, there are also defaulted maximums for the number of records to download, the number of records to print, and the number of records to export to Microsoft Excel.

These values are:

```
<add key="MaximumBrowseRecordsToDownload" value="50000" />
<add key="BrowseRecordsForPrintWarning" value="10000" />
<add key="BrowseRecordsForExcelWarning" value="10000" />
```

Setting the first two values too high can cause the client computer to freeze up as it attempts to process enormous amounts of data. Because of the record limitation of Microsoft Excel, the Export to Microsoft Excel option should never be set above 65,535 records. The “maximum records to download” is a hard limit and will cause the operation to abort.

Private Read-Only Buffers

The buffer pool reduces disk I/O by enabling the database to access recently used records from memory instead of reading from disk. Once the buffer pool becomes full, the least recently used (LRU) buffers are flushed to disk. This works quite well for random database access.

However, heavy sequential database access can monopolize the database buffer pool. These types of operations can, therefore, reduce the overall efficiency of the database by forcing random access database operations to read from disk.

Private read-only buffers do not participate in the LRU scheme. Applications that read many records quickly can benefit from private read-only buffers. The new QAD .NET UI browses could (depending on overall application usage) benefit noticeably from the use of private read-only buffers.

These read-only buffers do receive an allocation from the overall (`-B` startup parameter) public buffer pool, up to 25% of the total pool. Private read-only buffers are controlled by the `–Bpmax` startup parameter.

Any browse available in the QAD .NET UI can be made to use private buffers by inserting the program name into the `browse-buffer-config.dat` file:

`[Desktop install Directory]/com/qad/shell/interface/browse-buffer-config.dat`

```
/* This file is used for configuring the number of private buffers to use    */
/* for browses running in the AppShell. This is used for performance tuning. */
/* This number will be assigned to _MyConnection._MyConn-NumSeqBuffers.      */
/* In general, a good number to use is 6 ; consult Progress documentation    */
/* for more detailed calculations to use for assigning this value.           */
/* The format of lines in this file is: <browse program name> <num buffers>  */
/* For example: pp013 6                                                      */
pp013 6
so009 6
```

Menu Exclude

The following file enables programs to be excluded from the QAD .NET UI menu structure.

`[Desktop install Directory]/com/qad/shell/interface/menu-exclude.dat`

```
/* These programs will be excluded from the AppShell menu structure */
mgdomchg.p
mgufmt.p
mgurlmt.p
```

Hyperthreading Enabled CPUS

Intel Xenon (and similar) CPUs have a hyperthreaded execution stack, which enables them to present to the operating system as two CPUs instead of one. This is a very useful feature that can enable systems with low actual CPU counts to seem more responsive. However, the internal process of managing this feature actually reduces the maximum CPU throughput.

Depending on the usage profile of your system, it may be beneficial to disable hyperthreading. As always, extensive system testing should be conducted before committing to such a step.

CHAPTER 4

Planning and Implementation

This section provides general guidelines that should be considered when selecting hardware and deployment options for the QAD .NET User Interface.

Excessive User Demand and Missed Performance Expectations

System Sizing/Capacity Requirements Planning

Single/Two/Multi-Tier Installation

Network Considerations

Client Sizing

Excessive User Demand and Missed Performance Expectations

When users access the MFG/PRO and QAD .NET UI software, they are placing additional demand on the system, usually requiring both good response time and high throughput. When the user demand is high, it is possible that the system resources will come under load and fall out of balance. Performance in this case can suffer, with the user observing decreased system throughput and increased response time.

Unless performance is being actively managed and benchmarked, the user expectations of performance will be hard to quantify. “The system is running slow” or “my report is taking forever to come out” are not good for tracking and reporting system performance. It is far better to track quantitative information such as “My report is running 65% slower than the established upper boundary.” If the user and technical community do not understand what “normal” system behavior is, it will hard to work out what is “abnormal” behavior.

In order to track this quantitative performance information, it is required to have an established history of performance of computing (business) functions for comparative purposes. This means you need to establish benchmarking and monitoring of key MFG/PRO processes, and actively record the information over time.

A simple way to determine application performance is to benchmark key applications. A subset of QAD .NET UI programs should be chosen as key performance indicators (KPIs). The KPI applications should be run on a regular basis, at set times, with set selection criteria, with a view to timing how long they take to execute.

If data is gathered and tracked for these KPI programs over time, then the following information can be gathered:

- What are normal execution times?
- What are abnormal execution times?
- When do the abnormal execution times occur?
- Is the system, as a whole, getting slower over time?
- At what rate is the system getting slower?
- What is the upper limit of KPI program execution time before the system is determined to be suffering performance bottlenecks?

On the purely technical level, system performance needs to be tracked over time, using both spot checking and historical data automatically gathered within set sampling periods. Using this information, system performance trends in both the medium term (What do we suspect the system will behave like two months from now?) and the short term (Why is the system currently performing badly? Or why does the system perform badly from 3 pm to 5 pm daily?) can be established.

System Sizing/Capacity Requirements Planning

Introduction

At some stage during the life of the MFG/PRO and QAD .NET UI system, new hardware may need to be purchased, or existing hardware may need to be upgraded.

Whether this upgrade is merely to suit the current system setup with current growth patterns, or to provide for the implementation of new systems, a process called capacity planning can be used. Capacity planning is a useful way of determining these new hardware requirements without having to resort to complete guesswork and wishful thinking.

Capacity planning, at its most fundamental, is a method of determining the best server and equipment to effectively meet current and future workload/performance requirements—at minimum cost.

Capacity Requirements Planning

The capacity of the MFG/PRO system is determined by the workload placed upon it. Once the relationship between capacity and current workload has been determined, the same relationship can be used to plan what capacity is required for a higher workload.

The method to plan the capacity of the MFG/PRO Server is described in the following sections.

Measure Current Subsystem Utilization (Memory, CPU, Disk)

Decide if the current capacity is enough.

Is there currently a problem in the subsystem being studied?

Characterize the Workload

An understanding of the workload (the MFG/PRO application) that must be supported is required.

Build the Capacity Planning Model

This establishes the relationship between the workload and the capacity required. Many different models can be used, all of various levels of mathematical complexity. Simple linear regression techniques such as graphs are often satisfactory, however.

The model can be used to predict the required capacity (hardware) for future workloads, or what is required to handle the existing workload.

CPU Requirements

Since every MFG/PRO and QAD .NET UI installation is unique, it is not possible to provide absolute guidelines for the amount of CPU processing power required without conducting a formal capacity requirements plan.

However, internal testing using sample user workloads and user counts has enabled QAD to provide the following broad guidelines.

Very Light User Loads (1 to 10 users)

For very light user counts, a single Xeon CPU (or equivalent) should provide adequate performance.

Light to Medium User Loads (10 – 75 users)

As the user count increases, so do the capacity requirements of the CPUs. For more than 10 users, a two-way system is really needed to maintain good response time. A dual Xeon-based Linux server should be able to handle up to about 75 users with good performance. This is heavily dependent on the usage profile of the application. Some 75-user systems will run very well on a dual-Xeon server, where others may struggle.

Medium to Heavy User Loads (75+ users)

For heavier user counts (75+ users), a formal capacity requirements planning approach should be taken. QAD's Global Technical Services can help with this process.

Physical Disk Requirements

The QAD .NET UI does not in itself require disk resources. Disk activity is low and disk space required is minimal.

QAD .NET UI Interface (MFG/PRO extensions):	~ 150 MB per interface
Tomcat:	~ 10 MB
Tomcat QAD .NET UI Web application:	~ 150 MB per interface

Memory Requirements

An operating system process consists of two memory parts: a private data segment unique to each process and a shared library data segment that is used by multiple processes. In the following guidelines, a very conservative approach to setting memory capacity is taken, since it is better to over specify the capacity than under specify.

The `_progres` executables, for instance, can vary from as low as 15 MB to as high as 100 MB depending on the nature of the work being done. The actual size of the process in memory (the memory size of the process minus the memory size of the shared libraries) can vary upward from 5 MB. A memory footprint of 25 MB is chosen for the purposes of this memory sizing exercise. On specific customer systems, these assumptions should be carefully measured and evaluated to see if they are still applicable.

Below are the calculated memory requirements for components of the QAD .NET UI. These numbers can be used to derive the memory requirements for a specific installation.

_progres executable	~ 25 MB per process
AppServer	~ 45 MB per AppServer
Databases	As per normal Progress database tuning
Java	50 MB per AppServer
Tomcat	From 256 MB (default) up to 1024 MB and higher
Telnet daemons	0.5 MB each

Login daemons	0.5 MB each
Bash shells	1 MB each
Operating system	Varies, usually about 100 MB

At a simplistic level, basic multiplication can be used to determine expected maximum memory requirements (example only):

	20 users	100 users
_progres executable	20 x 25 MB = 500 MB	100 x 25MB = 2500MB
AppServer	2 x 45 MB = 90 MB	5 x 45MB = 225 MB
Databases	160 MB	160 MB
Java	2 x 50 MB = 100 MB	5 X 50MB = 250 MB
Tomcat	256 MB	1024 MB
Telnet daemons	0.5 x 20 MB = 10 MB	100 x 0.5 = 50 MB
Login daemons	0.5 x 20 MB = 10 MB	100 x 0.5 = 50 MB
Bash shells	1 x 20 MB = 20 MB	100 x 1 = 100 MB
Operating system	100 MB	150 MB
Total:	approximately 1.25 GB	Approximately 4.5 GB

Single/Two/Multi-Tier Installation

Depending on the number of users and geographic considerations, deployment may require more than one server. One server hosts the MFG/PRO database and another server runs the Tomcat components (Web server and Tomcat). This is known as a two-tier configuration, and is useful for taking load off the database server.

Database Server

Database, Telnet processes and MFG/PRO code

Tomcat Server

Tomcat / HTTP Server

Another variant of a two-tier installation is to shift the telnet processes and MFG/PRO code away from the database server; in this scenario, the database server only processes database requests. Business logic is processed on the Tomcat server.

Multi-tier solutions require very low network latency between the server components. It is worth connecting all server-side machines with dedicated Gigabit/Fiber Ethernet private networks on separate network cards.

Network Considerations

Internal testing shows the QAD .NET UI has higher bandwidth requirements under initial login and menu presentation (approximately 0.2 Mbits/s) compared to the Desktop interface (approximately 0.037 Mbits/s). This is an increase over standard Desktop 2.9. However, the HTTP traffic generated from within the maintenance screens using the embedded QAD .NET UI

screens is approximately 33% lower than the Desktop 2.9 screens (0.008 Mbits/s versus 0.012 Mbits/s).

Under peak conditions, and assuming no other network traffic, the following user counts could reasonably be supported over a local area network (LAN) connection:

50 clients on 10 base-T LAN

500 clients on 100 base-T LAN

Effect of Latency (Wide Area Networks)

Reduced throughput and increased network latency slow down the response times of the QAD .NET UI application. As a general rule, a latency of 200ms slows down response times (compared to a latency of <1ms) by a factor of three.

Client Sizing

Introduction and General Summary

From internal testing done on two client machines (minimum and preferred specification), the responsiveness of the QAD .NET UI interface is much higher on faster machines. The 2.5 GHz client typically responded in about half the time as the 1.5 GHz client. It is expected that clients under 1.5 GHz would continue the trend in slower responsiveness to the point where the application would be unusable in any real sense.

Preferred Client

2.5 GHz Pentium IV
1 GB RAM
1 x 40 GB IDE hard drive (7200 RPM)

Although CPU use can be spiky, the utilization of the CPU is low most of the time, with brief spikes to 70% as the data from the server is presented on the screen.

Minimum Specification Client

1.5 GHz Pentium IV
512 MB RAM
1 x 8 GB IDE hard drive (7200 RPM)

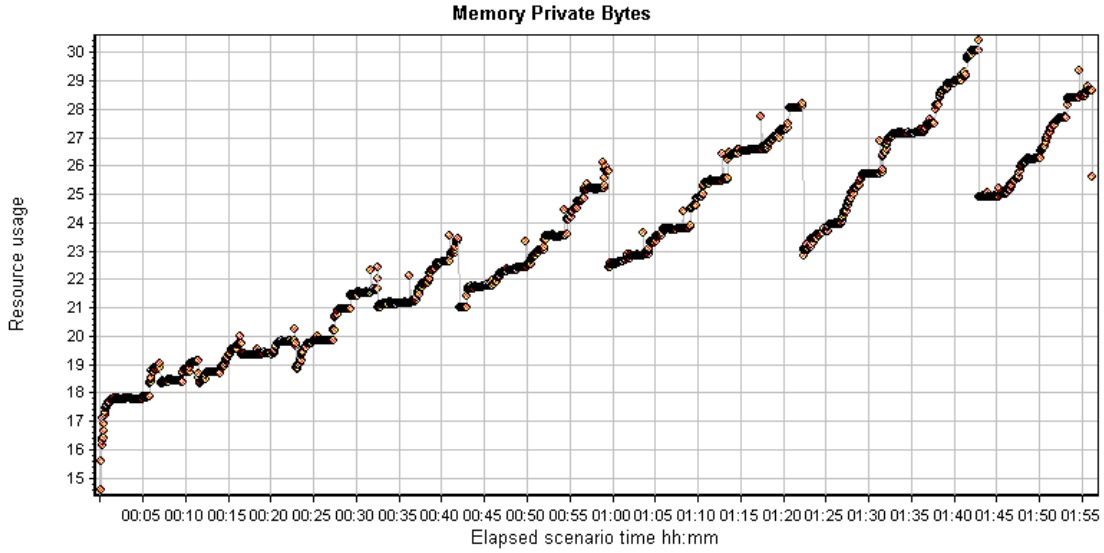
Similar to the preferred client, CPU usage is spiky. The utilization of the CPU is low most of the time, with brief spikes to 100% as the data from the server is presented on the screen. Available memory can fall as low as 80 MB under heavy and prolonged usage.

Disk Usage

Disk usage is minimal (less than 50 MB), and disk activity is also minimal. This interface is not expected to cause disk drive congestion.

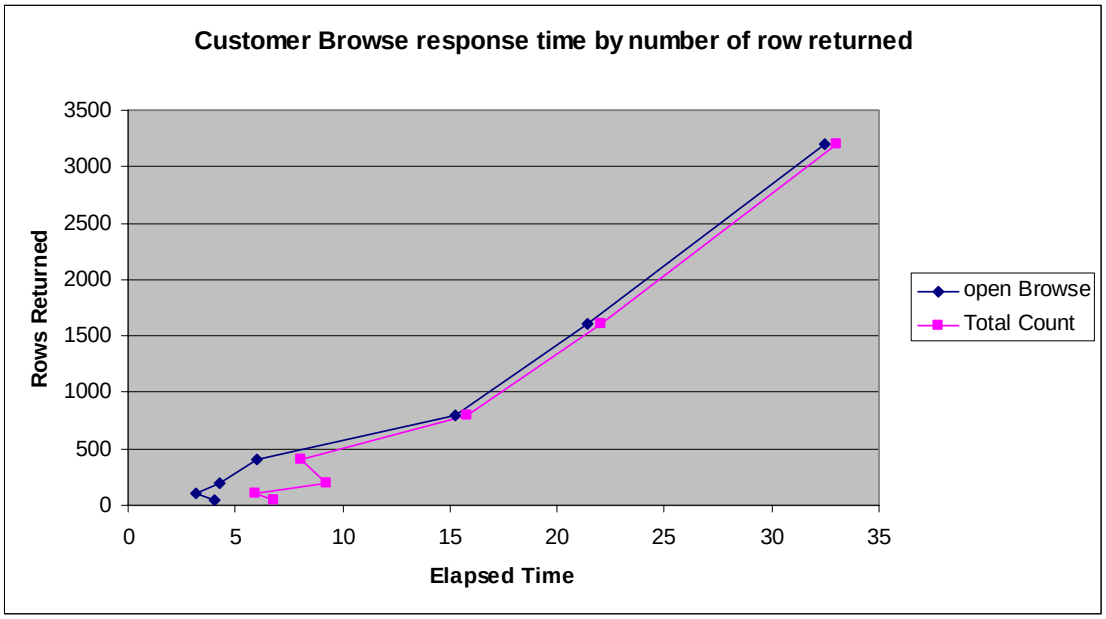
Memory Usage

- Initial launch of the QAD .NET UI client consumes about 25-30 MB of system memory.
- Memory usage can increase during use of the QAD .NET UI. When dealing with large data sets (in hundreds of thousands of records range), memory use increases rapidly. The QAD .NET UI clients have been seen to consume upwards of 100MB of memory during tests
- As can be seen on the following test, on a 512MB client, usage increases from about 18% (~92 MB) to 30% (~ 153 MB) when running a selection of QAD .NET UI browses and embedded Desktop screens over a 2-hour period.



Response Times of Browse by Number of Rows Returned

The response times for a QAD .NET UI browse show good scalability as the number of rows returned increases. Ignoring low record counts, there is a strong linear correlation between response time and rows returned, as the following graph shows.


Customer Browse response time by number of row returned
Rows Returned
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
35
Elapsed Time
open Browse
Total Count

CHAPTER 5

QAD Internal Benchmark Results

QAD conducted intensive internal performance testing and benchmarking of the QAD .NET UI. Selected summary information is presented in this section.

Embedded (QAD .NET UI) Desktop Versus Desktop

Application Scalability

Embedded (QAD .NET UI) Desktop Versus Desktop

Results are shown for the difference in response times moving between the frames in the following screens. Each test was run multiple times to generate the average and 90% response times.

- Customer Credit Inquiry (27.8)
- Issues – Unplanned (3.7)
- Receipts – Unplanned (3.9)

Embedded Desktop (QAD .NET User Interface)

Average 43.9 packets per second

Average packet size: 548.0 bytes

Average throughput: 0.192 Mbits/s (HTTP only: 0.008 MB/s)

Desktop 2.9

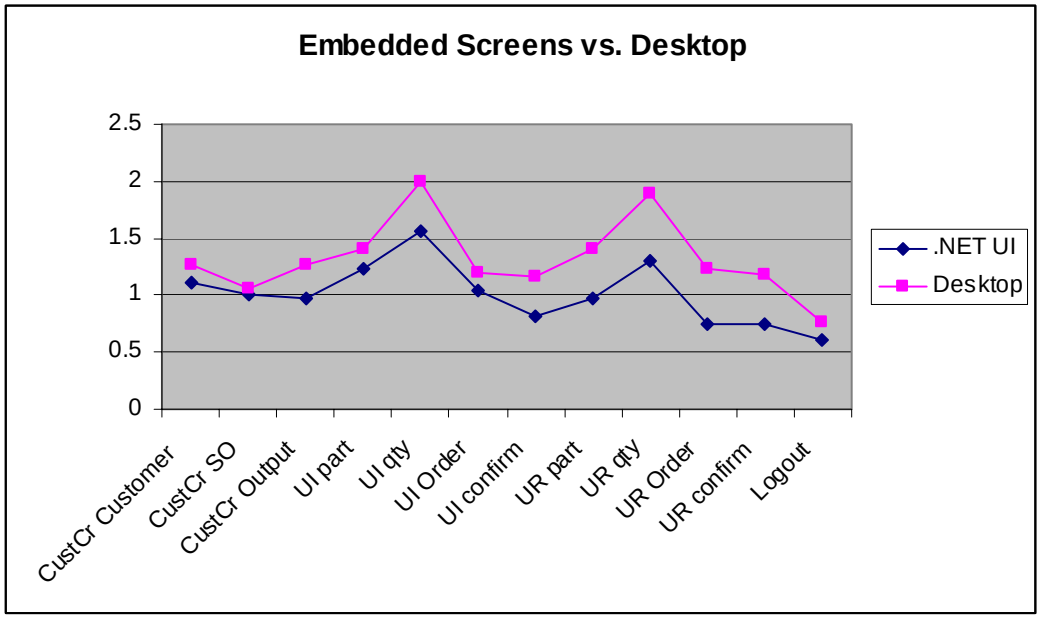
Average 10.85 packets per second

Average packet size: 426.0 bytes

Average bandwidth: 0.037 Mbits/s (HTTP only: 0.012 Mbits/s)

Response Time Difference

As can be seen from the graphical results of the QAD .NET UI Embedded Desktop versus Desktop 2.9 response time results, the embedded screens are on the order of 15 to 20% quicker. In the graph below, the items on the x-axis represent individual frames within the MFG/PRO maintenance screens.



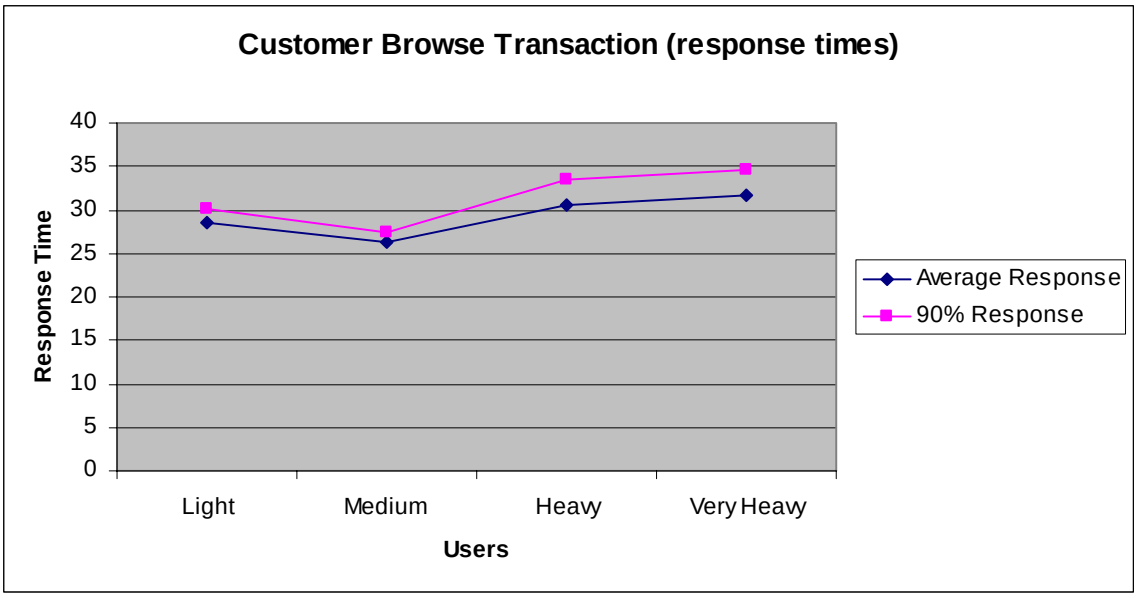
Application Scalability

This test is a black box test driven from a Progress 4GL script. A QAD .NET UI browse (adbr001.p – Customer Browse) was launched against an MFG/PRO database containing 44,351 customers. The following actions were then carried out:

- Display 1000 records
- Get the total record count (Max record count set to 50,000)
- Get next record
- Get previous record
- Get last record

The total response time was recorded. The purpose of this test was to track the scalability of the customer browse under load.

As can be seen from the graph below, response times remain fairly consistent as the user count increases, indicating good scalability.


Customer Browse Transaction (response times)
Response Time
40
35
30
25
20
15
10
5
0
Light
Medium
Heavy
Very Heavy
Users
Average Response
90% Response

CHAPTER 6

Reference

Test Hardware and Software

Test Software

Database Server

Dual 3.06 GHz Xeon CPUs (1 MB L3 Cache, 533 MHz bus speed)

4 GB RAM

Ultra 320 SCSI disk controller

4 x 15k RPM SCSI disk drives, striped (RAID 0)

Preferred Specification Test Client

2.5 GHz Pentium IV

1 GB RAM

1 x 40 GB IDE hard drive (7200 RPM)

Minimum Specification Test Client (plnt05)

1.5 GHz Pentium IV

512 MB RAM

1 x 8 GB IDE hard drive (7200 RPM)

Load Test Server

2 x Pentium II (450 MHz)

1 GB RAM

1 x 10 GB IDE hard drive

1 x 16 GB IDE hard drive

Application Test Software

Item	Vendor	Version
MFG/PRO	QAD Inc.	eB2.1 SP4
Progress Database Engine	Progress Software Corp.	10.0B03
Progress WebSpeed transaction server	Progress Software Corp.	10.0B03
Progress AppServer	Progress Software Corp.	10.0B03
Java SDK	Sun Microsystems, Inc.	1.5.0_04
Red Hat Enterprise Linux	Red Hat, Inc.	AS 3.0 RL 4
QAD .NET UI	QAD Inc.	1.0.152
QAD Desktop	QAD Inc.	2.9.0.108
Tomcat	Apache Software Foundation	5.5.9
Internet Explorer (client)	Microsoft Corporation	6.0.0028
LoadRunner load testing software	Mercury Interactive Corporation	8.1.1.1
Quicktest PRO functional testing software	Mercury Interactive Corporation	8.2.1.1

Load Test Software

The QAD test execution environment was made up of a number of parts controlled by LoadRunner, a commercial load testing tool from Mercury Interactive Corp.

The new browse functionality generates traffic on the network and on the server. This traffic is a proprietary Progress 4GL protocol, and the Mercury tools are not capable of recording or playing back this traffic. An in-house test harness was developed to simulate the traffic for a number of clients, doing simple browse transactions of opening and navigating the browse. A 4GL client was used and driven by a telnet session controlled by LoadRunner. This allowed the full test to be controlled by LoadRunner.

The final traffic generation was a full UI client. Quicktest PRO is Mercury's offering for functional automated testing of .NET applications. Quicktest was used to drive the full UI client.

Revision History

Rev	Effective	Revision Summary
1	May, 2006	Initial Release