QAD Desktop 2

Deployment Considerations

Database Engineering Team
May 24, 2004
Revision 4

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

Contents

Introduction

The purpose of this document is to assist customers in deploying Desktop in the most optimal fashion for their company. It discusses possible performance bottlenecks and how to overcome them. It also provides advice on common Desktop deployment issues. This document should be considered as a supplement–not a replacement–to the Desktop installation guide.

This document is only a starting point for understanding deployment and performance issues. For complex deployments, it is recommended that QAD Services be contacted to assess how best to improve system performance.

What is Good Performance?

MFG/PRO users want to complete a task, such as entering a sales order, as quickly as possible. During data entry, most users find performance is acceptable if field submits take no longer than a second or two. A submit is any action within a Desktop program screen that requires pressing the Enter key or choosing the next arrow.

No matter how many features a client has or how well the system performs otherwise, if submit times are slow, performance is unacceptable. However, reports and programs that require displaying large amounts of data can take several seconds to display; this is appropriate and acceptable for users.

Desktop can meet user performance demands when properly configured in a stable network and powered by adequate server and client hardware and software resources.

Another aspect of system performance that must to be considered is how well the system allows the addition of users; that is, how well the system scales. Desktop architecture allows its components to be distributed in order to allow the system to scale to support a large number of users in growing environments.

Client, Server and Network Interactions

The performance of any Web application, such as Desktop, can vary widely over different systems and with varying usage. Desktop is often part of a complex environment of widely distributed client machines that have different processing capacities, one or multiple servers that perform other duties, and a network that can have a highly variable pattern of usage. To receive good performance from Desktop, the system administrator must understand how Desktop operates within this environment.

The Desktop screens are rendered on the client after it receives data from the server. Communication between the client and the server is done via network lines, such as a LAN or a WAN.

Good performance is based on fast client response times. Three major resources that affect response times are the client, server and network. Each of these must be considered, measured and tuned to guarantee performance success. To optimize performance, understand these resources and how your system uses each one. Create a consistent way to evaluate performance results both before and after tuning.

Client

Desktop screens are rendered by the browser (Internet Explorer). The rendering time is a function of how complicated the screen is, how much data is being displayed, and the configuration and availability of system resources to the client.

Achieving optimal performance from the client is heavily dependant on the client hardware and software configuration. The hardware configuration–in particular the speed of the CPU–has the most significant client-side performance bottlenecks.

The minimum client software requirements are:

- Microsoft Windows 98, ME , NT, 2000, and XP
- Internet Explorer 5.5 SP2 or higher
- Java plug-in 1.3.1_04, 1.3.1_05, or 1.3.1_08

The minimum client hardware requirements are:

- Memory (RAM):
  - Windows 98 and ME: 128 MB
  - Windows NT, 2000, and XP: 256 MB
- 800 MHz or faster processor

These recommendations do not take into account any additional requirements resulting from overhead of other concurrent client-side applications such as e-mail or desktop publishing applications.

Client Performance

To measure response time in Desktop, start with a maintenance program such as Purchase Order Maintenance and measure the time it takes to move from one field to the next. Any time the program requires you to press Enter or click an action arrow to move, you are submitting data to the server and receiving back a new screen and data for display.

The first time a user brings up a screen, submits take substantially longer than subsequent submits. This is because the screens are not initally cached. Therefore, when measuring submit times, ensure accuracy by running through a screen at least once before evaluating performance.

Timing Tool

To collect precise submit times, enable the Desktop timing tool by adding the following line to the `config.js` file found in the `js` directory of your Desktop Web server installation:

```
var htmlTiming = "true";
```

There is no need to restart the Web server for this to take effect. You may, however, need to exit the client session and clear the Internet Explorer cache. (See below for details.) Individual submit times display at the top of the Desktop client screen. Average submit times also display. To reset the timer, select another menu item from the left side menu bar and then reselect the program you were testing.

Note The timing tool works only with HTML maintenance screens.

Screen Types

Desktop displays programs in HMTL, telnet, or as Java browse programs.

- HTML maintenance programs are those generated by the application server (Tomcat). Most MFG/PRO programs display as HTML programs in Desktop.
- Java browses are generated by a WebSpeed agent. WebSpeed also handles lookups and Desktop login, which also use Java.
- Embedded telnet screens look and function similarly to MFG/PRO character client screens. For telnet programs, the Web browser runs a telnet client that resides in a Java class file stored in `multinet.jar`. This telnet client is used for screens that do not function properly as HTML maintenance screens.

  `Multinet.jar` is downloaded the first time a user logs in. In addition to the embedded telnet client, it also contains Java files that pertain to WebSpeed functions.

Client-Side Caching

The files that comprise the HTML maintenance screens, such as Purchase Order Maintenance, are stored on the client in the Internet Explorer cache the first time they are requested. The next time a user needs to access a particular screen, Desktop is able to retrieve those files locally and avoid having to go back to the server to re-create the HTML page. This improves performance and reduces bandwidth consumption. By default, HTML files are stored for local use for 30 days.

The caching option in Internet Explorer should be set to Automatic. This allows the client to keep images and other reusable files locally after retrieving them the first time. Set automatic caching in Internet Explorer by selecting Tools|Internet Options|General|Settings|Automatically.

When making server-side changes to Desktop or any other changes that might impact the client, such as the introduction of custom code, clear the client cache. To clear the cache in Internet Explorer, select Tools|Internet Options|General|Delete Files|OK. The Desktop administrator will likely perform this function frequently during the initial testing and deployment phases.

> ***Note*** To force an update of locally cached screens for all clients, see version-specific URLs in the Server section below.

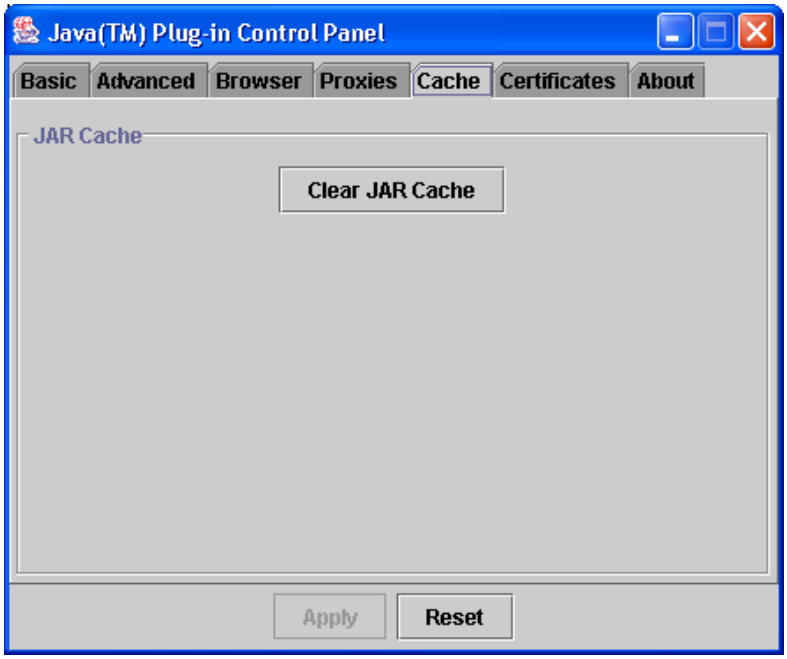
Multinet.jar File

As mentioned above, `multinet.jar` is downloaded and stored on the client the first time a user logs into Desktop. A `multinet.jar` file is stored under the `\Documents and Settings\`*`UserName`*`\java_plugin_AppletStore` heirarchy for each Desktop environment a user logs into. If someone uses three different Desktop environments (for example, test, production, and consolidated) three separate `multinet.jar` files are downloaded on the computer.

Futhermore, if another user logs on to the same computer and accesses these same three environments, a total of six `multinet.jar` files will reside locally.

`Multinet.jar` files typically change from one Desktop release to the other. Having an older version cached locally may cause problems. If you suspect this is the case, remove existing `multinet.jar` files to force a download of the latest version upon the next login.

`Multinet.jar` files are stored in the Java Jar cache. To clear the Jar cache, from the Control Panel select the Java Plug-in 1.3.1_0*x* icon. Next, click on the Cache tab and click Clear JAR cache. This clears all local copies of `multinet.jar` for this user.



Server

Depending on your deployment model, Desktop and MFG/PRO components can be spread across one or more servers (see Deployment Configurations section below for detailed information). Each of these servers needs to be monitored and tuned to ensure optimal performance. To obtain consistently good response times, servers should be dedicated for Desktop and MFG/PRO use whenever possible.

In addition to the base MFG/PRO system, there are three major Desktop server components:

1) Application server (Tomcat)
2) Web server (Apache, IIS, or iPlanet)
3) Progress WebSpeed broker and agents

In real-world Desktop deployments, the Web server and WebSpeed default settings usually provide satisfactory performance. However, the application server and its settings can be a significant bottleneck.

The application server can be a bottleneck for one of three typical reasons: insufficient CPU, insufficient RAM, or suboptimal Java settings.

Monitoring Server Resources and Performance

The application server can put a large demand on server memory and CPU resources. Therefore, it is crucial to monitor their usage.

On UNIX operating systems, tools to gather memory and CPU statistics include top, glance (HP/UX), monitor (AIX), and perfmeter (Solaris). On Windows servers, use Performance Monitor (under Start|Programs|Administrative Tools|Performance) or the Performance Tab in Task Manager.

Optimizing Server Performance

The following sections discuss tools, techniques and strategies used to monitor and optimize server resources and perfromance.

CATALINA_OPTS Parameter

The Java settings for the application server are specified by the CATALINA_OPTS environment variable. These setting directly affect Java and server interactions and thus affect server performance. For Tomcat, these settings are set in the `setenv.sh` file (`setenv.bat` on Windows). Pay special attention to the following settings:

- Set the system to be in server mode by using the `-server` parameter, if possible. This mode can improve Java performance dramatically.

  Note AIX's JDK does not support the `-server` switch.

- Set `-Djava.awt.headless=true`. This switch is necessary to ensure proper Desktop functionality.
- Set the minimum and maximum Java heap size by using the `-Xms` and `-Xmx` parameters. These determine the amount of memory made available to Tomcat.

  Example Setting `-Xms256m -Xmx768m` would allocate 256 MB of RAM to Tomcat initially and increase the heap size to a maximum of 768 MB as needed.

The complete line in `setenv.sh` would look similar to this:

```
CATALINA_OPTS="-server -Djava.awt.headless=true -Xms256m -Xmx768m"
```

Additionally, the following options improve Java performance for the application server. These switches and values vary by operating system and Java version. They may not be applicable to your Desktop environment. Test the impact of any of these switches before deploying in a production system:

```
-XX +UseParNewGC

-XX NewRatio=2

-XX MaxTenuringThreshold=0

-Xss768k
```

Version-Specific URLs

QAD recommends deploying Desktop using version-specific environment names. For example, name an environment *Production25* rather than *Production*.

Following this guideline results in a unique URL every time Desktop is upgraded. This ensures that end-users receive a complete set of new files rather than existing files from the servers cached locally. In effect, using version-specific Desktop environment names forces a global cache

clearing on the client side, thus avoiding having to manually clear the Internet Explorer cache for each user.

Minimum UNIX File Permissions

It is a common practice for system administrators to lock down Desktop files on the server-side as much as possible without hindering functionality. Following is a list of the main directories that are written to during a Desktop installation as well as recommended minimum permissions. Test these settings before deploying in a production environment.

- HTML Web server documents directory should be owned by *root* with permissions set to 500.
- CGI-BIN Web server scripts directory should be owned by *root* with permissions set to 555.
- Tomcat directory should be owned by *root* with permissions set to 500 for everything except for the logs directory which is set to 700.
- MFG/PRO directory should be owned by *mfg* with permissions set to 500.
- Change ownership of `connmgr.`*`DBSetName`* and `telnet.`*`DBSetName`* to the *mfg* user.

Server-Side Caching

The first time an HTML maintenance screen is requested by a user, it is created on the application server. The HTML files used to create the screen are cached in a subdirectory located under `webapps/`*`webappname`*`/cache`. These files are stored on the server to be used for any subsequent requests for that program, whether from the same user or a different user.

However, as explained in the Client section, once a user has called a program, it is also stored locally on the client machine. The client does not continually need to request the server's cached version.

In the course of testing and configuring Desktop, it may be necessary to clear the server cache if changes introduced by new code are not being reflected in the client.

Use these steps to clear the server-side cache:

1) Shut down the Connection Manager for the Desktop environment in question.
2) Go to the `webapps/`*`WebAppName`*`/cache` directory. *WebAppName* refers to your system name.
3) Remove all two-letter language directories beneath the cache directory, such as `us` or `fr`.
4) Start the Connection Manager for the Desktop environment.

Server Operating Systems

QAD supports running Desktop on the following server operating systems: AIX, HP-UX, Linux, Solaris, Tru64, and Windows.

Important The application server platform must support Java 1.4.x.

For up to 30 concurrent Desktop users, UNIX-based and Windows-based servers perform equally. However for more then 30 users, performance on Windows-based servers is drastically

degraded. This is due to a system-level bottleneck between Windows and the `_progres.exe` executable.

Therefore, at this time, Windows is recommended only for smaller deployments of roughly 30 concurrent users or less per server. QAD is working on alleviating this limitation to Windows performance. This guide will be updated when improvements are made in this area.

Installing Tomcat as a Windows Service

Installing Tomcat as a Windows service allows you to log off the server console without shutting down Tomcat. To install Tomcat as a Windows service, execute the following three lines from a command prompt. It may be easier to put them into a batch file and then execute the batch file instead.

The following options override the values specified in the `setenv.bat` because `setenv.bat` is not read when Tomcat starts as a service. In the following example, substitute the sample values with the corresponding correct values for your environment.

```
set JAVA_HOME=C:\j2sdk1.4.2_04

set CATALINA_HOME=D:\tomcat

%CATALINA_HOME%\bin\tomcat -install "Apache Tomcat"
%JAVA_HOME%\jre\bin\server\jvm.dll -Djava.awt.headless=true -server -Xms256m
-Xmx768m -Dfile.encoding=utf-8 -
Djava.class.path=%CATALINA_HOME%\bin\bootstrap.jar;%JAVA_HOME%\lib\tools.jar
-Dcatalina.home=%CATALINA_HOME% %CATALINA_OPTS% -Xrs -start
org.apache.catalina.startup.Bootstrap -params start -stop
org.apache.catalina.startup.Bootstrap -params stop -out
%CATALINA_HOME%\logs\stdout.log -err %CATALINA_HOME%\logs\stderr.log
```

Network

Insufficient bandwidth and excessive latency can adversely affect Desktop performance. If the data pipe is too small or if the data takes too long to go between the client and the server, Desktop response time will be unacceptable no matter how fast the server and client machines are. Therefore, it is of utmost importance to verify that each Desktop client is allocated adequate bandwidth, and ensure that the latency, also known as lag time, is not too great to hinder good performance.

The following sections discuss some useful network tools, and detail Desktop bandwidth and latency requirements. While these numbers are useful as guidelines, be sure to run tests to verify that performance is acceptable on your network alongside all other normal traffic.

Network Bandwidth Tools

There are many tools that can be used to measure bandwidth for a Desktop client; the following are just a sample.

Windows Performance Monitor. This tool measures bytes per second over any given network interface; results can be logged for further analysis.

Ethereal packet sniffer. The free utility (www.ethereal.com) not only measures how much bandwidth is utilized, but it also captures and replays the entire conversation between client and server for detailed examination.

Microsoft Network Monitor. This is a packet sniffer similar to Ethereal and offers some unique capabilities of its own.

Ping. This tool measures how long it takes a piece of data of a specific size to complete a round trip from the client to the server and back.

Tracert. This tool is similar to Ping but goes a step further. It pings each router along the way until it reaches the final destination. This not only provides response times, but also the number of hops the data goes through. By doing this, Tracert also helps pinpoint where performance may be degrading your network.

These tools are useful to determine how fast the response time is between the Desktop client and the server. The slower the response time, the higher the latency is said to be.

Besides looking for slow response times, you should also look for dropped packets. A high percentage of dropped packets (for example 10%) can ruin performance on an otherwise adequately sized network pipeline.

Bandwidth Requirements and Planning

The network requirements for bandwidth can be estimated for a given period by multiplying the average bandwidth usage of a typical user by the number of users on the system.

However, estimating the average bandwidth usage per Desktop user can be problematic because the Desktop client is, for the most part, stateless. For much of the running time of Desktop, there is no active transaction on behalf of the client and, therefore, no network bandwidth consumption.

Also, bandwidth requirements can vary significantly from function to function. QAD tests show that the average bandwidth consumption is 16 kbps (kilobits per second) after the first time a screen is accessed. This conclusion was based on measuring a variety of browses, lookups, and HTML maintenance screens.

For reports, bandwidth consumption varies based on the size of the report. The larger the report, the more bandwidth that is consumed. Finally, it should be noted that the login process takes approximately twice as much bandwidth (32 kpbs) as actually using the Desktop client.

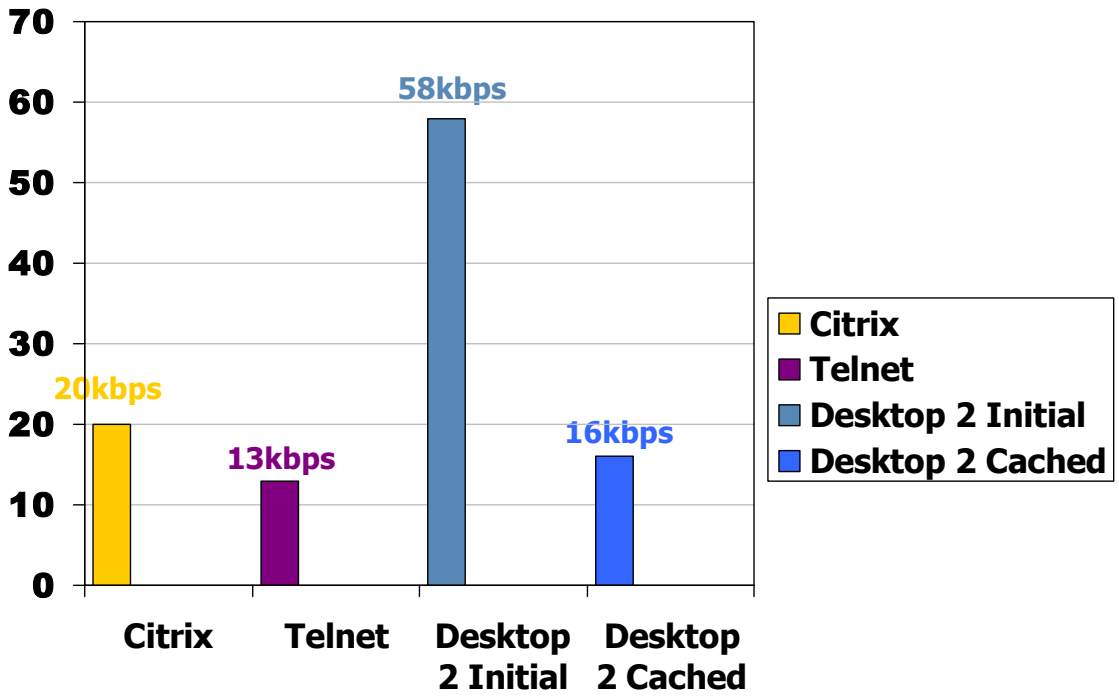
Figure 1 shows the average bandwidth consumption for several different MFG/PRO clients. MFG/PRO character telnet clients use an average of 13 kbps per second.

Citrix clients are the WAN solution for the MFG/PRO GUI client and use an average of 20 kbps.

In certain scenarios, such as low-bandwidth transcontinential deployments, using Citrix or Terminal Services to run Desktop can result in better response times and more consistent performance. If you are planning on placing Desktop clients thousands of miles away from their servers, this is a configuration worthy of consideration.

An average bandwidth of 58 kbps is consumed the initial time a screen is accessed in Desktop. This number drops to an average of 16 kbps after caching.

Figure 1. Average Bandwidth Consumption



The network planner should allow for peak consumption by all users on a particular WAN link.

Client Architecture Differences

When comparing bandwidth requirements between Desktop and other MFG/PRO clients, it is important to keep in mind how bandwidth is consumed differently between these clients.

For telnet and Citrix users, updates to the screen are sent to the client as soon as they are received from the server. A steady stream of data goes back and forth during a typical user session. But with Desktop, a different phenomenon occurs.

As stated above, during the time data is being entered between submits, there is no network traffic activity between the Desktop client and server. Only on submit is the information for a screen frame sent to the server and data for the next frame received. This means that Desktop tends to have a spike-like pattern of bandwidth usage, periodically going from no bandwidth to maximum bandwidth.

Due to this variance in bandwidth patterns, other MFG/PRO client types may, practically speaking, perform better than Desktop in limited bandwidth situations with everything else being equal. Testing of real-world scenarios is necessary to ensure that Desktop performs adequately when remotely deployed.

Initial Download

The first time a client uses Desktop, three files are automatically downloaded:

- The Java runtime environment installer (8.2 MB)
- The Adobe SVG Viewer installer (2.3 MB)
- The `multinet.jar` file (900 k).

Together these three files are approximately 11.5 MB.

When a client connects from the local LAN, the impact of downloading these files is negligible. There is just the overhead of additional time required to install the Java runtime environment and SVG Viewer program to take into consideration.

However, when a Desktop client connects to the server for the first time from a remote, low-bandwidth connection, downloading these files can take 15 minutes or more. Furthermore, during this time, there is no visible response on the client that indicates why the process is slow. Take this into consideration when deploying Desktop to remote clients for the first time.

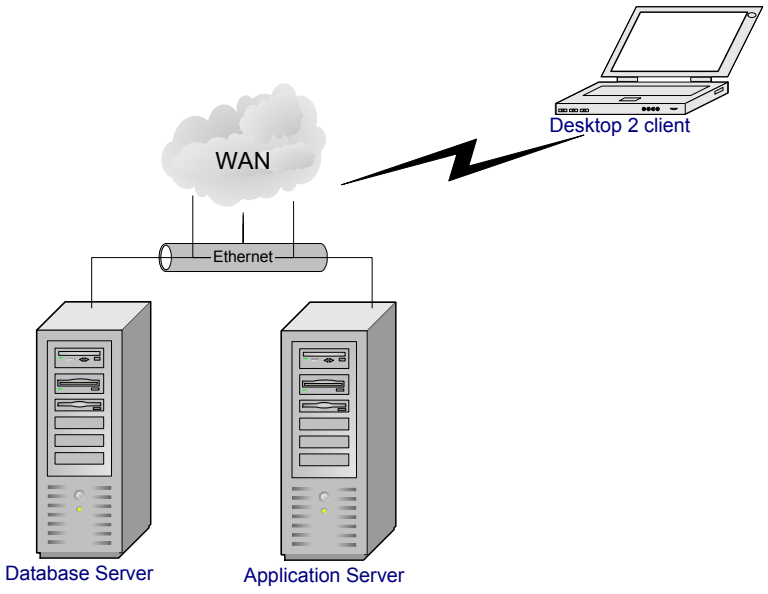
Network Latency Tests and Results

Latency tests were performed under two different configurations. The first configuration consists of a two-tier deployment with the application server local to the database server. The second configuration is also two-tier but with the application server local to the client. The following sections summarize each of the scenarios and include our findings. The average response times in seconds were calculated using the HTML Timing Tool. The maximum latency tested was 500 ms.

Latency Test with Application Server Local to the Database Server

In our first round of tests, the latency was applied between the client and the application server. The application server resided on the same segment as the database server and was connected to it via a switched 100 mbps Ethernet connection. The next figure illustrates this configuration.

Figure 2. Application Server Local to Database Server



We tested various degrees of latency (0 ms through 500 ms) at three bandwidths (64, 128, 256) to simulate real-world WAN conditions.

Table 1. Application Server Local to Database Server Test Results

Desktop 2 Latency Test Application Server *Local to Database Server*						
	0 Latency			100 Latency		
	64k	128k	256k	64k	128k	256k
Inv Detail Report	1.41	3.08	2.78	1.51	2.59	2.61
Multi-Item Transfer	2.14	1.62	1.83	2.24	1.18	1.75
WO Maintenance	0.94	0.87	0.88	1.15	0.77	0.82
	200 Latency			300 Latency		
	64k	128k	256k	64k	128k	256k
Inv Detail Report	3.03	2.66	2.61	3.16	2.64	2.61
Multi-Item Transfer	1.20	1.27	1.63	1.24	1.19	1.59
WO Maintenance	0.78	0.77	0.86	0.78	0.79	0.85
	400 Latency			500 Latency		
	64k	128k	256k	64k	128k	256k
Inv Detail Report	1.91	3.64	3.24	1.82	3.67	3.24
Multi-Item Transfer	2.89	2.69	2.51	3.24	2.99	2.51
WO Maintenance	1.46	1.34	1.27	1.57	1.48	1.27

Observations

The average response time for a given test varied somewhat even when bandwidth and latency remained constant. When comparing (zero) 0 ms latency to 500 ms latency, average response time increased in every case. The results above indicate that for optimal performance, latency should be 300 ms or less.

Latency Test with Application Server Local to Client

In the second set of tests, the latency was applied between the application server and the database server. The application server resided on the same segment as the Desktop client and was connected to the database server via WAN link. The next figure illustrates this configuration. We tested the same combinations of latency and bandwidth as we did in the first round of tests.

Figure 3. Application Server local to Desktop 2 client

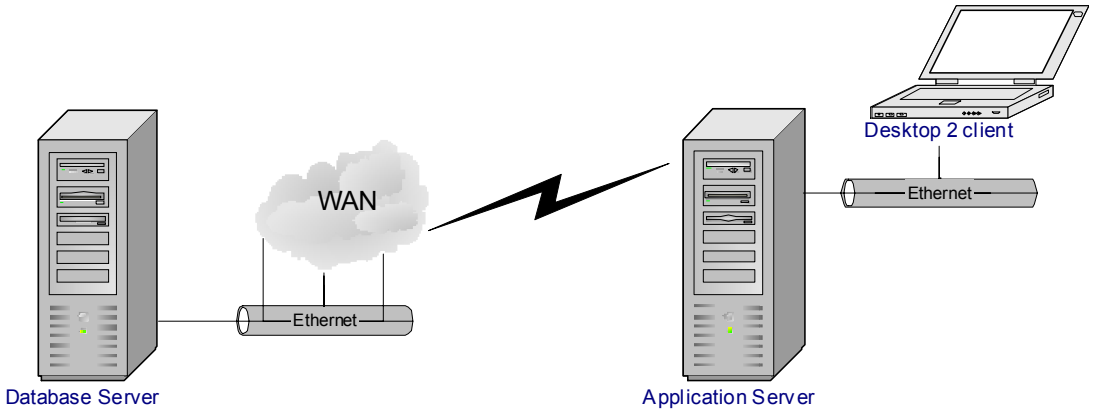


Table 2. Application Server Local to Client Test Results

Desktop 2 Latency Test Application Server *local to client*						
	0 Latency			**100 Latency**		
	64k	**128k**	**256k**	**64k**	**128k**	**256k**
Inv Detail Report			9.95			10.51
Multi-Item Transfer	1.29	1.68	1.47	2.28	2.99	2.78
WO Maintenance	1.66	1.35	1.15	2.82	2.23	2.20
	200 Latency			**300 Latency**		
	64k	**128k**	**256k**	**64k**	**128k**	**256k**
Inv Detail Report			12.79			16.95
Multi-Item Transfer		4.66	4.09	6.64	4.00	5.50
WO Maintenance	3.81	3.52	3.18	5.30	4.98	4.33
	400 Latency			**500 Latency**		
	64k	**128k**	**256k**	**64k**	**128k**	**256k**
Inv Detail Report			6.25			7.29
Multi-Item Transfer	5.75	5.57	5.16	9.72	9.13	8.61
WO Maintenance	6.25	5.72	5.51	7.75	7.49	7.04

Observations

In nearly every case, the average response time was slower compared to the first round of tests. At low bandwidths, tests involving reports (Inventory Detail Report and Multi-Item Transfer) either took an inordinate amount of time or failed to complete. This is indicated by the grayed out boxes in the table above.

In conclusion, we found the only viable configuration option for multi-tier Desktop deployments is when the application server is local to the database server. Furthermore, for optimal performance latency should not exceed 300 ms.

Deployment Configurations

Depending on the number of users and geographic considerations, deployment may require more that one server. A single-tier configuration is when one server is used to host the MFG/PRO database and Desktop components.

In many cases, one server hosts the MFG/PRO database and another server runs the Desktop components (Web server and Tomcat). This is known as a two-tier configuration. Two-tier is recommended for optimal performance. There are several variations of a two-tier configuration.

Finally, there can be multiple database and application servers assisting in failover and scaling. This is known as n-tier configuration.

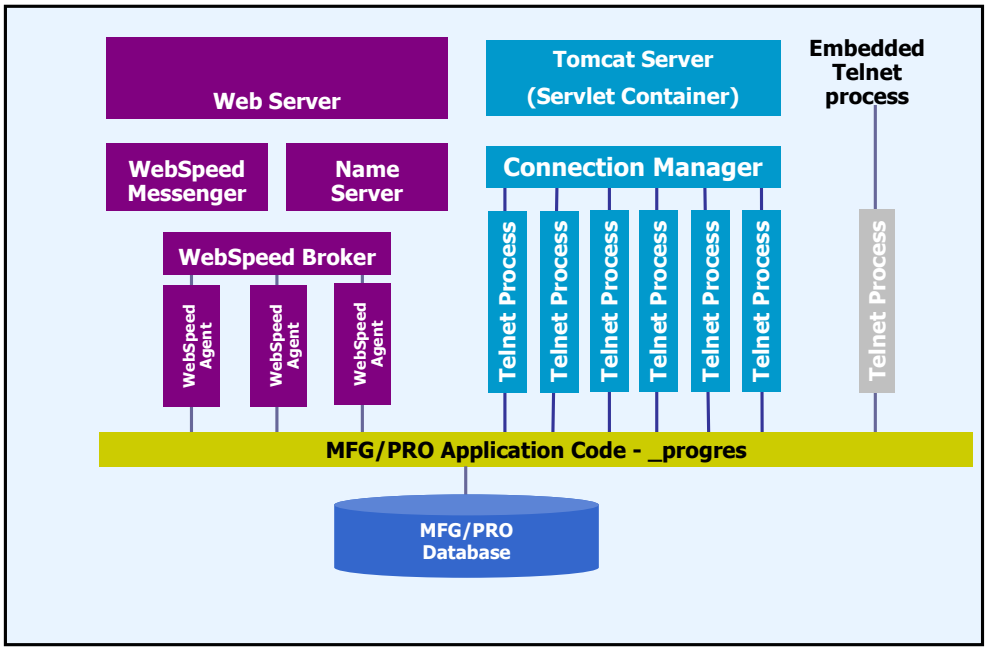
The configuration you choose depends on the number of users concurrently accessing the system, but can be affected by other factors as well, such as how your users are geographically located and whether your system is required to support other kinds of users (such as character session users).

> ***Note*** The below numbers are for UNIX systems only. As stated in the UNIX versus Windows section, Desktop on Windows presently has a 30 concurrent user bottleneck per server.

Single-Tier Configuration (1 to 100 concurrent users)

- In this configuration, Desktop components run on the same machine as MFG/PRO databases.
- The server should have at least 512 MB RAM free for use by Tomcat. At peak active hours, the server's CPU should average 70% utilization or less. The additional CPU required to run Java processes and the XML engine could overburden a database server that is already pushing the upper limit of utilization.

Figure 4. Single-Tier Configuration

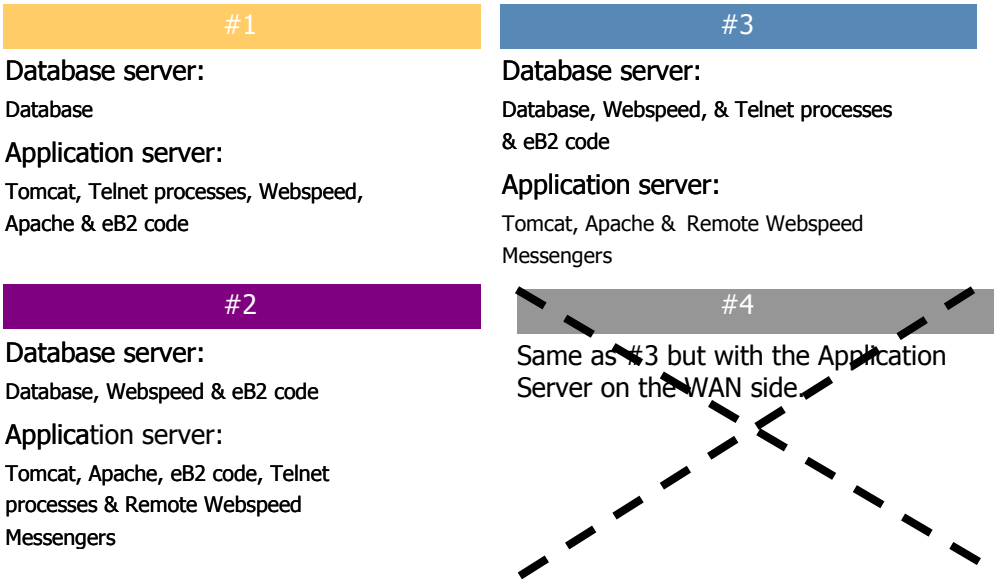


Two-Tier Configuration (100 or more concurrent users)

- In the two-tier configuration, there is a database server and an application server.
- The database server hosts the MFG/PRO database while the application server hosts Desktop components including the Web server and Tomcat.
- The application server should have a dual processor, minimum speed of 1.2 GHz, and 2.0 GB of RAM. The recommended operating system is Linux.

Theoretically, there are four primary ways to organize a two-tier configuration:

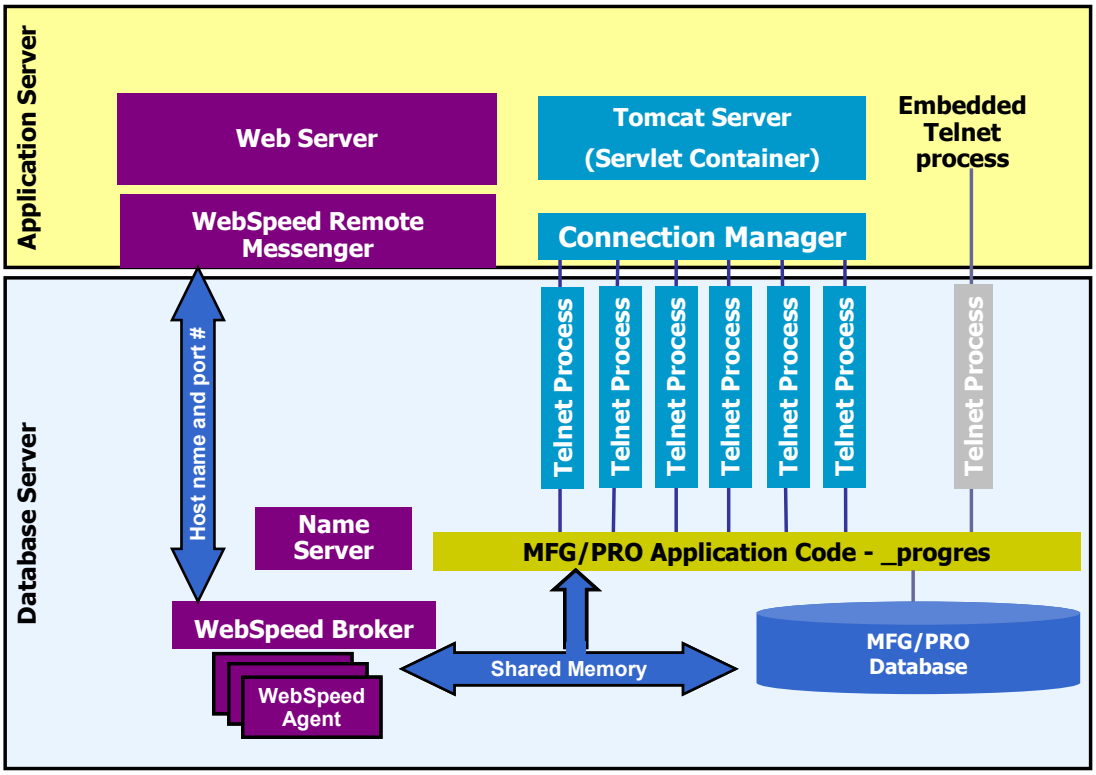
Figure 5. Two-Tier Configuration Options



- **Configuration #1** is set up with the application server running the Web server, WebSpeed, Tomcat, and the Connection Manager sessions. The WebSpeed agents and Connection Manager `_progres` sessions connect to the MFG/PRO database using Progress Client Networking. WebSpeed is used for login, browses, and lookups in Desktop. Tomcat is responsible for all the HTML maintenance screens in Desktop.
- **Configuration #2** also relies on Progress Client Networking for the Connection Manager `_progres` sessions. The difference is that the WebSpeed agents run locally in shared memory mode on the database server. A WebSpeed remote messenger handles requests on the application server and forwards them to the WebSpeed agents running on the database server. The agents do all the actual processing of 4GL code and then return the results to the remote messenger.
- **Configuration #3**, the Web server and Tomcat run on the application server while both the WebSpeed and Connection Manager sessions run in shared memory mode on the database server. The Connection Manager `_progres` sessions connect to the database server via telnet and process all MFG/PRO code locally, bypassing any network protocols. Only the results are sent back to the Connection Manager and Tomcat.
- **Configuration #4** is the same as Configuration #3 except that the application server is located on the WAN side along with remote Desktop clients. Latency tests reveal that this is not a viable deployment configuration at this time. Refer to the Latency section for details.

Based on QAD's findings, configurations that use client networking, such as Configuration #1 and Configuration #2, consume greater amounts of bandwidth with large browses and reports**.** If your company performs large browses and generates large reports, QAD does not advise running in these configurations. QAD is currently working on improvements in this area.

Figure 6. Two-Tier Configuration



N-Tier Configuration (100 or more concurrent users)

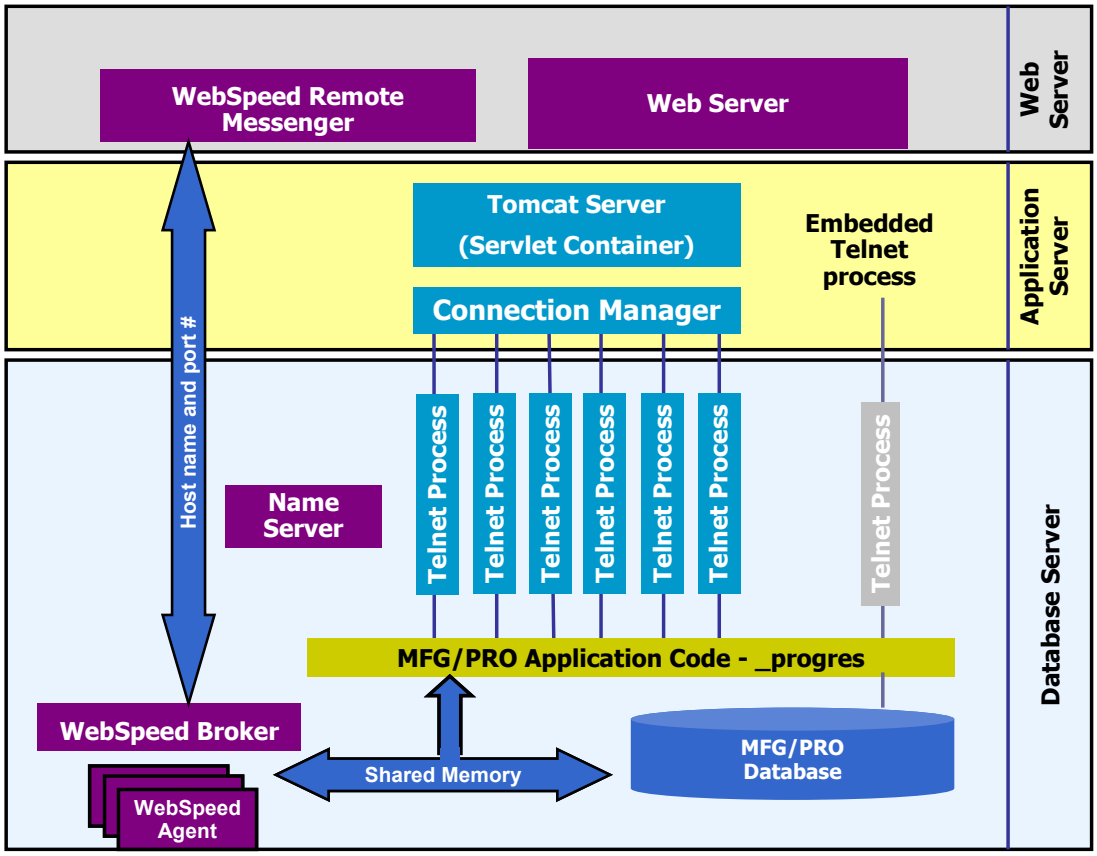
Desktop installations can be extended to run in an n-tier environment. An n-tier configuration goes beyond the basic two-tier configuration by optionally splitting out the Web server from the application server, as well as providing the ability for multiple servers at each deployment layer.

The reasons for utilizing an n-tier deployment structure include:

- A very large user base
- A geographically dispersed environment
- Security considerations
- Redundancy and failover

N-tier deployments can scale both vertically and horizontally. By breaking out the Web server from the application server and putting it on a dedicated machine, the Desktop deployment is being scaled vertically.

Figure 7. Vertical n-Tier Configuration

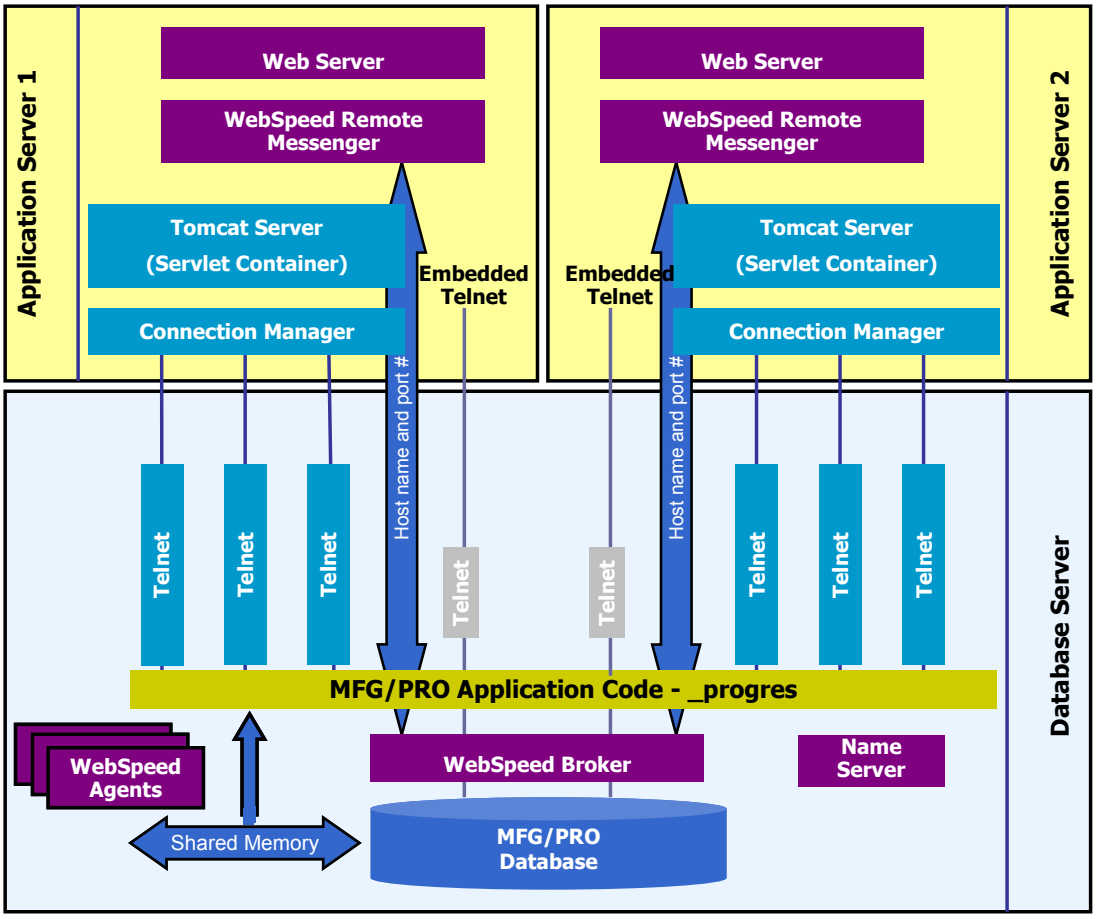


This type of scaling distributes the burden of the application server and allocates more resources to both the Tomcat and Web server components.

Vertical scaling may be done for security purposes as well. Although not configured by default, Desktop can be set up so that all server requests (including Tomcat) must be made through the Web server, typically on Port 80. This way, the Web server is isolated behind a firewall or set of firewalls, as in a DMZ.

Desktop can also be configured to have multiple application and Web servers running simultaneously. In this type of deployment, the servers are scaled horizontally. Benefits of horizontal n-tier scaling include the ability for load balancing of users and failover of servers. Also, horizontal n-tier deployment allows multiple sites to efficiently access a central database.

Figure 8. Horizontal n-Tier Configuration



This figure shows that the application servers can reside at single or multiple locations. For the greatest scalability, consider using a combination of vertical and horizontal n-tier configurations.

Conclusion

The starting point of any performance tuning is to set realistic goals for how well the application can be expected to perform within a given system. In the case of a Web application, this is often a matter of knowing what loads the server and network must bear and how well the client machines can render pages within a browser.

Even with no specific knowledge of system configuration and hardware, better hardware enables better overall system performance. Further, it often pays to consider deploying Desktop components in such a way that the load on system CPUs is more evenly distributed. Finally, as with any Web application, understanding the network and its potential bottlenecks is necessary to optimize performance.

Because simple, small-sized systems may have simple single-tier solutions, following the installation documentation, as well as the guidelines documented here, may provide sufficient performance results.

Two-tier and n-tier setups can be technically complicated, and there is no substitute for experience when troubleshooting a problem in such a deployment. More complex systems are best analyzed by experienced technical consultants in order to find the optimal solution. For this reason, it is recommended that QAD Services (or qualified QAD service partners) be employed for any setup that goes beyond the expertise of a system administrator.

QAD is continuously testing and improving solutions. For the latest information on Desktop, check for updates on http://www.qad.com/.