documentum

Documentum Performance and Tuning

October 1999

Authors:

Ed Bueche' [edward.bueche@documentum.com]

Chase Harris [chase.harris@documentum.com]

documentum

TABLE OF CONTENTS

documentum

Executive Summary

This document is meant for Documentum Administrators, application developers, and Integrators and is intended to be a tool for improving performance of customer deployments. It is meant to help speed up deployments of Documentum's software by providing techniques and tips to diagnose performance issues and tune them. The information is applicable for pre- and post-deployment situations.

By far the most important suggestion is to avoid performance issues by doing some important work up front in a project's "development" phase. Preventing performance issues always leads to significant savings later on. The top suggestions we have to prevent performance issues during deployment and production phases are to:

- Compile a list of top, most-frequently-executed "operations" (or commands) and their desired response time,
- Instrument your "custom" application so that response times can be measured,
- Test to ensure that these response times are met during development testing,
- Attempt to conduct pre-deployment performance tests using similar sized Docbases as those used in production in similar environments (systems and telecommunication infrastructure), and expected user load, and
- Ensure that Administrative personnel are well trained in OS, RDBMS, and Documentum administration.

In many cases it is difficult to measure response times from the application. For example, the deployment might just entail customizing Documentum's off-the-shelf client software (like the Desktop client in Documentum 4i). When this occurs response time can be measured using the Documentum DMCL trace facility. This document describes how to enable and interpret this trace information. We also provide some heuristics to help engineers understand whether the performance "issue" is a client-side one or a server-side one.

Once it has been determined that the problem resides on the server side of the application, various tools can be brought to bear to help diagnose and tune the issue. We survey both Operating System level tools and Relational Database-level tools that are useful in this purpose. From the Operating system level we cover both Windows NT and UNIX tools useful to measure CPU and memory consumption, and disk activity. There are many specialized tools for this purpose in the market place. We cover the *free* ones typically shipped with the OS.

We treat the RDBMS tools in a similar fashion. We examine some of the basic areas that typically need to be tuned: buffer caches, query plans, and disk space. We focus on Oracle, Sybase, and the Microsoft SQL Server. Again, there are many specialized tools to make this effort easier, we, however, cover the tools provided by the vendor with the product.

Finally, we cover Administrator and Developer-center issues around the Documentum Docpage (EDM) server and the RightSite server.

For the Administrator, Integrator, and Developer we hope this document is useful in providing a survey of tools and techniques that can be used to successfully solve performance issues. For managers this document will hopefully help in the planning process to avoid performance issues and to help understand the types of tools (and resources) that need to brought to forward to solve them.

Overview of Architecture

Documentum employs a complex architecture that is configurable in different ways depending on a number of influences such as:

- distribution of users and content
- usage patterns
- administration requirements

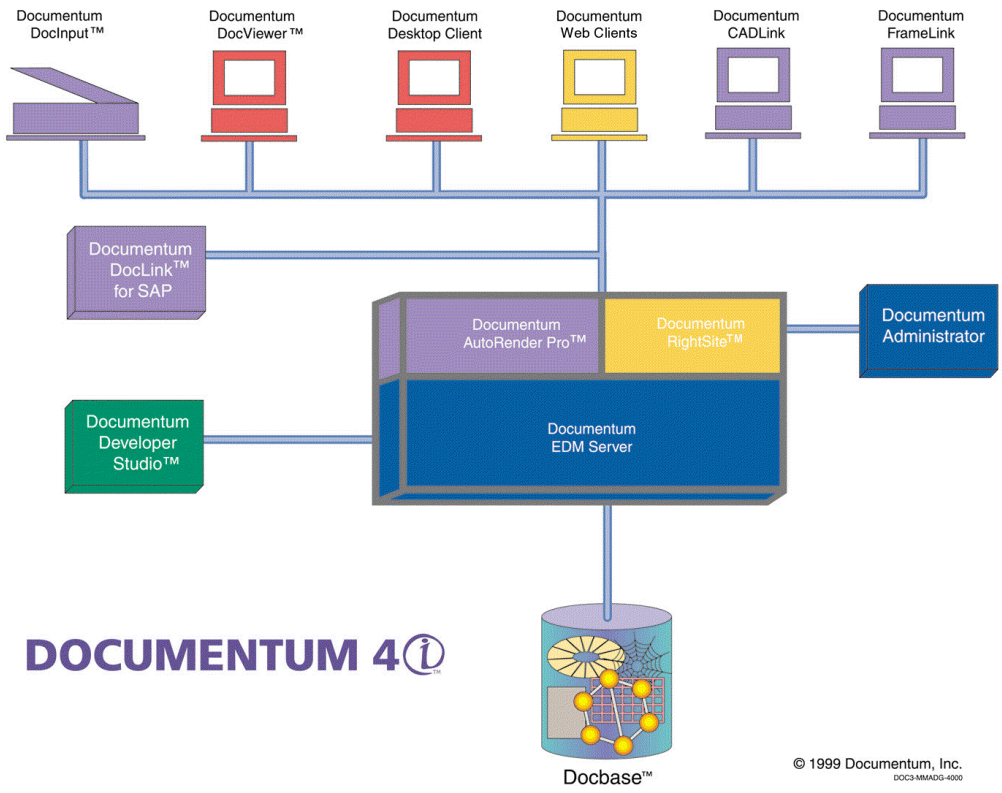


FIGURE 1: Documentum Product Family

Access to content and to attributes is possible over the Web or using popular tools from Microsoft, Documentum, and other third parties. This is shown above.

The base-server technology consists of Documentum's Enterprise Document Management (EDM) server (in pre-Documentum 4i versions also known as the DocPage server). This server provides a variety of services that include flexible storage management, transaction monitoring and resource allocation, compound document support for OLE and SGML documents, image management, custom relationship definition and management, workflow, configuration management, and event notification for all system and document users. This server is supported on UNIX (Solaris, AIX, and HP-UX) and Windows NT. Client software request services of the EDM server via DMCL (the Documentum Client library).

The Documentum RightSite is the Web-based server technology that links corporate intranets with web sites and enables content access for multiple users from their Web browsers. RightSite serves up dynamic Web pages, created in response to a user query and brings all Web pages

under the secure control and management of the Docbase respository. The RightSite server interacts with the EDM server using DMCL.

Documentum's server products employ a flexible N-tier, multi-server architecture that scales with a customer's load. This architecture can run on single hosts or multiple server hosts, depending on the capacity of the servers and the administrative preferences of a company. This is illustrated below.

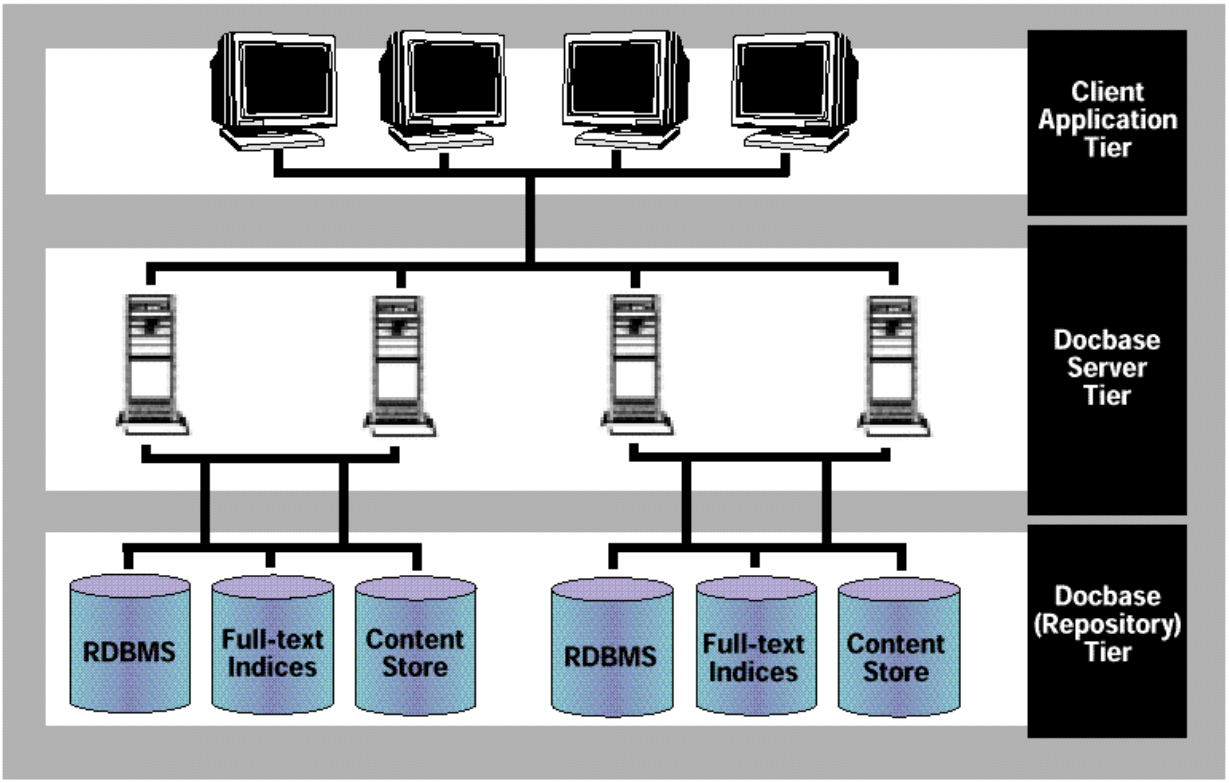


Figure 2: Documentum's N-Tier, parallel-server architecture

Documentum also supports a full range of distributed, replicated, and federated sites. Your environment may be configured in one of the following ways:

- Single Central Docbase: All content and attribute information reside on a single site. This site, of course, could be a large multi-tier server complex, but it is essentially in one place and managed as a single unit.

- Single Docbase - Multiple Servers with Content Placement: In this case content can be placed close to remote users for efficient network access. Enterprise sites are typically geographically dispersed, and in some cases it is quite economical to locate content close to the users that access it frequently.

- Single Docbase - Multiple Servers with Content Replication: In a replication scheme the content can be replicated to other sites in an effort to ensure that all remote users can have

fast local access to the documents that they most use. Web sites can also be versioned and managed by replicating web content to a Docbase on the other side of a corporate firewall.

- Multiple Docbases in a Federated Environment: A group of cooperating Docbases that share a common definition of documents, users, groups, and ACLs is called a federation.

The complexity of this environment provides many areas in which performance bottlenecks can occur. For example, the customized application may use inefficient coding techniques, the database server may not be optimally configured, the client or server machines may be under-configured for the workload, or the network bandwidth may be inadequate to support the requirements of the application.

This document is meant to help you isolate and resolve performance issues in any Documentum environment.

Underlying Database Structure

Documentum applies an object-oriented layer on top of the relational structure of the underlying database. The object attributes are actually stored in a number of tables. This is managed by the DocPage server, and is in most cases transparent to the users and applications. The DocPage server will translate DQL and API calls to the appropriate SQL statements for the RDBMS back-end. Each object type will have up to two corresponding tables in the database. For example, consider the object structure for company ACME:

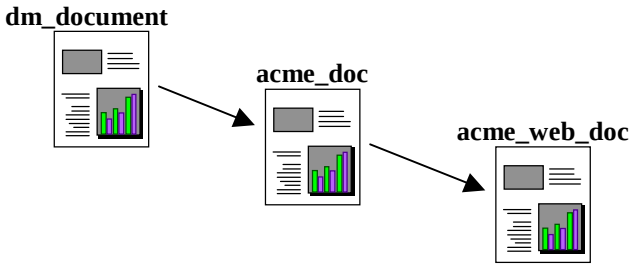
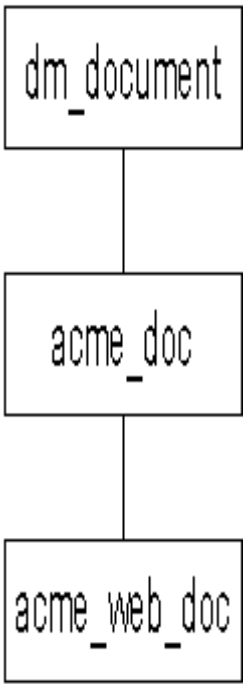


Figure 3: ACME's types

All corporate documents have a common set of attributes. These are defined in an object type called "acme_doc" which is a subtype of dm_document.

In addition, ACME is managing web content through the docbase, so they've defined a subtype of acme_doc that contains web-specific attributes.

This hierarchy and a subset of the object attributes can be displayed as follows:



dm_sysobject/dm_document

Attribute	Attribute Type	Repeating?
r_object_id	dmID	N
object_name	STRING	N
title	STRING	N
keywords	STRING	Y
i_contents_id	dmID	N

acme_doc (supertype dm_document)

Custom Attribute	Attribute Type	Repeating?
department	STRING	N
product	STRING	Y
document_class	STRING	N

acme_web_doc (supertype acme_doc)

Custom Attribute	Attribute Type	Repeating?
publication_date	TIME	N
review_date	TIME	N
expiry_date	TIME	N

In Documentum 4i, the underlying table structure is:

dm_sysobject_s
r_object_id
object_name
title
i_contents_id

dm_sysobject_r
r_object_id
i_position
keywords

acme_doc_s
r_object_id
department
document_class

acme_doc_r
r_object_id
i_position
product

acme_web_doc_s
r_object_id
publication_date
review_date
expiration_date

If a custom type consists only of additional single- or repeating-valued attributes, only the _s or _r table will be created. Note in the above example that there are no repeating attributes on acme_web_doc, so there is no acme_web_doc_r table.

In EDMS98 and prior releases, the underlying table structure is slightly different:

dm_sysobject_s
r_object_id
r_object_id_i
r_object_id_ti
object_name
title
i_contents_id
i_contents_id_i
i_contents_id_ti

dm_sysobject_r
r_object_id
r_object_id_i
r_object_id_ti
i_position
keywords

acme_doc_s
r_object_id
r_object_id_i
r_object_id_ti
department
document_class

acme_doc_r
r_object_id
r_object_id_i
r_object_id_ti
i_position
product

acme_web_doc_s
r_object_id
r_object_id_i
r_object_id_ti
publication_date
review_date
expiration_date

Some important things to note for EDMS98 docbases:

1. Every dmID type attribute is represented by 3 columns in the underlying tables. The r_object_id is a character representation of the ID, and the _i and _ti values are integer representations. These columns are maintained automatically by the DocPage Server.
2. The primary key of the _S tables is the r_object_id_i column, not the r_object_id column. The primary key of the _R tables is composed of r_object_id_i and i_position. **The r_object_id column is not indexed.** This is especially important when retrieving objects using an ID value, and is explained thoroughly in section on Documentum Application developer performance issues.

Why is this important?

When designing, developing or analyzing the performance of a Documentum application, it is important to understand the underlying table structure. Even a simple query such as:

```
select r_object_id, object_name, department, product
from acme_web_doc
where expiration_date is nulldate
```

will result in a five-table join being performed at the RDBMS level.

If care is not taken to select only those attributes that are required, and to construct WHERE clauses with indexed columns, the resulting join could present a very apparent performance problem!

documentum

Performance Issues: Preventing and Measuring

In this section we should cover some common origins for performance problems, how they might look in a Documentum environment, and how to get some basic information on them.

Performance problems in Documentum are typically manifested with poor response time for interactive commands an/or poor throughput for batch operations. The key in resolving these problems is to properly replicate the problem and characterize it in a measurable way. When measuring the problem it is important to focus on the poor response time or the poor throughput as the major metric. That is, it is less useful to characterize a problem as: “response time is slow and the CPUs of the server machine are at 80% busy”. In this case the cause of the slow response time may not have anything to do with the 80% CPU busy. Its far better to characterize the problem as: “When operation X is executed, the response is 60 seconds, when clearly it is expected to be 3 seconds”.

Performance issues can be tuned away, fixed in software, or eliminated by buying additional hardware. In the latter case, the problem is a special type of performance problem called a “capacity planning” issue. Although for most of this paper we concentrate on identifying problems, its good to understand the origin or performance issues. In most cases, preventing performance issues is significantly less costly than fixing them later. The following are some common origins to performance problems:

- Insufficient detail on user response time and operation throughput requirements,
- Infrequent or incomplete tracking and testing of performance during application development cycle,
- Inability to simulate the production environment within a development infrastructure,
- Insufficient hardware or networking resources to support the production load,
- Lack of knowledge of DBA and OS administration activities, and
- Interference of other applications running on the same host.

Many of the above are straightforward, however, the one on requirements and testing deserves some attention. Application developers or administrators should have a list of top metrics that they think the users will be using to judge the performance of their system. For example, a simple matrix might look as follows:

Operation (From Browser)	What this does	Expected Maximum Average Response time
Find and Edit workitem	User fills in dialogue and submits search for workitem. It is then displayed in browser and user can start editing it.	3 seconds from submit time to display of work item.
Complete Work Item	After user fills in work item, then they submit this to be checked in with the appropriate files.	3 seconds from the submit time to notification that checkin was done.

TABLE 1: Example Response time requirements

Initial requirements might have to be adjusted to reflect the realities of the corporate infrastructure. For example, all users would love to have less than 1 second response time for all

commands. But it might be extremely cost prohibited to provision each remote site with the network bandwidth necessary to achieve this.

In summary, our top suggestions for avoiding performance issues are as follows:

- Compile a list of top, most-frequently-executed “operations” (or commands) and their desired response time,
- Instrument your “custom” application so that response times can be measured,
- Test to ensure that these response times are met during development testing,
- Attempt to conduct pre-deployment performance tests using similar sized Docbases as those used in production in similar environments (systems and telecommunication infrastructure), and
- Ensure that Administrative personnel are well trained in OS, RDBMS, and Documentum administration.

Measuring Documentum Response Time

After determining the desired response times, as mentioned above, the next most important task is measuring response times. When developing an application, one “best practice” is to instrument an application such that the user perceived response time can be measured. This, however, may not always be possible, because, for example, one might only be customizing an existing application like the Documentum Desktop Client.

In such cases, Documentum provides some application API measuring capabilities that are sometimes useful in understanding why a particular command is not running as fast as desired. When this API tracing is enabled each DMCL command is written to a file along with its response time. This can look as follows:

```
# Time:        0.000 sec  'OK'
API> fetch,s0,0c246fa6800004c1
# Time:        0.601 sec  'OK'
API> get,s0,0c246fa6800004c1,i_has_folder[0]
# Time:        0.000 sec  'T'
API> readquery,s0,select "object_name", "a_content_type", "r_object_type",
"r_lock_owner", object_name, r_object_id, a_content_type, r_object_type,
r_lock_owner, r_link_cnt, isreplica,"object_name" from dm_sysobject where
folder(id('0c246fa6800004c1')) and a_is_hidden = false order by
"object_name",r_object_id
# Time:        6.699 sec  'q0'
API> count,s0,q0
# Time:        0.000 sec  '12'
API> get,s0,q0,_names[0]
# Time:        0.000 sec  'object_name'
```

Note: Command response time

Each line that begins with “API>” represents the command to sent via DMCL to the DocPage Server and each line beginning with the “# Time:” represents the response time for that command in seconds. Hence in the case above most of the commands took less then 10 msec, except for the readquery command which took almost 7 seconds.

If one or more queries are the problem, then this is typically the best way to identify them. This trace is also useful in determining whether the performance bottleneck is the software acting as a Documentum client or the Documentum Docpage server/DBMS server. That is, given some scenario that maps to a series of DMCL calls, one need only compare the user perceived response time with the sum of all of the response times in the trace. For example, in the trace above the sum of all response times in the trace fragment is 7.3 secs. If the user perceived

response time is 60 seconds, then the something else is likely to be causing the problem aside from the Documentum server. However, if the users are experiencing 8 second response times, then our readquery method above is the major contributor and the solution will require some attention on the server machine.

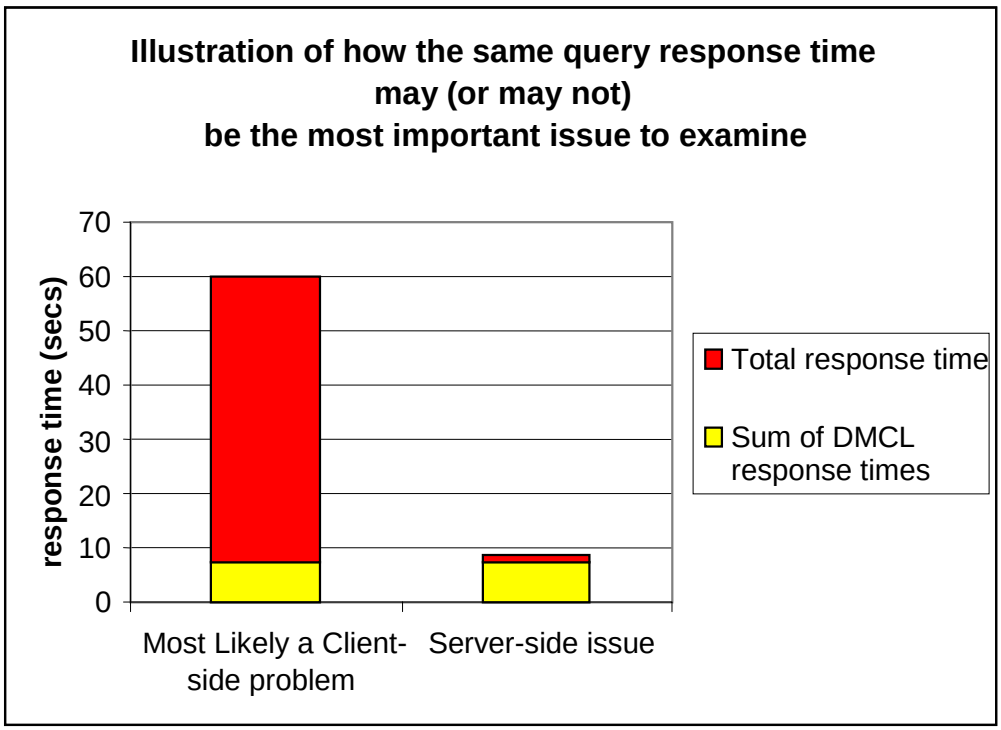


FIGURE 4: How DMCL trace can help identify source of poor response time

Once a problem scenario has been identified as being either a client or a server issue, then more tools can be brought to bear to determine exactly what causes the problem. In this document we will survey many tools that can be useful in determining server-side performance issues. For client side performance issues it is typically best to employ software profilers (like the Numega TrueTime or Rational's Visual Quantify).

This API tracing is enabled in several ways:

1. By issuing the "trace" method to the server (see Documentum Reference Manual). For example at the IAPI prompt:

   ```
   IAPI> trace,c,10,'mytrace.trc'
   ```

   Which, in the above example, turns on the trace and directs the output into the file mytrace.trc.

   This API command can be issued from any language that can make DMCL API calls. For example, from Docbasic:

   ```
   Ret = dmAPIExec("trace,c,10,'mytrace.trc'")
   ```

In addition, from Workspace one can enable tracing by selecting the Help (menu) → About Workspace (item) and pressing the CTRL key at the same time. When this is done an "interactive message testing" dialogue appears. The trace API method can be invoked in the same manner as in the IAPI from this dialogue. Once invoked, all Workspace interactions will be traced until the end of the session or until the interactive message tester is brought up again and the trace disabled.

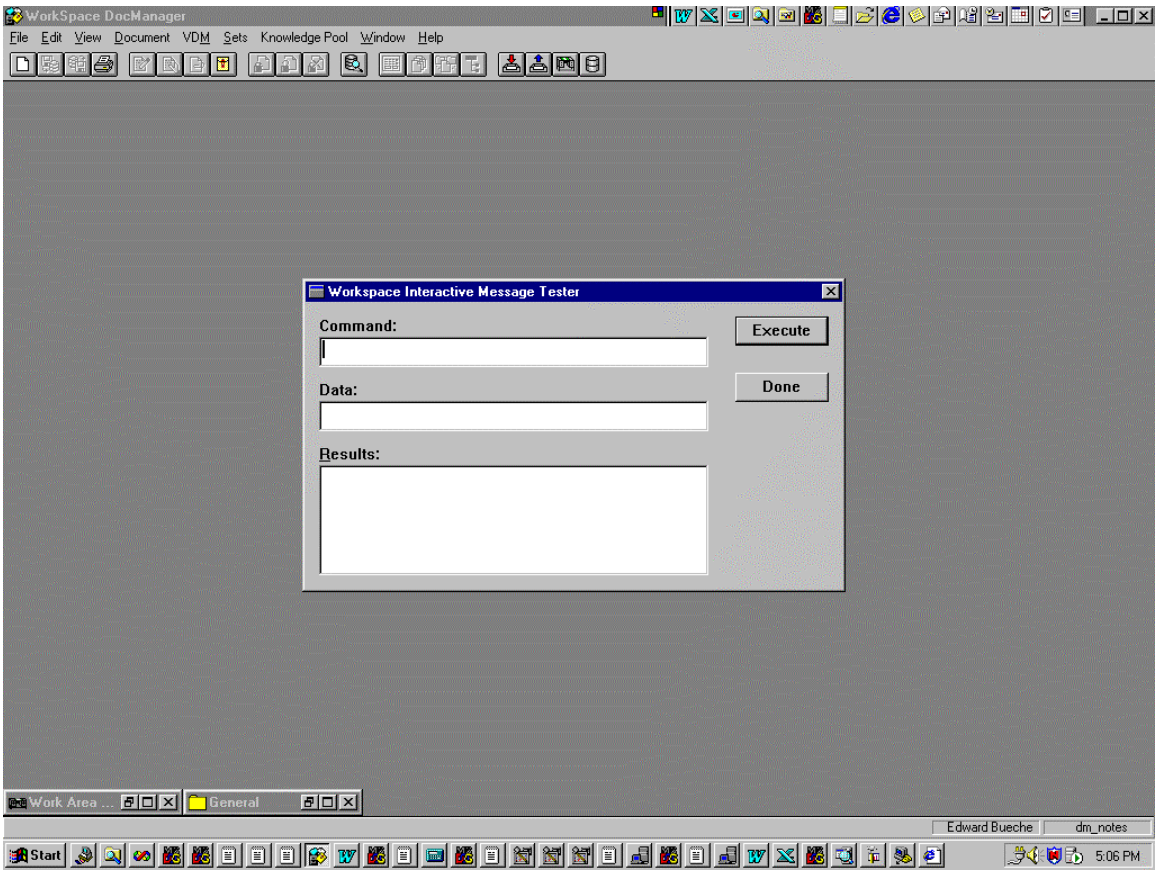


FIGURE 5: Interactive Message tester within Workspace

This can also be enabled in the Documentum Desktop client integration with the Windows Explorer. To do this select the Applications (menu) → QA Tools → Interactive Message Tester (item).

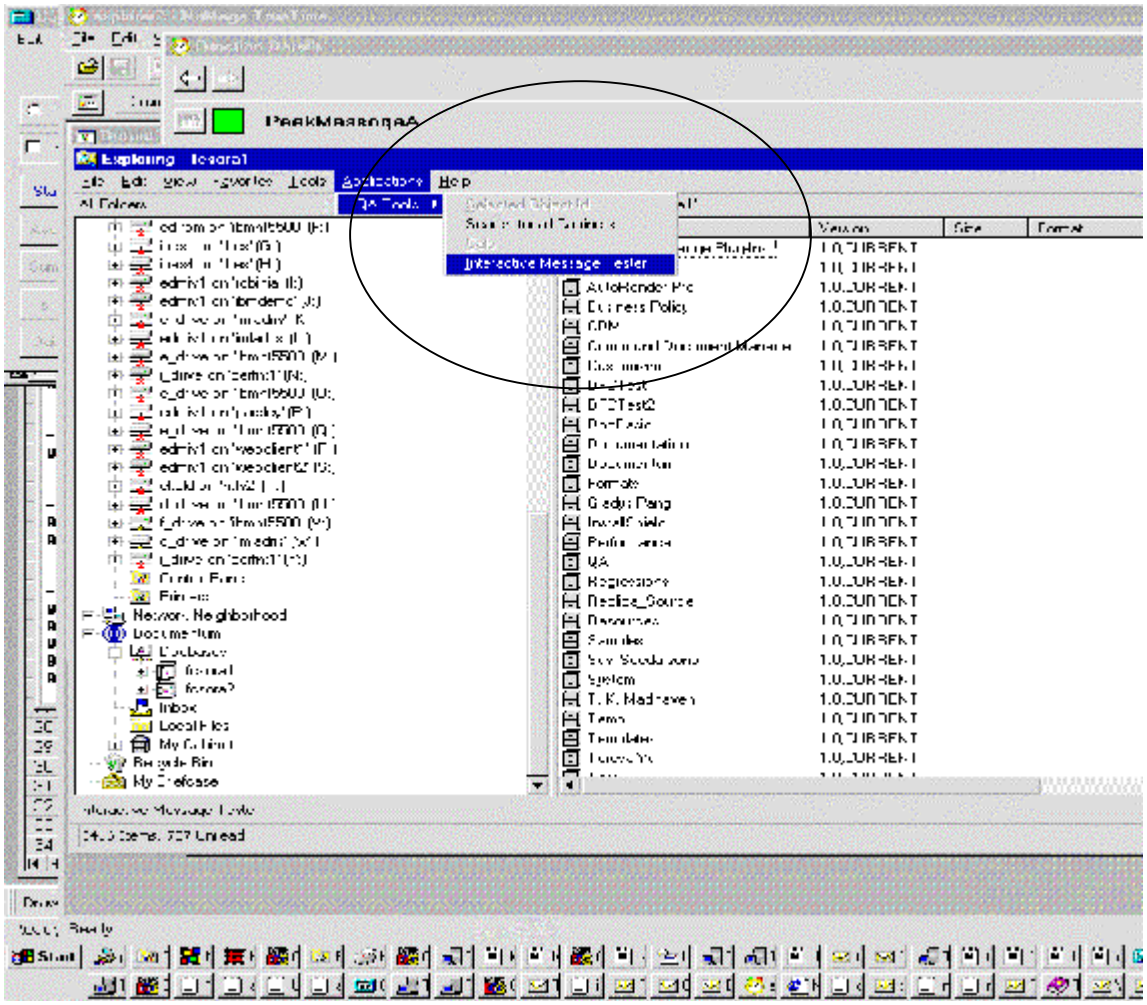


FIGURE 6: Desktop Client Interactive Message Tester

To disable tracing just re-issue the trace method with a level of 0:

```
trace,c,0
```

2. The tracing can also be enabled by turning it on via the client dmcl.ini file:

```
[DMAPI_CONFIGURATION]
trace_level = 10
trace_file = “mytrace.trc”
```

This method is useful when the startup of an application needs to be examined or it is difficult to get the application to issue the trace method. For example, this is probably the easiest way to collect this trace with the Documentum RightSite server.

The downside to this method is that the it traces all sessions that use that dmcl.ini file and the file has to be edited to turn the tracing off. Also, each new session will overwrite the previous session's trace file since they all share the same name for the trace file.

Tracing, however, is not always straight forward. Some Documentum server API's can be customized by user-defined scripts. These scripts need to enable tracing explicitly else the bottleneck might not be easily identified. For example, suppose that we have a workflow that has been customized (using the Workspace interface) to execute a post_forward method. See below.

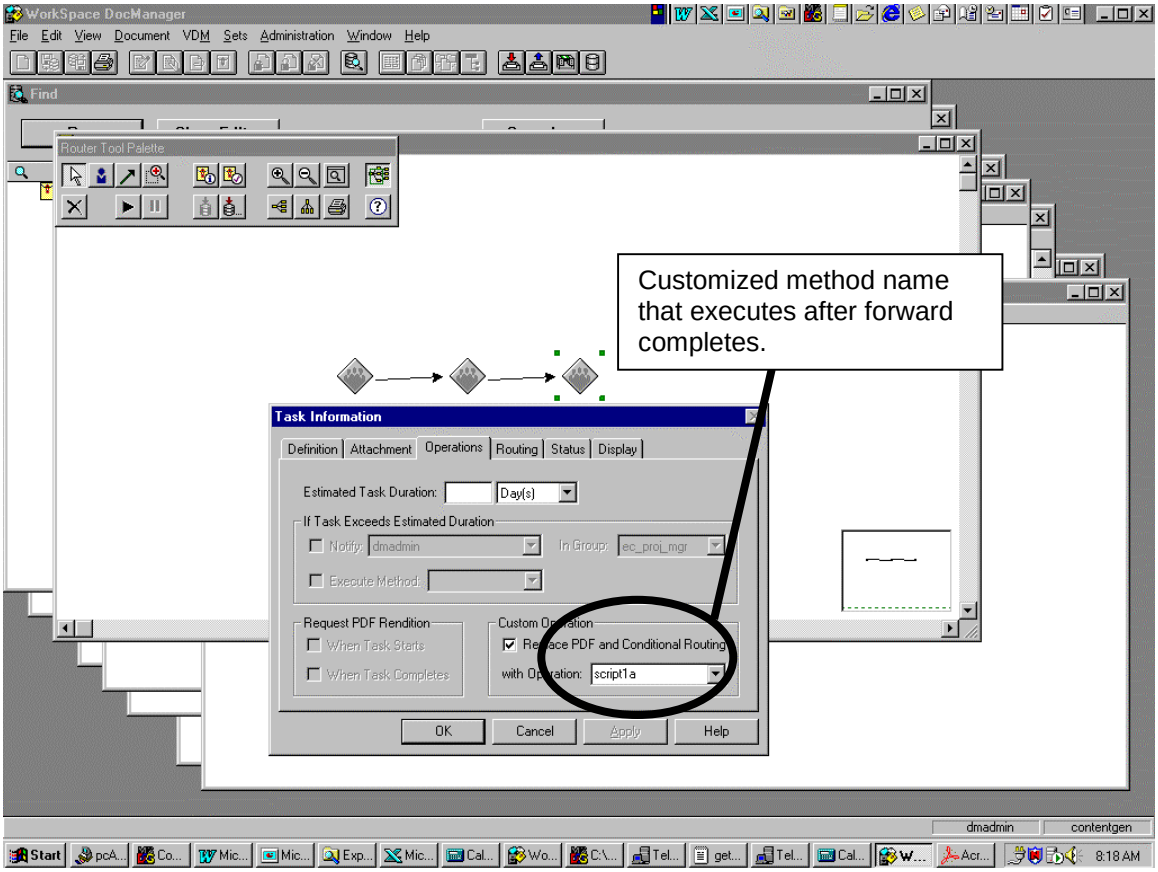


Figure 7: Method customization within Workspace

This method object is located with all of the other methods and it's content object is the script that is executed once the method "forward" is executed. Let us suppose that this method takes long to execute (say 30 seconds). Then when the user traces the operation of forwarding the router (via Workspace), then the trace from workspace will look as follows:

```
API> get,s0,0b003a6780043fd5,_changed
# Time:         0.000 sec  'F
API> forward,s0,18003a6780042b71,2
SE>
# Time:        31.259 sec  'OK'
API> anyevents,s0
# Time:         0.010 sec  ' '
```

Trace shows that server method forward takes 31 seconds. However, the customized script execution time is covered within this time as well.

However, the problem lies not with the forward method, rather the customized script that is run “under-the-covers” once forward completes execution. It would be best to modify that customization script to generate a trace file as shown below (assuming that the customized script was a Docbase script).

```
sub main1

       ‘ The following code fragment will create a trace file in a
       ‘ /tmp with a random suffix ending the file name. It then measures
       ‘ response time using the facility provided in Docbasic and writes
       ‘ this information out to that file.
       ‘
       ‘ Some of the error checking has been stripped from this example
       ‘ to make it easier to read.

       ransess = Random(1,10000)
       filename$ = "/tmp/tempf" & ransess

       On Error goto TempOpenError
       Open filename$ For Output Access Write Shared As #1

       ‘ measure starting time of operation

       start& = Timer

       str1 = "connect,contentgen,dmadmin,yahoo"
       sid = dmAPIGet(str1)
       if (sid = "") then
              dmExit(-1)
       end if


       ‘ Turn on Documentum trace as well.

       apitrace = "'/tmp/tempf_api" & ransess & "'"
       rci = dmAPIExec("trace," & sid & ",10," & apitrace)
       if (rci = 0) then
              dmExit(-1)

       end if

       ‘ Real work is done here!!

       rci = dmAPIExec("disconnect," & sid)
       if (rci = 0) then
              dmExit(-1)
       end if

       ‘ Take ending time stamp and write out the information to
       ‘ the trace file

       endt& = Timer
       iline$ = "connect time " & endt& - start&
       print #1, iline$
       close #1
       exit sub
TempOpenError:
              print "Open of tempfile failed " & tempfile$

                 print "END: FAIL"
              Exit sub

end  sub
```

documentum

OS and Hardware

In the previous section we discussed Documentum application tracing as a tool for figuring out which command consumes most of the time. It is also typically useful to monitor and trace at the OS and RDBMS level. In this section we survey some useful points on Operating System-level tools that can help to characterize performance issues. Later we will do the same for Database tools.

The Operating System-level tools are essential for measuring the consumption of the CPU, memory, and disks. On shared systems (i.e., systems that run several types of applications (or servers) they are also useful in figuring out which application (or server) is consuming the majority of the resources.

In this section we will cover tools on both Windows NT and on UNIX.

Using the Windows NT Performance Monitor with Documentum

This section will provide lots of detail on how to collect and analyze data from the NT performance monitor in a Documentum Deployment. The focus will be to outline the most important objects/counters relative to Documentum and how to collect and view performance monitor logs. In later sections we will illustrate things using the NT performance monitor based on the background provided in this section.

Windows NT is shipped with an excellent tool for measuring and analyzing performance called the Windows NT Performance Monitor. This tool and its uses are documented extensively in sources such as the Windows NT Resource kit and in white papers from Microsoft. In this section we cover some of the “best practices” in using this tool to analyze Documentum performance.

There are several concepts that we examine:

- Performance Monitor basics (background)
- Important Performance Monitor objects and counters,
- Charting vs. Reporting
- Logging vs. real time viewing
- Exporting a chart or report to a file
- Changing the scope of analysis
- Turning on the NT disk counters

In later sections we will use the information outlined in this section to illustrate how to diagnose certain problems.

Performance Monitor Basics

The NT performance monitor is a utility that comes with NT to measure system and application performance. The various OS subsystems and selected applications are setup to generate performance data that the NT performance monitor will sample. This information can be viewed in real time or logged to a file and viewed later. The NT performance monitor is located at Start→Programs→Administrative Tools (Common)→Performance Monitor.

Important Performance Monitor Counters and Objects

When analyzing Documentum performance the following are the most important (or noteworthy) counters:

Object	Counter	Notes
Logical Disk	Disk Transfers/sec Disk Reads/sec Disk Writes/sec	These three counters are "throughput" based counters that can provide an estimate as to how busy a disk is (or set of disks are). Although some single disks can perform 100 disk sequential IO's per second, this is typically not possible for random disk IO's.
	Avg. Disk Sec/Transfer Avg. disk Sec/Read Avg. Disk Sec/Write	These counters are an indicator of the response time for the subsystem. The time (in seconds) includes the time the disk request is queued waiting for service. These counters can be an excellent way to spot disk bottlenecks or RAID configurations that don't work well. Normally, a the Avg. disk secs/write is 15 msecs or less. Some RAID 5 configurations can drive this time up to 40 msec per write.
	Avg. Disk Queue Length	This can be used to measure disk bottlenecks. A large avg. queue length is typically a good indicator.
Memory	Available Bytes	This counter indicates the number of bytes of unused memory. This counter is a useful indicator of whether or not additional RAM is needed.
	Pages Read/sec Pages Write/sec Pages Out/sec Page In/sec	These counters can be useful in determining if paging is causing any disk IO. In NT memory is written out to the paging file as the Operating System attempts to trim memory. This keeps memory usage low, but could lead to excessive disk IO to the paging file.
	Pages/sec	This is an indicator of the paging activity. Paging activity could be an indication of memory contention (i.e., everything does not fit in memory). Some page faults however, do not lead to any disk IO.
Network Interface	Packets/sec Packets Sent/sec Packets Recv/sec	These throughput counters can be good indications of bottlenecks in the networking hardware of a server. However, 100Mbps interface boards can process a large amount of packets per second.
	Bytes Total/sec Bytes sent/sec Bytes Received/sec	
Objects	Processes	On a system with just the DBMS and the EDM Server this counter is not very useful. However, on systems with RightSite it can be used to get an external count of number of active RightSite sessions.

	Threads	Both Oracle and Documentum have one thread per connection.
Paging File	% Usage % Usage Peak	This counter is useful in determining if additional paging file space is needed.
Processor	% Privileged Time % Processor Time % User time	These counters show CPU time for each individual processor. They are most useful when trying to spot any unbalanced usage of CPU (or when something is bottlenecked with a single threaded application).
Process	% Privileged Time % Processor Time % User time	These counters are not very useful for multi-threaded processes like SQL Server, Oracle or the Docpage server. They are useless for multiple-process architectures like RightSite.
	Page Faults/sec	This can be an indication of whether this process is having any memory contention with another process. However, as a process grows it will have many virtual page faults that are not cause for alarm.
	Page File bytes	How much this process is consuming in the paging file.
	Thread Count	For Oracle and the EDM server this is a good indication of the active sessions.
	Working Set Working Set Peak	These counter are a good indication of how much real memory the process is consuming.
System	% Privileged Time % Processor Time % User time	These counters are a good indication of the total CPU consumption. These are single values that summarize the values over all processors.
Thread	% Privileged Time % Processor Time % User time	In general, these counters are not much use for the Documentum architecture. This is primarily because when a session is made inactive (because, for example, the 5 minute inactivity timer expired) the thread will be destroyed. The Performance Monitor does not allow one to track such ephemeral instances very well.
SQL Server	All objects and counters	
Oracle / NT	All objects and counters	

TABLE 2: Performance Monitor counters of Interest

Logging vs. Real time display

The NT performance can either examine current data that is generated in real time or it can view data that has been previously logged. Real time display is less useful in most cases than logging for display later. Logging allows for the capture of the data to be analyzed off-line, after the test is over or the scenario has stopped. When viewing a “real-time” display, the information is off the display fairly quickly. This could lead to having to rerun the scenario often, which is painful. In addition, real-time display is more costly in CPU time than logging. After performance data has been logged it can be viewed and analyzed in many different ways. In addition, a log file can be sent to someone else for analysis.

To enable logging on the performance monitor select View→Log. This will switch the mode to logging mode. The next step is to define a list of objects to log. The performance monitor does not allow for specific counters to be logged, rather, the entire object and all of its counters are logged.

The next step is to select Options→Log to get the dialogue for defining the logfile name and the sampling interval. This is shown below.

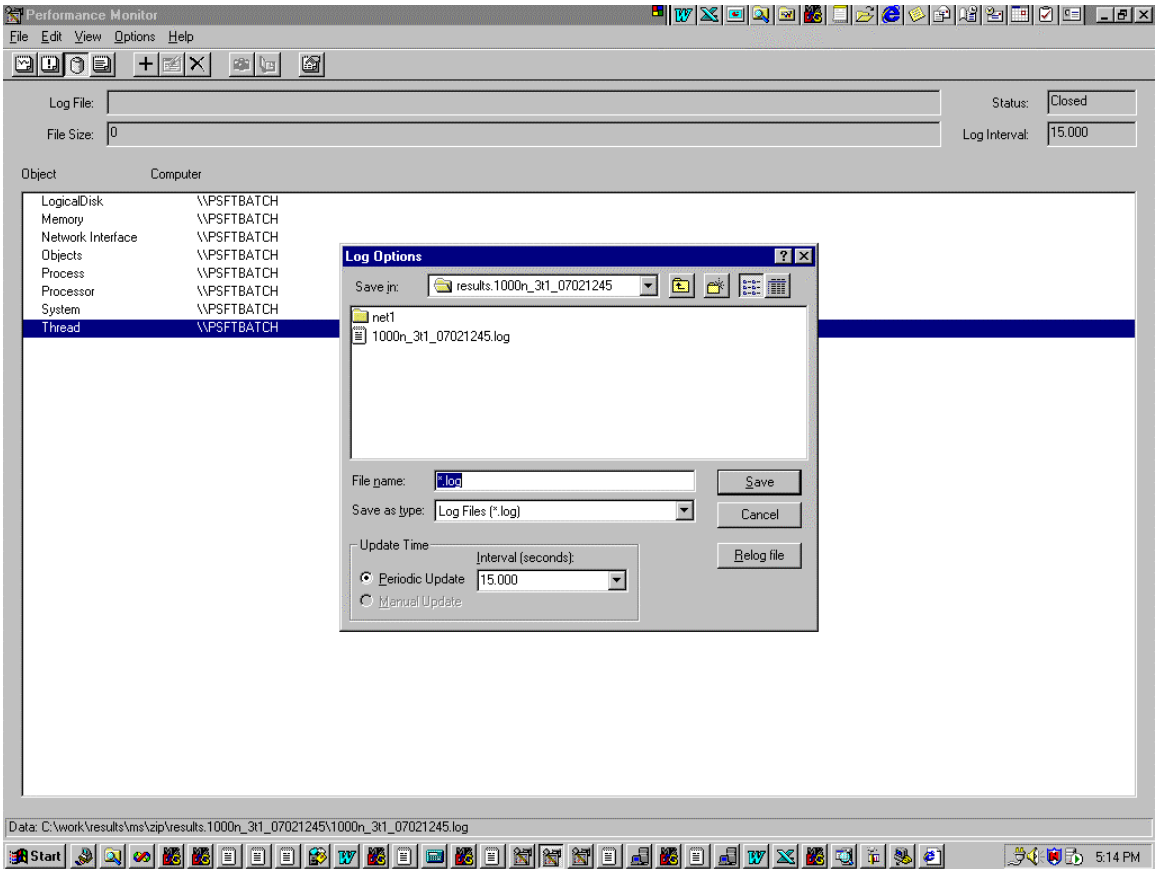


FIGURE 8: Creating a log with the Performance Monitor

From here one can decide the log file name and the sampling interval. The default sampling interval is 15 seconds, which is sufficient unless the logging time is going to be excessive. That is, logging on 15 second intervals for several hours will create a very large file. To start the logging just press the “Start Log” button on this dialogue. When this is done the “status” (top right hand side of the screen) will indicate a state of “Collecting”. When one desires to stop the logging this Options→Log dialogue will show a “Stop Logging” button for this task.

To view a log just start up a performance monitor, select Options→Data From (while in Chart mode) and pick the desired log file. This is shown below. Once the log file has been read various counters can be selected and displayed in chart mode or report mode.

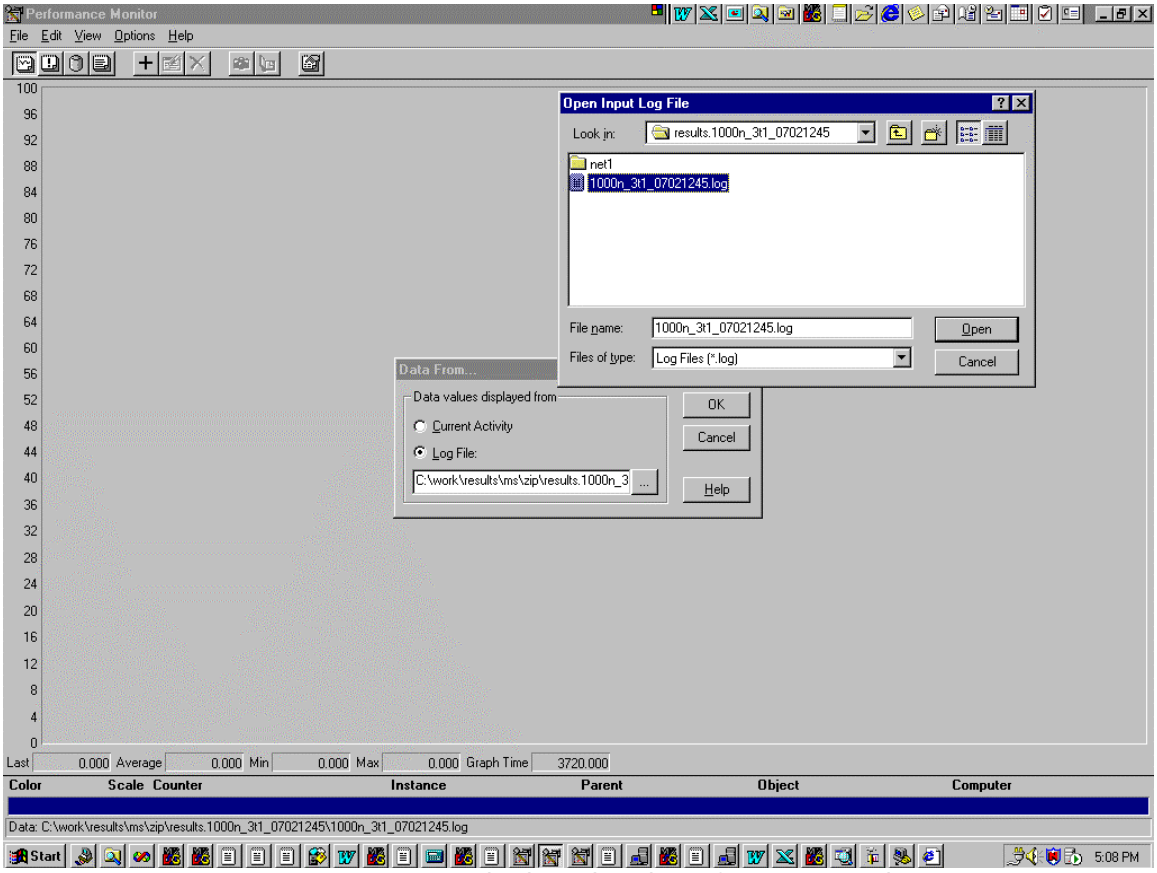


FIGURE 9: Viewing a log via performance monitor

Charting vs. Reporting

There are two modes of analyzing the data in the performance monitor: charting and reporting. They provide different views, but are both useful in different circumstances. A chart provides the changes in a value over time, while the report provides average statistics.

Charting is most useful when trying to understand some transient peak behavior or when trying to narrow down the test sample to the most important part of the test. For example, suppose we wanted to analyze some performance counters for a test period where the Documentum application was servicing many users. This particular test ran for more than an hour and then all clients exited when the test ended. One good way to get an indication of what time period to study is to plot the % Total Processor counter for the System object. To do this select Edit→Add_to_Chart and when the Add to Chart dialogue appears select the System object and then the % Total Processor time. When this is plotted it will look as shown below. Notice the “ramp-up” (test is starting) and “ramp-down” (test is completing) activity.

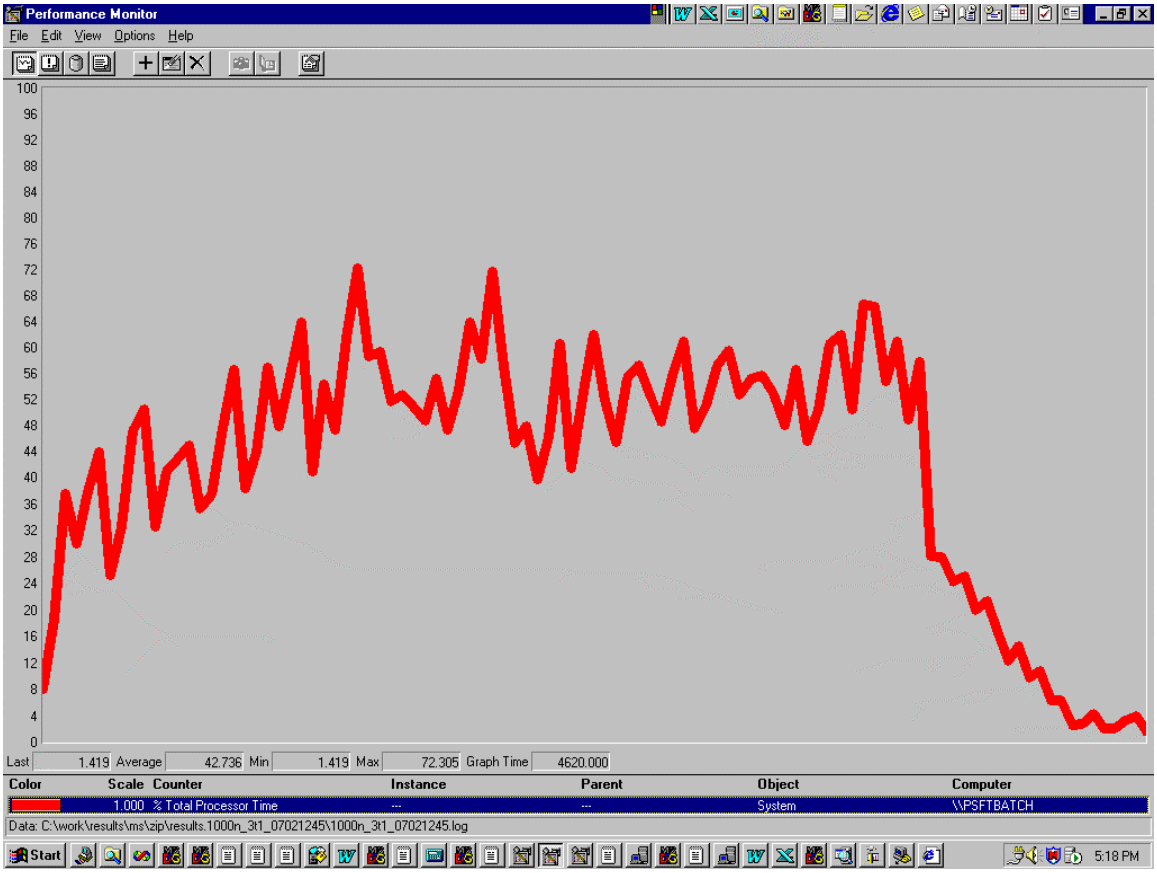


FIGURE 10: Charting CPU consumption from a previously saved log

We are less interested in the results at the end of the test when the clients were finishing their work. To exclude these results from the sample we modify the Time window by selecting the Edit→Time window menu item and then move the sliding bars to exclude the results we don't want. Once this has been changed then all results will be displayed relative to the time period selected.

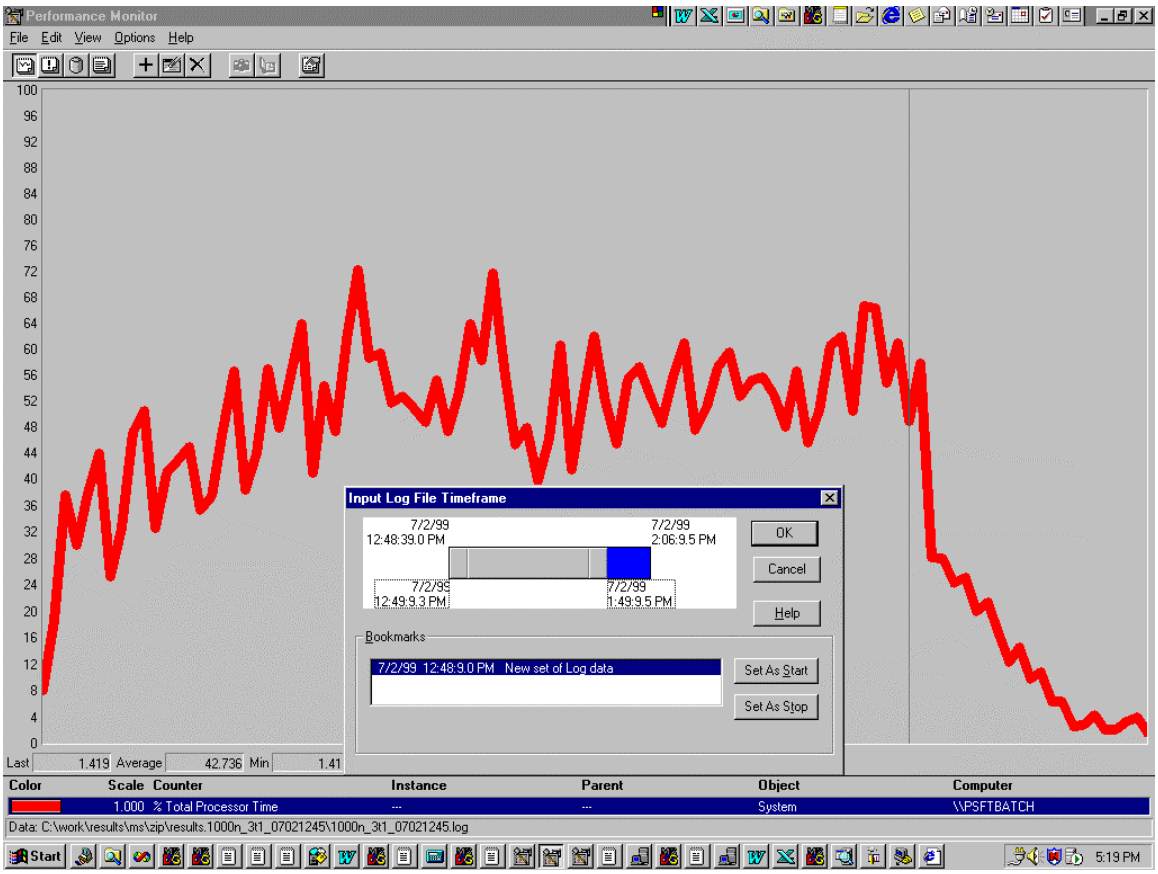


FIGURE 11: Changing the Time window for a performance monitor log

This facility is very useful when attempting to get average statistics using the report option because the “ramp-down” period will influence the average results in a way that is not useful.

Reporting is useful when characterizing steady state performance of some interval. It is also useful when trying to display a large number of counters and values. In chart mode it becomes very difficult very quickly to see more than a few counters. However, the performance monitor report view can do this in a nice fashion. To move into the report view select View→Report and then start to add counters using the Edit→Add to Report menu item.

Exporting a chart or report to a Text file

Once a behavior has been characterized (either as a chart or a report) and there is a need to show this to someone else, then either the entire log file can be sent or the chart/report can be exported to an ASCII file. To export one of these select File→Export Chart or File→ExportReport menu item.

Turning on NT disk counters

One important object in the Performance monitor is the Logical Disk. This provides such information as the “total number of disk transfers per sec” and the “avg. sec per disk transfer”. By default, however, these counters are not populated with any useful values. To enable these counters one must execute the diskperf program (with the –y option). For example:

```
diskperf –y
```

You must reboot the server for the disk performance statistics to be collected.

To check to see if the disk counters have been enabled just type in “`diskperf`” (as shown below) and the output of the command will indicate if the counters are enabled.

```
C:\>diskperf

Disk Performance counters on this system are currently set to never
start.

DISKPERF [-Y[E] | -N] [\\computername]

  -Y[E] Sets the system to start disk performance counters
        when the system is restarted.

     E  Enables the disk performance counters used for measuring
        performance of the physical drives in striped disk set
        when the system is restarted.
        Specify -Y without the E to restore the normal disk
        performance counters.

  -N    Sets the system disable disk performance counters
        when the system is restarted.

  \\computername        Is the name of the computer you want to
                       see or set disk performance counter use.

C:\>
C:\>
```

documentum

Measuring Performance on UNIX

In this section we briefly outline some of the most common tools available for measuring performance on UNIX. As with the performance monitor we recommend logging performance information to a file and analyzing it later.

How to Measure CPU utilization

There are several tools typically available on UNIX to measure CPU consumption. One of the most common tools is vmstat(1) which provides the total CPU consumption (over all processors). This is available for all Documentum supported UNIX operating systems: Solaris, HP/UX, and AIX. However, one unfortunate aspect of this commonality is that each operating system has a slightly different version of vmstat. A sample of this output for Solaris is shown below (the CPU values are circled on the far right):

```
procs     memory            page            disk          faults      cpu
 r b w   swap  free  si  so pi po fr de sr s1 s3 s4 s7   in   sy   cs us sy id
 0 0 0   6984 26024   0   0  9  7  7  0  0  0  0 44  0  759 11706 1897 8  5 87
 0 0 0 20451952 21160456 0 0 1  0  0  0  0  0  4  0  0  163  804  205  0  1 99
 0 0 0 20451680 21160336 0 0 0  0  0  0  0  0  0  0  0  139  340  150  0  0 100
 0 0 0 20451784 21160376 0 0 0  1  1  0  0  0  0  0  0  131  450  162  0  0 100
 0 0 0 20451504 21159856 0 0 0  0  0  0  0  0  4  0  0  161  554  163  0  1 99
 0 0 0 20451504 21159840 0 0 0  0  0  0  0  0  0  0  0  131  322  133  0  0 100
 0 0 0 20450880 21159528 0 0 0  0  0  0  0  0  0  0  0  132  457  167  0  0 100
 0 0 0 20450776 21159384 0 0 0  0  0  0  0  0  0  0  0  132  646  163  0  0 99
 0 1 0 20435080 21150528 0 0 583 0 0  0  0  0  4  0  0  292 7941  576  1  1 97
 0 1 0 20420144 21142168 0 0 1  0  0  0  0  0  0  0  0  565 2991 1004  0  1 99
```

This output is generated invoking vmstat with an iteration number (number of times you want it to sample) and an interval (the time between samples). Notice that on Solaris, the first sample should be disregarded as it shows the average values since system startup. The CPU portions of this report are at the far right hand side. This report provides user space CPU usage, system space (equivalent to NT “privileged time”) and idle time. The AIX version of this report is similar, but different:

```
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
  0  1 67748 928177   0   0   0   0    0   0 744 11640 703  3  2 44 51
 0  1 68371 927578   0   0   0   0    0   0 567 10578 347  6  1 75 18
 0  1 69564 926359   0   0   0   0    0   0 528 5377 258  3  1 45 51
 0  1 71824 924047   0   0   0   0    0   0 614 7214 431  5  1 39 55
 0  1 74913 920853   0   0   0   0    0   0 677 13410 558 11  2 23 65
 0  1 77712 917975   0   0   0   0    0   0 625 9215 468  5  2 23 70
 0  1 79384 916278   0   0   0   0    0   0 584 5051 390  6  1 50 42
 0  1 81186 914414   0   0   0   0    0   0 612 9911 437  7  2 60 32
```

Notice that on the AIX report there is a “wa” column. This indicates when the system was idle waiting for IO to complete. The solaris “id: (idle) column is equivalent to the AIX id + wa columns. The wait-for-IO column is useful to identify conditions where the application is waiting on disks that are not moving fast enough.

An alternative way to measure the total CPU is to use the sar –u (for Solaris, HP/UX) or sar –p (for AIX) command. The output from SAR appears as shown below (for Solaris):

```
SunOS docu14 5.6 Generic_105181-12 sun4u    03/13/99

23:51:08    %usr    %sys    %wio   %idle
23:51:18       0       1      33      66
23:51:28       0       0      50      50
23:51:38       0       0      55      44
23:51:48       0       1      47      52
23:51:58       0       0      38      62
23:52:08       0       0      53      46
23:52:18       0       0      55      44
23:52:28       1       1      67      30
```

Note that as with the AIX vmstat command the wait for IO value is provided. The main benefit of using SAR –U (-P on AIX) is that each value actually has a timestamp (hh:mm:sec) on the far left hand side. This is useful when trying to correlate the CPU values with other external events (like users experiencing poor response time).

All of these operating systems also provide tools to watch the consumption on each CPU (in a multi-CPU environment):
- Solaris: mpstat(1)
- HP/UX: sar -Mu
- AIX: sar –P

Each of these has a different format and as with NT, these are best used for finding a single threaded problem.

How to Measure Memory Consumption

The vmstat(1) tool is also the best way to measure memory consumption for these operating systems. An example output for HP/UX is shown below:

```
procs           memory                  page      faults                      cpu
    r     b     w      avm    free si   so  pi po fr de  sr  in     sy    cs  us sy id
    0     0     0    12486  640338  0    0   0  0  0  0   0 1253   6028  1242  10  3 87
    0     0     0    14583  638677  0    0   0  0  0  0   0 1107    230    69   0  0 99
    0     0     0    14583  638661  0    0   0  0  0  0   0 1106    117    54   0  0 100
    3     0     0    21362  636377  0    0   0  0  0  0   0 1418   6219   420  19  1 80
    2     0     0    28199  633562  0    0   0  0  0  0   0 1179   4513   249  20  1 79
    4     0     0    52778  625892  0    0   0  0  0  0   0 1209  10126   465  61  4 35
    4     0     0    76070  618137  0    0   0  0  0  0   0 1214  10733   488  61  3 35
    4     0     0    99026  610299  0    0   0  0  0  0   0 1220  10962   529  62  4 34
```

Some important fields on this report are:
- Avm - The number of Active virtual pages (8K per page)
- Free - The size of the free list (in pages)
- Si – processes swapped in
- So – processes swapped out
- Pi – pages faulted in
- Po – pages faulted out

On Solaris, as shown earlier, the important memory-related fields are:
- Swap – current amount of swap space in Kbytes
- Free – size of free list in Kbytes
- Pi – pages faulted in
- Po – pages faulted out

On AIX (shown earlier) the important memory –related fields are:
- Avm – average memory used
- Fre – free memory available

- Po – pages faulted out
- Pi – pages faulted in

As shown above, only Solaris provides swap (or page file) usage in vmstat. The other way to get this information is using the swap(1) program.

How to Measure disk activity

On Solaris and HP/UX disk activity is best measured using “sar –d”. A sample output from this command is shown below:

```
SunOS docu14 5.6 Generic_105181-12 sun4u    03/13/99

00:03:10   device        %busy   avque   r+w/s  blks/s  avwait  avserv

00:03:10   docu14:vold(pid831) 0     0.0       0       0     0.0     0.0
           c0t4d0             0     0.0       0       0     0.0     0.0
            c3t5d0            0     0.0       1      20     0.0     1.4
           c3t5d0s2           0     0.0       0       0     0.0     0.0
           c3t5d0s3           0     0.0       0       0     0.0     0.0
           c3t5d0s4           0     0.0       1      20     0.0     1.4
           c5t4d0             0     0.0       1       4     0.0     0.4
           c5t4d0s2           0     0.0       0       0     0.0     0.0
           c5t4d0s3           0     0.0       0       0     0.0     0.0
           c5t4d0s4           0     0.0       0       4     0.0     1.4
           c4t4d0             0     0.0       0       0     0.0     0.0
           c7t5d0            36     0.4      46     188     0.0     9.5
           c7t5d0s2           0     0.0       0       0     0.0     0.0
           c7t5d0s3           0     0.0       0       0     0.0     0.0
           c7t5d0s4          36     0.4      46     188     0.0     9.5
           c6t5d0             0     0.0       0       0     0.0     0.0
           c1t2d0            15     0.4      11     167     0.0    37.5
           c1t2d0s0           0     0.0       0       0     0.0     0.0
           c1t2d0s1           0     0.0       0       0     0.0     0.0
           c1t2d0s2           0     0.0       0       0     0.0     0.0
           c1t2d0s3          15     0.4      11     167     0.0    37.5
           c1t2d0s7           0     0.0       0       0     0.0     0.0
           c1t3d0             0     0.0       0       0     0.0     0.0
           c1t3d0s2           0     0.0       0       0     0.0     0.0
```

The %busy, r+w/s, blks/s, and avserv fields are the most important:

- %busy – the percentage of time the disk was busy servicing a request. This item is extremely useful in spotting an overwhelmed disk (or stripe set).

- r+w/s – the number of disk reads (& writes) that occurred on the drive per second

- blks/s – Each request could cover a number of blocks. This count covers the number of blocks per second processed.

- avserv – Average time (in milliseconds) to service each transfer request (includes seek, rotational latency, and data transfer times) for the device

An unfortunate aspect of the Solaris “sar –d” is that the names of the disks (far left hand side) do not by default match up with the names of the drives. In fact, in the sample above the actual output from sar –d did not generate names like c3t5d0s2 (which although not very user friendly do match up with the RM6 configuration software), rather ones like “sd20,c” which cannot be easily mapped to the names generated in the RM6 software.

To measure disk activity under AIX it is best to use the iostat(1) command. This command operates in a similar fashion to vmstat(1). A sample output from this command is shown below:

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk15          2.3      15.6       3.9        468         0
hdisk16          3.4      17.3       4.3        520         0
hdisk17          2.0      10.7       2.7        320         0
hdisk18          6.1      35.3       8.8        984        76
hdisk19          2.4      10.1       2.4        256        48
hdisk20          1.8      10.7       2.7        320         0
hdisk21          5.4      38.8       9.7       1164         0
hdisk22          1.5       7.2       1.8        216         0
hdisk23          2.5      27.7       5.2        548       284
hdisk24          4.4      25.6       6.4        752        16
hdisk26          4.7      25.1       6.3        752         0
hdisk27          3.2      17.5       4.4        516         8
hdisk28          1.6       7.6       1.9        228         0
hdisk29          2.6      13.9       3.5        416         0
hdisk30          2.1      11.3       2.8        340         0
hdisk31          2.6      15.1       3.8        452         0
hdisk32          1.2       9.6       2.4        288         0
hdisk33          3.1      18.1       4.5        544         0
```

These fields have the following meaning:

- %tm_act – Indicates the % of time the disk was active.
- Kbps – indicates the amount of data transferred to the drive (read or written) in KB per sec.
- Tps – indicates the number of transfers per second that were issued to the drive. A transfer is an I/O request to the physical drive. Multiple logical requests can be combined into a single I/O request.
- Kb_read – The total number of KB read
- Kb_wrtn – The total number of KB written

How to Measure Network Activity

The tool for measuring network activity is nearly identical for all three operating systems: netstat(1). In fact, Windows NT even has a netstat program. The output of the netstat program is as follows:

```
input   (lo0)      output           input   (Total)    output
 packets  errs  packets  errs colls  packets  errs  packets  errs colls
     129     0      137     0     0  1884364     0  1844167     0     0
       0     0        0     0     0    10262     0     9869     0     0
       0     0        0     0     0     9179     0     8817     0     0
       0     0        0     0     0    10093     0     9692     0     0
       0     0        0     0     0    10757     0    10361     0     0
```

As with the previous commands a time interval can be specified to take samples over a test period. The program periodically reprints the header and when it does it also prints the cumulative totals for each counter since boot time. In this sample above (30 sec intervals) there were nearly 10,000 packets sent and received each 30 seconds. The downside of netstat is that this sampling

report does not provide the number of bytes transferred via the network card or a date to correlate the data with other events.

Netstat has many different options. To produce the output above one can run:

```
netstat -I lo0 30
```

The –I option will sample the packet statistics for the interface board every N seconds (in the case above, every 30 seconds). Unfortunately, each operating system will name the interfaces differently. The best way to obtain the interface board name is to run “netstat –r” (as shown below):

```
Routing Table:
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
localhost            localhost             UH        0  19624  lo0
172.16.0.0           beefcake.documentum.com  U         3   1197  hme0
```

Netstat can also be used to display the currently active tcp connections.

How to Measure Per-Process CPU consumption in UNIX

There are two “off-the-shelf” ways to measure per process CPU consumption on UNIX. Probably the most common way is to run the “ps” command and look at the CPU consumption for long running processes. For example the following is an excerpt from a “ps –ef” on Solaris:

```
  oracle 18116     1  0 21:35:06 ?        0:06 ora_lgwr_EDMI
  oracle 18118     1  0 21:35:06 ?        0:00 ora_ckpt_EDMI
    root 23779 23778  0 23:15:05 pts/5    0:00 sleep 30
    root 23753 19235  0 23:14:05 pts/5    0:00 ksh
    root 19233   609  0 15:57:34 ?        0:00 in.telnetd
 dmadmin 18523 18125  0 22:57:03 pts/0    0:24 ./documentum -docbase_name EDMI -
security acl -init_file /opt1/documentum/dba/c
   bench 19235 19233  0 15:57:34 pts/5    0:01 -ksh
    root 23784 23753  0 23:15:07 pts/5    0:00 /usr/bin/ksh ../kill_measure
 dmadmin 18125     1  0 21:35:41 pts/0    0:12 ./documentum -docbase_name EDMI -
security acl -init_file /opt1/documentum/dba/c
netscape 26668     1  0 17:08:36 ?        0:09 /opt1/documentum/RightSite_81/pro
duct/bin/dwvsm
    root 23778     1  0 23:15:05 pts/5    0:00 sh -c sleep 30
   bench 19164 19162  0 15:54:36 pts/0    0:00 -ksh
  oracle 18138     1  0 21:36:45 ?        0:01 oracleEDMI (LOCAL=NO)
  oracle 18136     1  0 21:36:45 ?        0:01 oracleEDMI (LOCAL=NO)
    root 19162   609  0 15:54:36 ?        0:00 in.telnetd
    root 23783     1  0 23:15:06 pts/5    0:00 /usr/bin/sleep 120
```

CPU seconds consumped since start of process

Process name + arguments

Given the ps output it is fairly easy to grep through this for the component (e.g., ora_, or documentum) and sum the total. The downsides to this approach are that:

- It is difficult to compute sums of times in MM:SS format.
- The resolution is 1 second,
- If a process starts and terminates between the sampling period (suppose one runs ps twice with 5 minutes in between) then the information the execution of that process is not available (it started after the last ps and terminated before the current ps call).

That is, this approach works best for a few long running processes, not many short lived ones.

Another approach is to use the standard UNIX accounting mechanism. This mechanism will track the performance of processes that terminate. Hence, it works well for many, short lived processes. When a process terminates its information is written to a p-accounting file. This can be viewed later in a report. To enable the accounting one can run:

```
rm filename
/usr/lib/acct/accton filename
```

where *filename* is the accounting file. It is typically a good idea to put this file on a set of drives that are not heavily used and/or are not limited to a single spindle. To turn off the account execute:

```
/usr/lib/acct/accton
/usr/lib/acct/acctcms -a filename > output_report_name
```

This will generate a report that looks as follows:

```
TOTAL COMMAND SUMMARY
COMMAND    NUMBER      TOTAL      TOTAL     TOTAL    MEAN   MEAN   HOG  CHARS     BLOCKS
NAME         CMDS    KCOREMIN   CPU-MIN  REAL-MIN  SIZE-K CPU-MIN FACTOR
  TRNSFD      READ

TOTALS       5292 25220232.43    110.28  14329 228683    0.02   0.01   331 7542208    11419
oracle        797 24226566.00     20.18   6147 1200632    0.03   0.00    13 05899520        0
dwserver      168   766076.12     68.64   1358 11160    0.41   0.05    583 412672        998
document      621    91358.25      8.56   3303 10675    0.01   0.00    317 862336       1896
ns-httpd        3    75760.98      3.73    461 20292    1.24   0.01    604 348928       1784
vxva.sl         1    52122.96      6.75    497 7717    6.75   0.01   3507 48672          15
proctool        1     3645.44      0.74    674 4949    0.74   0.00    801 50528           9
dmbasic       163     1681.92      0.44   0.70 3837    0.00   0.62      9
```

The most important column in this report is the “TOTAL CPU-MIN” one which indicates the number of CPU minutes that all processes with the “COMMAND NAME” consume. The report above is for a workload that ran on a host-based system (all server software running on the same host). The usage report covers the Netscape http server (ns-http), the Documentum RightSite server (dwserver), the Documentum DocPage server (document), and Oracle.

Note that this is only for processes that have terminated. If a process has not terminated, its information will not show up in this report. The following are the long running processes of each of these servers that might not get accounted for in this manner (because they were still running when the accounting summary report was run):

- DocPage Server: the root process that spawns processes as new sessions are established.
- Oracle: the Oracle processes – PMON, CHKPT, SMON, DBWR, etc.
- Rightsite: the dwvsm (Rightsite root process)
- Netscape: Normally this CPU won’t be accounted because the http server rarely terminates.

Memory and Swap Space

This section will discuss memory and swap space concepts. It will also cover pointers on how to configure both UNIX and NT to support large memory. It will help the reader figure out how to detect memory/swap space exhaustion. It will also go over how you can figure out how much memory is being consumed by the RDBMS and the Documentum servers (for UNIX and NT). That is, it will cover the NT performance monitor counters for memory and the UNIX tools (swap and vmstat).

Background on Virtual & Physical Memory

Before we move into the memory sizing discussion it is good for us to review the concepts of virtual memory and physical memory and how these relate to sizing memory. Firstly, virtual memory is a concept or service provided by the operating system (and hardware) that allows each process to operate as if it had exclusive access to all physical memory. All processes active at the same time have this abstraction. It turns out, however, that a process only needs a small amount of this memory to perform its activities. This small amount, called the *process working set*, is actually kept in memory. The operating system will manage sharing physical memory among the various working sets.

Unfortunately physical memory is a limited resource. When the operating system wants to conserve physical memory or manage the fact that all of the working sets won't fit into physical memory, it moves the excess into a paging file or swap file. In addition, although a process may think it has exclusive access to all of its virtual memory the operating system will transparently share many of the read-only portions (i.e., program instructions).

Therefore, sizing memory on a server system is essentially figuring out how much the process working sets are going to consume. These concepts are illustrated below.

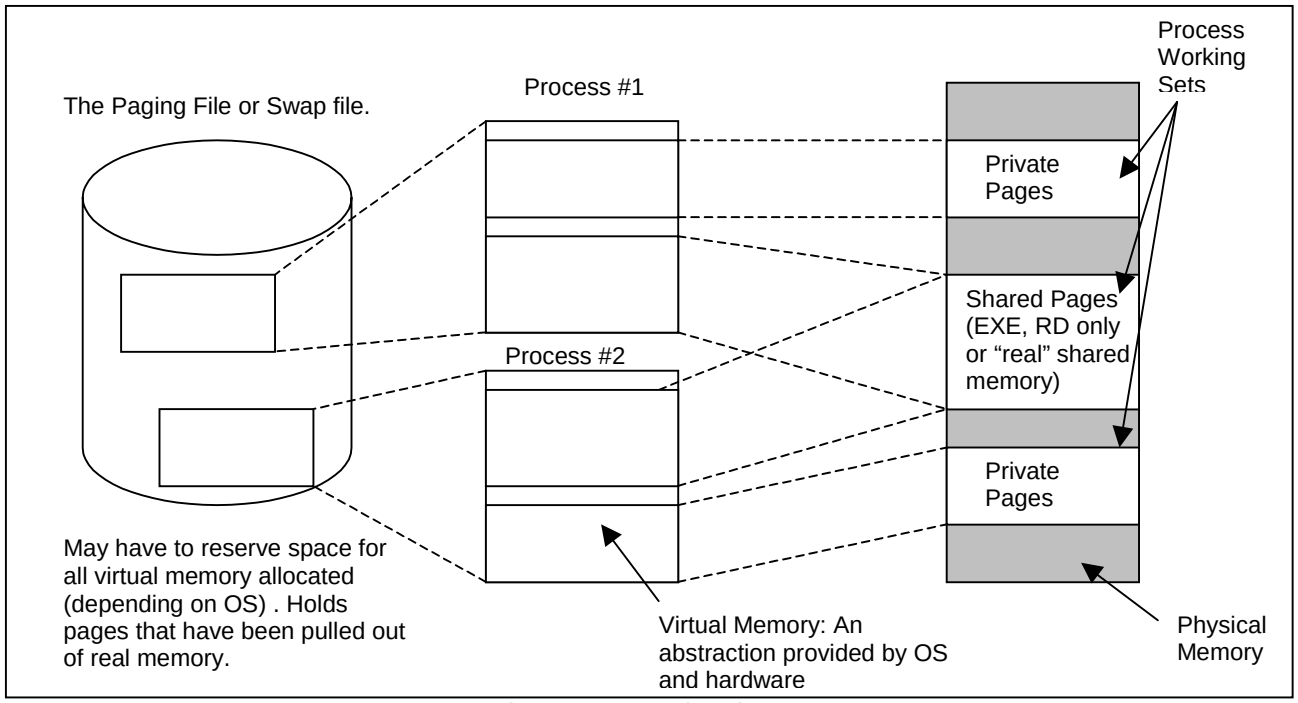


FIGURE 12: Real Memory vs. Virtual Memory

In memory sizing there are two things to configure: Physical memory and Paging file space. To size physical memory you only consider working sets. Aside from the DBMS you should be primarily concerned with the non-shared working set pages of a process. To size the paging file you consider the virtual memory allocated. If you run out of either, problems could arise. If memory is exceeded, then lots of I/O could occur to the paging/swap file leading for poor performance. If the space in the paging/swap areas are exceeded then commands may fail. Although some Operating Systems will clearly distinguish between the two, they typically give out messages that are confusing and vague.

On a typical installation of the Documentum server software (which might include the RightSite, Docpage server, and a DBMS server) the real memory usage is quite likely to be dominated by various caches. A cache is a memory area used to hold some object so that the software won't have to perform some expensive operation (read data from disk, from network, etc). The size of some of these caches can be controlled by the administrator, but others are automatically sized. However, as a particular process grows, it is typically filling out its caches with information to make its operations less expensive. Therefore, excessive memory usage is not always a "bad" thing. Typically it is allowing other operations to be more efficient. The most important sizing task is to ensure that the cache needs do not outstrip available physical memory.

DBMS caches

The most dominant cache is likely to be the DBMS data cache. Luckily, its size is under administrative control. The DBMS uses the data cache to minimize the disk I/Os that it has to perform for data/index rows. It is significantly less expensive for a DBMS to find 100 rows in its data cache than to find it on disk. A production server system with many documents will likely need hundreds of Mbytes, perhaps even one or more Gbytes for this cache to ensure acceptable performance. Sizing in such a fashion will reduce disk I/Os significantly. (This is covered in more detail later.)

Several DBMS's also have caches for SQL statements that are repeatedly executed. These caches conserve on the number of CPU cycles needed to do things like security validations, parsing, and execution plan preparation. It is typically good to give these caches plenty of memory. Check with RDBMS documentation for more details.

DocPage Server caches

The Docpage server has several caches used to conserve on operations like DBMS interactions, CPU cycles, and network operations. Most of the caches are small (less than 1M byte) and bounded by the number of objects. By far the most dominant memory usage by the DocPage server lies with its "global type cache". This cache holds structures used to enable fast access to the DBMS tables that make up a type's instances. The size of this cache is limited by the number of types in the system. The amount of real memory consumed is determined by the number of instances/types that are accessed. Although this cache is called the "global" cache, it primarily functions as a per-session access structure. Each Docpage process/thread will have its own copy.

RightSite Server Caches and Work Areas

The memory usage of RightSite is dominated by three memory areas:

- DMCL object cache
- Docbasic Compiled Code, and
- Temporary Intermediate Dynamic HTML construction memory.

The DMCL object cache contains memory used to hold recently referenced Docbase objects (the size of this is controlled via the dmcl.ini file). The Docbase compiled code contains the precompiled Docbasic code for the dynamic HTML executed by RightSite. It will be at most as large as the on-disk cache of pre-compiled Docbasic routines and its size is controlled by the DW_MAX_CACHE_SIZE RightSite environment variable. Finally, the temporary, intermediate Dynamic HTML construction memory is the area used by RightSite to construct a dynamic HTML page. The actual "used" portions of this memory will vary over time, however, once such a page is used it will be "in" real memory until paged out or until the process terminates. All of these areas can grow Mbytes in size depending on the workload.

Memory Shortfall on Windows NT

If the Swap space (or paging file) has been setup properly on NT then the primary indication of a lack of memory will be poor response time or throughput. The poor performance is caused by the OS having to page out (and in) the working sets of the processes that need memory. In such a case there are several NT performance monitor counters which are useful:

- Available memory (measured in bytes)
- Pages out (or in) per sec
- Working set sizes for the DBMS and for the DocPage Server.

An example of how these counters might interact is shown below. In this example, as the Docpage server grows in size (as more users log in) available memory approaches 0 bytes. As soon as it does, the system starts to page out the two processes in an effort to trim memory.

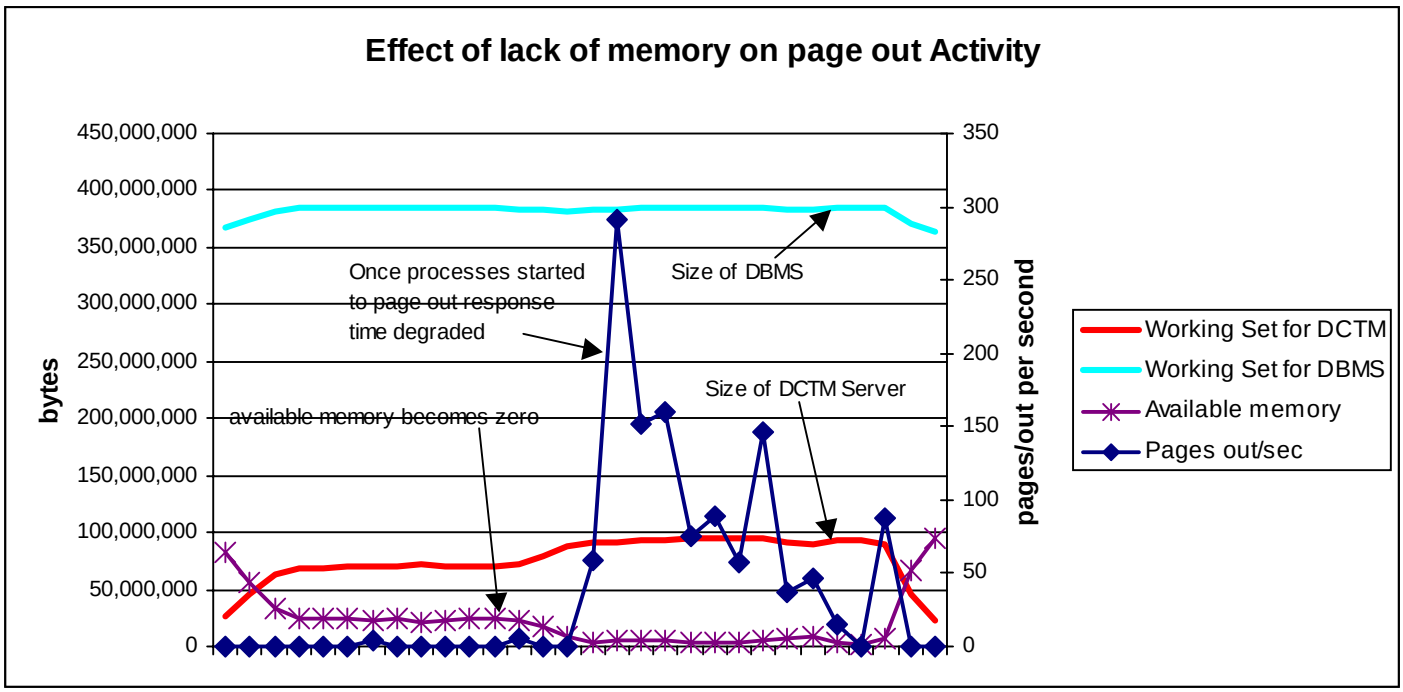


FIGURE 13: Illustration of Memory shortfall in NT (From NT Perfmon Counters)

Memory Shortfall on UNIX

On UNIX the above behavior is very similar and for the most part can be viewed using vmstat. The “free” memory column in the report will get very close to 0 and when this occurs there will be a large amount of activity on the pi (page in) and po (page out) columns.

Swap Space (or Paging File Space) shortfall on Windows NT

Insufficient swap space (or paging file space) can show up as bad performance, but it can also show up as problems like the inability to start new connections. Typically NT will create an error “popup” that indicates there is “Insufficient Virtual Memory”. This really means that there is no space left in the paging file for additional processes.

A shortfall in the paging file space on NT can be detected using the Paging File object counters of the Process object. The graph below shows a system that was making heavy use of memory supporting 500 users/hour that were accessing the Docbase through RightSite. The system had a total of 4G of memory and a 4G paging file. In this test the paging file usage peaked at 80%. The DBMS usage was constant (most of it due to the buffer cache). The DocPage Server usage was fairly small (less than 200M for all active users), and the RightSite multi-process architecture consumed the rest as users were added to the test. In this test 75% of the 500 users served in the hour were active at any one time.

This graph was constructed from an NT performance monitor chart that was exported to Excel. Note that the Process object allows us to see the page file byte usage for each of the processes. The NT performance monitor does not handle the multiple RightSite processes very well (since they all have the same name [dwserver.exe]) hence we inferred the RightSite usage from the _total instance in the Process object.

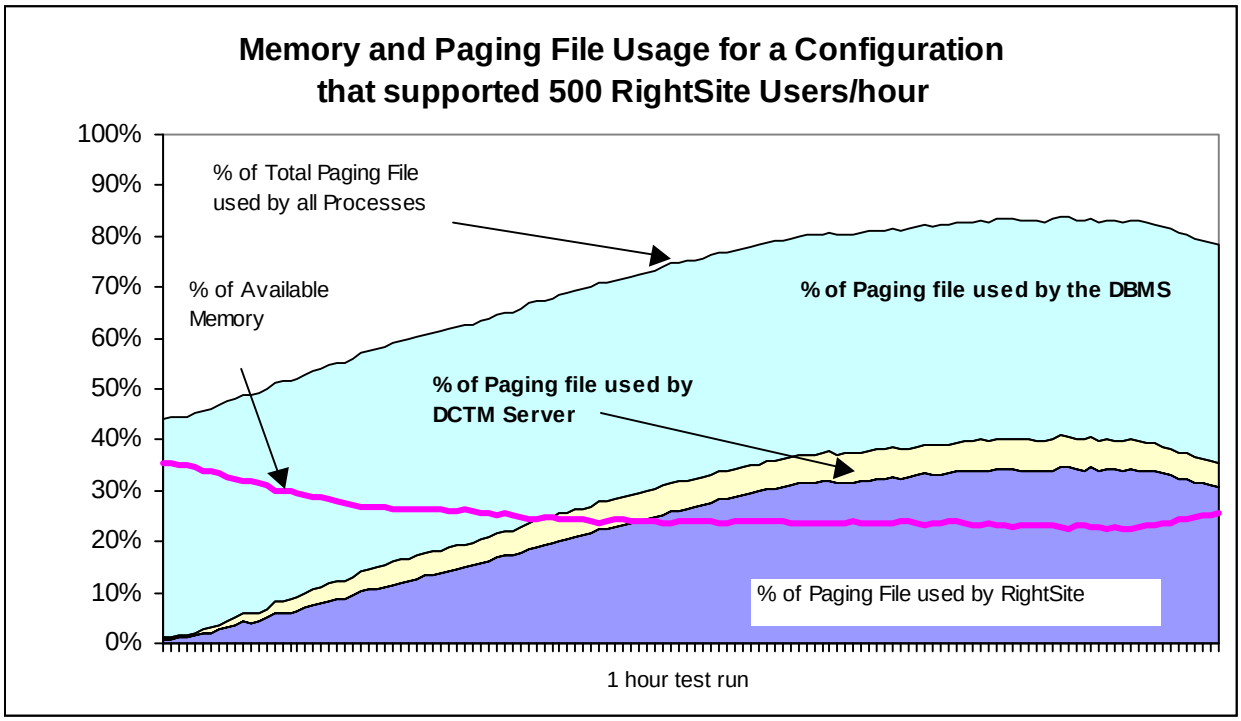


FIGURE 14: Memory and Paging file usage example

Swap Space (or Paging File) shortfall on UNIX

On UNIX a paging file shortage may look very much like a memory shortfall. Miscellaneous commands will fail with an “out of memory” error. This will happen even though there may be plenty of memory available. The two best ways to determine if a paging file shortfall is occurring are to use the “swap” command or the “vmstat” command.

Enabling Asynchronous I/O

I/O to a disk can either be synchronous or asynchronous. Synchronous disk I/O implies issuing a request and waiting for a response as to its completion. Asynchronous disk I/O's allow the caller to issue many requests and to perform other actions while these I/Os are pending. Operating systems typically simulate asynchronous I/O by copying the data into the Operating system buffer cache and writing it out on the side.

All RDBMS can benefit from asynchronous I/O. For some versions of UNIX it is necessary to enable asynchronous I/O explicitly. Check your UNIX and RDBMS manuals to gather more details on how to enable this feature.

Do You Have Enough Drives?

This section will cover the concepts of drive space vs. drive access time. It will describe RAID and some of its benefits and limitations. It will work through some examples and show how disks can affect performance of the Docpage server & RDBMS. It will cover more detail on UNIX and NT on how to detect that your disks have become a bottleneck.

Disk Striping

One important concept on disk sizing is "disk striping". This is essentially taking a piece of data and physically placing parts of it on separate disks. When the piece of data is requested all of the drives participate in returning the data. This might seem at first glance to be a technique that would take longer than just getting the data from a single place (it certainly requires more processing). However, given that the unit's are typically large (many Kbytes) its unlikely that the single disk could have gotten the data in a single operation anyway. It would have taken multiple operations. By striping the data its possible to get the drives to work in parallel, thus allowing the operation to happen much faster. This is illustrated below. The "striping" logic of an OS (or disk array firmware) makes a group of disks appear as a single disk or volume. It will break up a disk request into multiple disk requests across multiple drives.

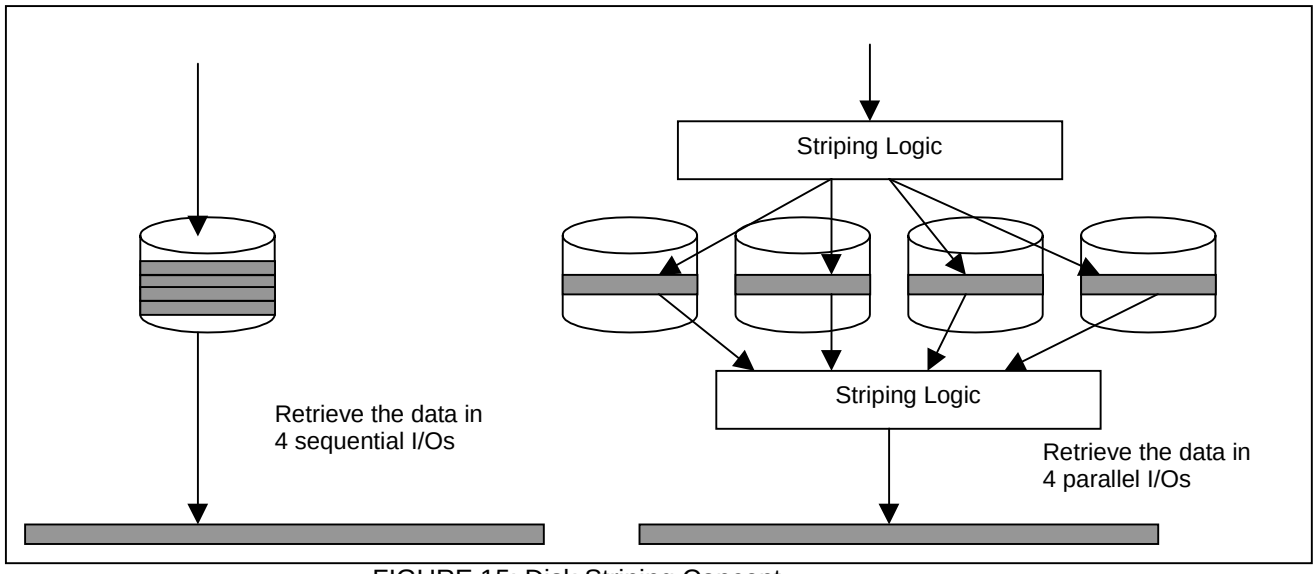


FIGURE 15: Disk Striping Concept

One key component of the disk striping performance is the size of the data stripe (data block). The smaller this block size is the more parallel drives that might be able to used for a single I/O. The more parallel drives, the better performance. This is true up to a point, however, because if the size is too small the overhead of dealing with the stripe exceeds any performance gains from the striping. Also, if the stripe is too big then I/O's are likely to queue up on a single drive. An extreme example of a poor stripe size would be if an administrator had many individual disks and striped the data via Oracle by creating multiple table space files across the independent disks. That is putting a single, large, sequential portion of the table space on each drive. Once a request is made for a portion of the table, it is likely that the I/O's are going to be concentrated upon a single drive in an uneven fashion. Hence, in general, RAID0 (striping without parity) as described above, will outperform tablespace or DBMS device level disk striping.

However, RAID0 has some disadvantages. If a single drive fails then the entire stripe set fails. That is, four disks in a RAID 0 stripe set (or logical drive) have a shorter Mean Time Before Failure (MTBF) than a single drive. This is because any one of the 4 physical drives can bring the logical drive down.

There are two major ways to protect performance, yet maintain reliability: mirroring and striping with parity. With mirroring (or RAID 1) the data is written to two drives. If one drive fails then the data can be read from the other. When data is striped over a set of mirrored drive pairs this is termed RAID1+0 or RAID10. In the other technique, 'striping with parity' (or RAID5), parity information is written out in addition to the data. This parity information can be used to recreate data if a drive fails. The parity information is always written to a drive that does not contain the data that was used to create the parity code and in addition, is spread over the drives for better reliability.

The downside for RAID1+0 is the cost of the additional drives. The downside for RAID5 is the extra I/Os needed to write out the parity information. In general, we've found that this access penalty for RAID5 is fairly severe for DBMS files. It can provide decent performance for Docbase content.

Disk Space vs. Disk Response Time

There are two disk sizing considerations in a Documentum installation: the space required for the data and the response capacity. In many respects they are almost independent items from each other. Engineering the disk space entails ensuring that there is sufficient room to store the data (or temp data). Providing sufficient access capacity revolves around ensuring that there are sufficient disk spindles (or arms) to gather all of the data in a reasonable timeframe.

The following example illustrates the difference. Suppose that there is some DBMS table that is 4 GB in size. Clearly this table could fit on a single 9 GB drive. However, suppose that the DBMS needed to scan the entire table. If it used 16K blocks then 262,000 disk I/Os would be needed to scan the table. If this scan took place over a single disk drive (at 10 msec an I/O) then it might take 44 minutes to scan the table (assuming that no pre-fetching occurred). However, if the table was striped over 10 drives in a RAID 0 stripe then it might actually only take 4 minutes to scan the table if one read was able to hit all 10 drives (by small stripe units). Notice that ten 9 GB drives offer 10 times the space of that was sufficient for the table, however, meeting the space requirements did not necessarily meet the response time requirements.

Detecting Disk Space Problems

We only briefly cover detecting disk space problems. Avoiding them is a capacity planning issue and covered elsewhere.[1] On Windows NT disk free space is reported by command line tools like "DIR" or through the GUI file management tools like the Windows Explorer or the File Manager (WINFILE.EXE).

On UNIX the disk free space can be displayed using commands like "`df -k`". This option displays the disk space in Kbyte units available in each file system.

Detecting Disk Response Problems

There is a general rule of thumb that disks can service I/Os with a service time of 10 to 15 msec. With certain types of hardware disk controller caches writes can be reduced to 4 msecs. One important thing to detect (especially when using RAID) is how close to these numbers your access time is when servicing a large number of I/Os.

Poor Response times on Windows NT

The Logical disk object/counters outlined earlier in Table 2 are the best way to detect disk access problems on Windows NT. For example, the following three counter values were obtained from 3 different sources:

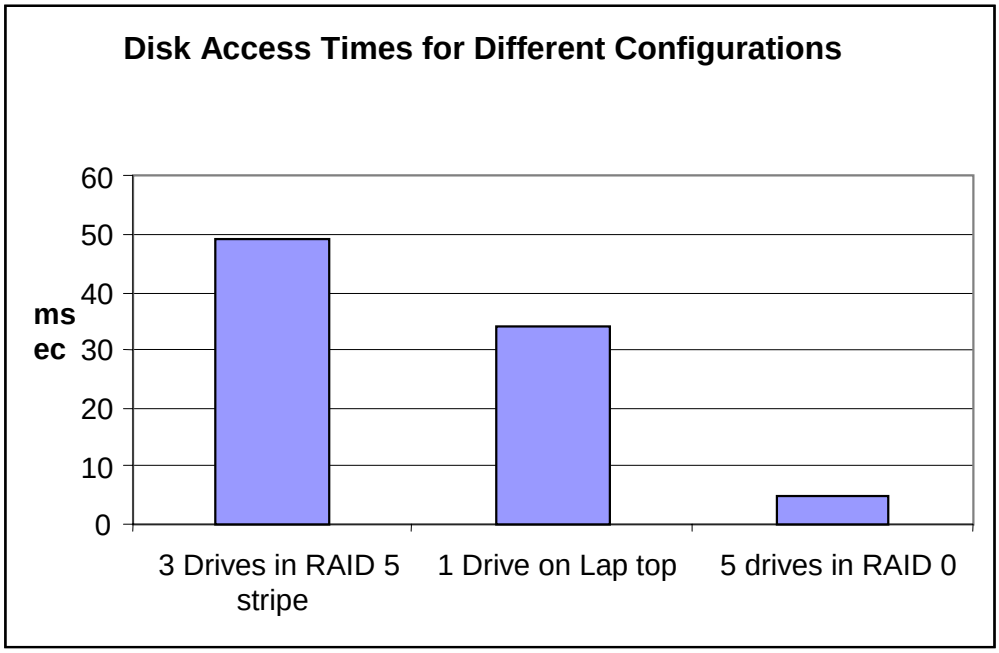


FIGURE 16: From "Avg Secs per disk Transfer" of NT Perf Mon

The important point about the access times shown above is the implication for how many I/Os each configuration can accomplish. For example, based on the graph above each of the configurations could conceptually accomplish the following number of I/Os:

[1] "*System Sizing Guide, version 2*", Documentum

Configuration	I/Os per second
3 Drives in RAID 5 stripe	20
1 Drive on Laptop	30
5 drives in RAID 0	200

TABLE 3: Implied maximum throughputs from the Drive example above

That is, by configuring RAID 5 on three drives on a large server machine it is possible to reduce the capacity of 3 drives to be less than a single drive on a slow laptop (the data from which this was taken consisted of only 30% writes). RAID 0 is typically not an option in a production environment, however, RAID 1+0 (mirroring then striping) can be.

Poor Response Times on UNIX

The behavior outline on Windows NT above can also happen on UNIX. The penalty of RAID 5 can be unexpectedly large. As mentioned before using "SAR –d" (HP/UX and Solaris) or iostat (AIX) you can get not only the number of disk I/Os, but the response time. The following example illustrates the possible performance penalty that could arise on RAID 5 (when used for the DBMS files):

Configuration	Response time at a rate of 50 I/Os per second	Implied number of I/Os per second
RAID 5 Logical Unit with 10 drives	56 msec	17
RAID 1+0 logical Unit with 8 drives	13 msec	76

TABLE 4: An example comparison of RAID based on UNIX measuring tools

We strongly recommend avoiding RAID5 for RDBMS files, devices, and temporary file caches of Documentum and RightSite.

documentum

Relational Database

The following section covers diagnostic and tuning issues for Relational Database Management Systems (RDBMS) when used with Documentum. The following concepts are discussed:

- Sizing RDBMS data caches,
- Sizing database tables, files, and devices,
- Generating query plans for slow, problematic queries, and
- Several vendor specific issues (e.g., Sybase row level locking, Oracle BEQUEATH)

For generic database issues we attempt to cover how to accomplish or diagnose the issue relative to Oracle, Sybase, and Microsoft. These notes are meant to summarize how to accomplish the tasks, rather than providing all of the necessary information to use the tool. Please refer to the documentation for each RDBMS vendor for complete details on their commands and usages.

Is that Cache BIG Enough?

This section will cover the concept of RDBMS caches. The important concepts to understand are the relationship of RDBMS caches to disk I/O and the relationship of docbase size to cache space needs. This section will also summarize how to determine the cache sizes for each relational database and how to determine if the data caches for these are getting decent hit rates.

As mentioned in the previous section, in a typical installation of the Documentum server software (which would include, say the RightSite, Docpage server, and a DBMS server) the real memory usage is quite likely to be dominated by various caches. A cache is a memory area used to hold some object so that the software won't have to perform an expensive operation (read data from disk, from network, etc. The most dominant cache is likely to be the DBMS data cache. Luckily, its size is under administrative control. The DBMS uses the data cache to minimize the disk I/Os that it has to perform for data/index rows. It is significantly less expensive for a DBMS to find 100 rows in its data cache than to find it on disk. A production server system with many documents will likely need hundreds of Mbytes, perhaps even one or more Gbytes for this cache to ensure acceptable performance. Sizing in such a fashion will reduce disk I/Os significantly. (This is covered in more detail later.)

Several DBMS's also have caches for SQL statements that are repeatedly executed. These caches conserve on the number of CPU cycles needed to do things like security validations, parsing, and execution plan preparation. It is typically good to give these caches plenty of memory. See the Documentum Sizing guide for more details on estimating how much memory is recommended for the RDBMS in general.

In the following sections we provide a few examples and tips to detect cache memory issues across various database vendors and Operating systems.

Oracle on Windows NT

The most important counters to view with respect to the Oracle data cache when running on Windows NT are:

Oracle7 (% physreads/gets) – This is the percentage of time a read to disk was required to fulfill a buffer access request. It is the data cache buffer miss ratio.

Logical Disk (Transfers/sec) – This is the throughput of disk I/Os to a partcular logical drive. The logical drives of interest are those used to store the DBMS data.

Logical Disk (Avg Sec per Disk Req) – As mentioned previously this will give a sense of the response time of the logical drive.

Memory (Available Memory) – Indicates how much memory is not being used by the system.

Process (Working Set) – for the Oracle instance of this Process object this counter will indicate how much memory Oracle is using for its work.

A (real) example of these counters in action is as follows. A single processor NT system was hosting both the Docpage server and the Oracle database. Performance was poor (slow response on various commands). Logs from the performance monitor indicated that during the periods of slow response our five counters of interest looked as follows:

Counters	Values
% physreads/gets	13
Transfers per sec	60
Avg sec per Disk Req (msec)	34
Available mem	175M bytes
Oracle working set	16 M bytes

These periods of “slowness” were identified by narrowing in on when the disk I/Os per second reached their peak value. The report values were restricted to those periods (using the Time Window to narrow a Performance Monitor log result set, as shown earlier).

These counters clearly showed that the Oracle data cache was sized too small. First, the buffer hit ratio was only 87%, when for a good system it should be around 95 to 100% (if possible). The disk response time was 34 msec per I/O and the disk throughput was 60 I/Os per second. The disks were unlikely to perform much more work. Most telling, however, was that there was 175M bytes of memory not being used on the system and the Oracle process was only consuming 16M bytes of memory. The system had sufficient of memory for the docbase (it was 20,000+ documents in size), however Oracle was not getting a chance to use it.

Oracle was not purposely sized to be small, rather, the initial database defaults for the data cache size were not changed after Oracle was installed. The defaults are very small. Once more memory was given to Oracle the performance improved significantly.

Oracle on UNIX

To obtain the % physreads/gets value on UNIX it is necessary to use the Oracle utlbstat.sql and utlestat.sql utilities (typically called Bstat/Estat scripts). The utlbstat.sql script will enable certain counters at the start of a measurement period. The utlestat.sql will turn off this sampling and print out a report for viewing. These scripts can be executed using the sqlplus utility. Once the report has been generated obtain the “per second” values for “db block gets” and “physical reads”.

Sybase

The Sybase buffer cache utilization is best found using the sp_sysmon stored procedure. This stored procedure is extensively documented in the Sybase Performance and Tuning guide. This can be invoked in the following fashion from within isql:

```
sp_sysmon “00:05:00”
go
```

Which will sample the counters for 5 minutes and display a report once complete. The following is a sample excerpt from this report on the buffer cache area:

```
Data Cache Management
---------------------

  Cache Statistics Summary (All Caches)
  -------------------------------------
                                per sec      per xact       count  % of total
                           ------------  ------------  ----------  ----------

    Cache Search Summary
      Total Cache Hits          70401.8       13159.2     4224108      97.3 %
      Total Cache Misses         1921.5         359.2      115291       2.7 %
  -------------------------  ------------  ------------  ----------
    Total Cache Searches        72323.3       13518.4     4339399
```

The above example shows that the disk buffer cache hit ratio was 97% (quite good).

This sp_sysmon data will also contain information on Sybase CPU consumption and disk I/O activity.

Microsoft

The Microsoft SQL Server 7 counters for data cache utilization (in the NT performance monitor) include:

SQLServer:Buffer Manager (Buffer Cache Hit Ratio) and
SQLServer:Buffer Manager (Cache Size [in pages]).

The Microsoft SQL Server 7 manages its memory dynamically. It attempts to consume all free memory on the system, but will cut back its memory usage as other applications consume more. It is possible allocate a fixed amount of memory to SQL Server. This is accomplished by changing the properties of the server from within the Start→Programs→SQL server 7.0→Enterprise Manager.
The SQL Server property sheets include a “memory” sheet that allows the Administrator to fix the memory size.

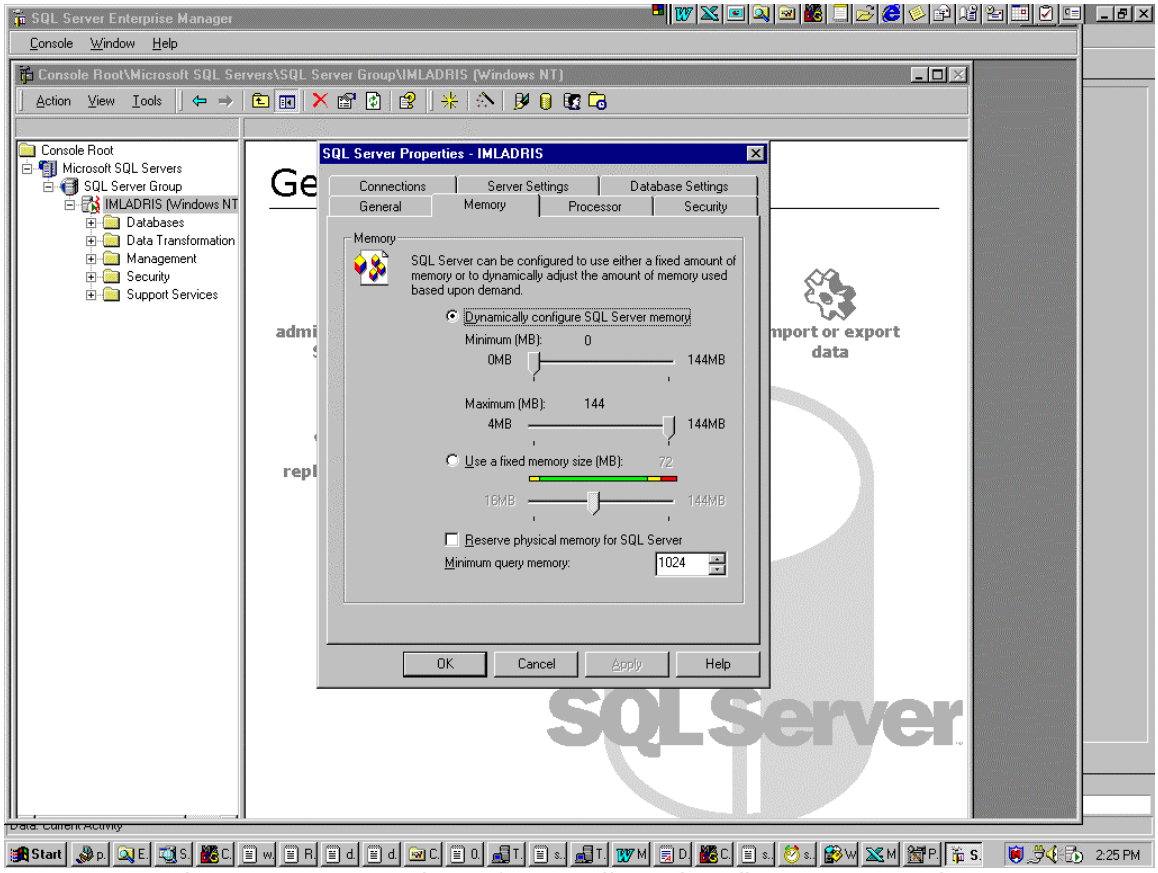


Figure 17: Property Sheet of SQL Se\\rver that allows memory to be set

Do I have enough space for my Database files?

In a typical Documentum installation a few tables will utilize the majority of the RDBMS allocated. One goal for sizing the database files is to ensure that there is sufficient space for these large tables as they grow. There are a few other considerations for each of the RDBMS vendors that we will address below.

Generally, in small Docbases require little attention to RDBMS space utilization. This typically only becomes an issue when the Docbase becomes large (hundreds of thousand to millions of objects/documents).

Oracle

Oracle allocates space based on extents, the sizes of which can be controlled by the administrator. It is typically advised to minimize the number of extents that a table consumes. Having too many extents can contribute to poor access times for that table.

Prior to 4i, it has only been possible to change the extent allocation behavior by modifying the default extent definitions for the Oracle tablespace that Documentum uses. This is because the tables are created behind the scenes by the DocPage Server. Since all tables are created in the same table space, this could lead to either a table with too many extents or lots of unused disk space as some of the small tables had large extents.

For example, suppose we have a Docbase with 1,000,000+ documents (all of type dm_document). This implies, for example, that the dm_sysobject_s table (table for the single attributes of dm_sysobject) will have 1,000,000+ rows. Suppose this table had an average row size of 350 bytes. Then this table will consume at least 350M bytes of disk space (not including its indexes). One strategy for allocating extents for this table would be try to limit its extent allocations to 10 extents. If these were equal size extents then each would be 35M bytes. Now a 35M byte extent is ok for dm_sysobject_s, but for a table like dm_format_s (which might have only 200 rows at 100 bytes a piece) a single extent of 35M would cause a significant amount of wasted space (nearly 35M bytes). On the other hand, if the default extent size was set to 1M byte (to limit the space wasted on the small tables) then the dm_sysobject_s table would need 350 extents!

In Documentum versions prior to 4i the best strategy to handle this extent dilemma was to size the first extent rather small, and have subsequent extents have large allocations. For example, the DBA might execute the following statement to modify the default storage parameters for a tablespace that will contain the objects for a very large docbase:

```
ALTER TABLESPACE DM_SAMPLE_docbase (
      initial           10M
      next              100M
      pcincrease        0
      minextents        1
      maxextents        120
) ;
```

Hence the first allocation for each table in this case would be 10M bytes, and all subsequent extent allocations (that would occur when the current extent is filled) would be at 100M bytes.

Another approach that is often taken involves analyzing the tables and indexes in an existing docbase, and rebuilding the objects using export/import. The Oracle import utility allows you to generate a script containing the SQL used to rebuild the objects. The DBA will edit the storage

parameters in this script, run it to pre-create the objects, and then import the rows. However, this can be quite time-consuming. It is much better to plan ahead for storage requirements, then to try to correct the problems after the fact.

With Documentum 4i it is possible to customize the storage space in a more complete fashion. The server.ini file can be modified to specify extent sizes and storage areas for types.

Firstly, system types are grouped into one of three categories:

- Large extents: (default: initial size 100K and next extent size 1M)
- Default extent size: (default: initial size 20K and next extent size 100K)
- Small extent size: (default: initial size 10K and next extent size 50K)

Types in the large extent class include dm_sysobject and some (but not all) of its subtypes. Types in the small extent class include items like dm_location. The complete list of system types to storage category is provided in the Documentum EDM Server installation guide. The sizes of each extent class can be changed in the server.ini. For example, the following overrides the default sizes:

```
[FUNCTION_EXTENT_SIZES]
database_ini_ext_large=10M
database_ini_ext_small=1M
database_ini_ext_default=1M
database_next_ext_large=100M
database_next_ext_small=10M
database_next_ext_default=10M
```

You may choose to customize a particular system type's storage parameters (initial and next extent). User defined types pick up the extent class of their super-type. This behavior cannot be changed.

```
[TYPE_EXTENT_SIZE]
database_ini_ext_dm_document=10M
database_next_ext_dm_document=100M
database_ini_ext_dm_cabinet=100K
database_next_ext_dm_cabinet=100K
```

To gather the sizes of existing Documentum objects within Oracle select from the following views:

```
sqlplus docbase_owner_name/password

SQL> set linesize 500;
SQL>
SQL> select segment_name, bytes from user_segments where
segment_type='TABLE';
```

For tables, and:

```
SQL> set linesize 500;
SQL>
SQL> select segment_name, bytes from user_segments where
segment_type='INDEX';
```

For indexes.

If you wish to predefine which tables and indexes should be created in which tablespaces, you may add entries to the [FUNCTION_SPECIFIC_STORAGE] and [TYPE_SPECIFIC_STORAGE] sections of the server.ini.

FUNCTION_SPECIFIC_STORAGE allows you to define the default locations for small and large tables and indexes. For example:

```
[FUNCTION_SPECIFIC_STORAGE]
database_table_large=dm_large_ tbsp _data
database_table_small=dm_small_ tbsp _data
database_index_large=dm_large_tbsp_idx
database_index_small=dm_small_tbsp_idx
```

Specific types can be redirected to storage areas by customizing the TYPE_SPECIFIC_STORAGE section.

```
[TYPE_SPECIFIC_STORAGE]
database_table_dmi_object=dm_massive_tbsp_data
database_index_dmi_object=dm_massive_tbsp_idx
```

This can be used to isolate the most active tables and indexes on the fastest disk drives, or balance I/O across drive groups.

Sybase

In this section cover how to check the size of each table in sybase. Sybase has fixed sized segments and fixed sized devices. That is, there is no autoextend capability (ability to automatically grow the database file). Hence, once the space allocated for a database has been exceeded on a particular device either additional space needs to be allocated within that device, or an additional device is required. Given these restrictions quite important to get the allocation of space correct for Sybase to avoid any additional work or unpleasant surprises in the future.

The current space allocation for a database may be determined using the sp_helpdb stored procedure (from within Sybase's isql program):

```
1> use db_EDMI_docbase
1> sp_helpdb db_EDMI_docbase
 name                    db_size       owner                    dbid
         created
         status

 ----------------------- ------------ ----------------------- ------
        -------------
        ------------------------------------------------------------------------
-----------------------------
 db_EDMI_docbase               2010.0 MB sa                           5
         Jul 28, 1999
         no options set


(1 row affected)
 device_fragments               size          usage                free kbytes
 ----------------------------- ------------ -------------------- -----------
 db_EDMI_docbase_data             1000.0 MB     data only                748096
 db_ EDMI_docbase_data1            300.0 MB      data only               307200
 db_ EDMI_docbase_data1            300.0 MB      data only               307200
 db_ EDMI_docbase_data1            400.0 MB      data only               259936
 db_ EDMI_docbase_log              10.0 MB       log only                 10240
```

The current size of a table in Sybase may be determined from the sp_spaceused stored procedure, for example (from within Sybase's isql):

```
1> sp_spaceused dm_sysobject_s
2> go
 name                 rowtotal    reserved        data       index_size   unused
 -------------------- ----------- --------------- --------   ---------- ---------------
 dm_sysobject_s       63177       55370 KB        42894 KB   10282 KB      2194 KB

(1 row affected)
(return status = 0)
```

Inside the Documentum portion of the database the names of all Documentum tables can be found using the following query:

```
select name from sysobjects where type = 'U'
go
```

Space usage can also be estimated. This is extremely useful when planning for growth (example executed from within Sybase's isql program):

```
1> sp_estspace
2> go
USAGE:
sp_estspace table_name, no_of_rows, fill_factor, cols_to_max, textbin_len, iosec
where
      table_name  is the name of the table,
      no_of_rows  is the number of rows in the table (>0),
      fill_factor is the index fill factor. Values range from
                   0 to 100. (default = 0,in which case internal
                   fill factor will be used)
      cols_to_max is a list of the variable length columns for which
                   to use the maximum length instead of the average
                        (default = null)
      textbin_len is length of all the text and binary fields per
                   row ( default =  0).
      iosec       is the number of I/Os per second on this machine
                   (default = 30)
Examples: sp_estspace titles, 10000, 50, "title, notes", null, 25
          sp_estspace titles, 50000
          sp_estspace titles, 50000, null, null, null, 40

1> sp_estspace dm_sysobject_s, 1000000
 name                          type         idx_level Pages        Kbytes

 ----------------------------- ------------ --------- ------------ ------------
 dm_sysobject_s                data                 0       325355       650711
 d_1f003a6780000000e           nonclustered         0         6580        13160
 d_1f003a678000000e            nonclustered         1           44           88
 d_1f003a678000000e            nonclustered         2            1            2
 d_1f003a678000002f            nonclustered         0        23811        47620
 d_1f003a678000002f            nonclustered         1          568         1134
 d_1f003a678000002f            nonclustered         2           14           28
 d_1f003a678000002f            nonclustered         3            1            2
 d_1f003a6780000032            nonclustered         0        21740        43480
 d_1f003a6780000032            nonclustered         1          474          946
 d_1f003a6780000032            nonclustered         2           11           22
 d_1f003a6780000032            nonclustered         3            1            2
 d_1f003a6780000108            clustered            0         6580        13160
 d_1f003a6780000108            clustered            1           44           88
 d_1f003a6780000108            clustered            2            1            2

(1 row affected)
 Total_Mbytes
 ----------------
           823.31
 name                          type         total_pages  time_mins
 ----------------------------- ------------ ------------ ------------
 d_1f003a6780000000e           nonclustered         6625          218
 d_1f003a678000000f            nonclustered        14089          259
 d_1f003a678000002a            nonclustered        22226          304
 d_1f003a678000002f            nonclustered        24394          316
```

```
 d_1f003a6780000032            nonclustered        22226          304
 d_1f003a6780000108            clustered           331981         1988
(return status = 0)
```

Microsoft SQL Server

As with Sybase, the Microsoft SQL Server has the sp_spaceused command which can be used to gather the sizes of each table. It also has a GUI admin tool which can be used to obtain the same information (although it is difficult to move information displayed in the GUI tool to another file). This SQL Server administrator tool is also useful for displaying the space used in a database. This is shown below.

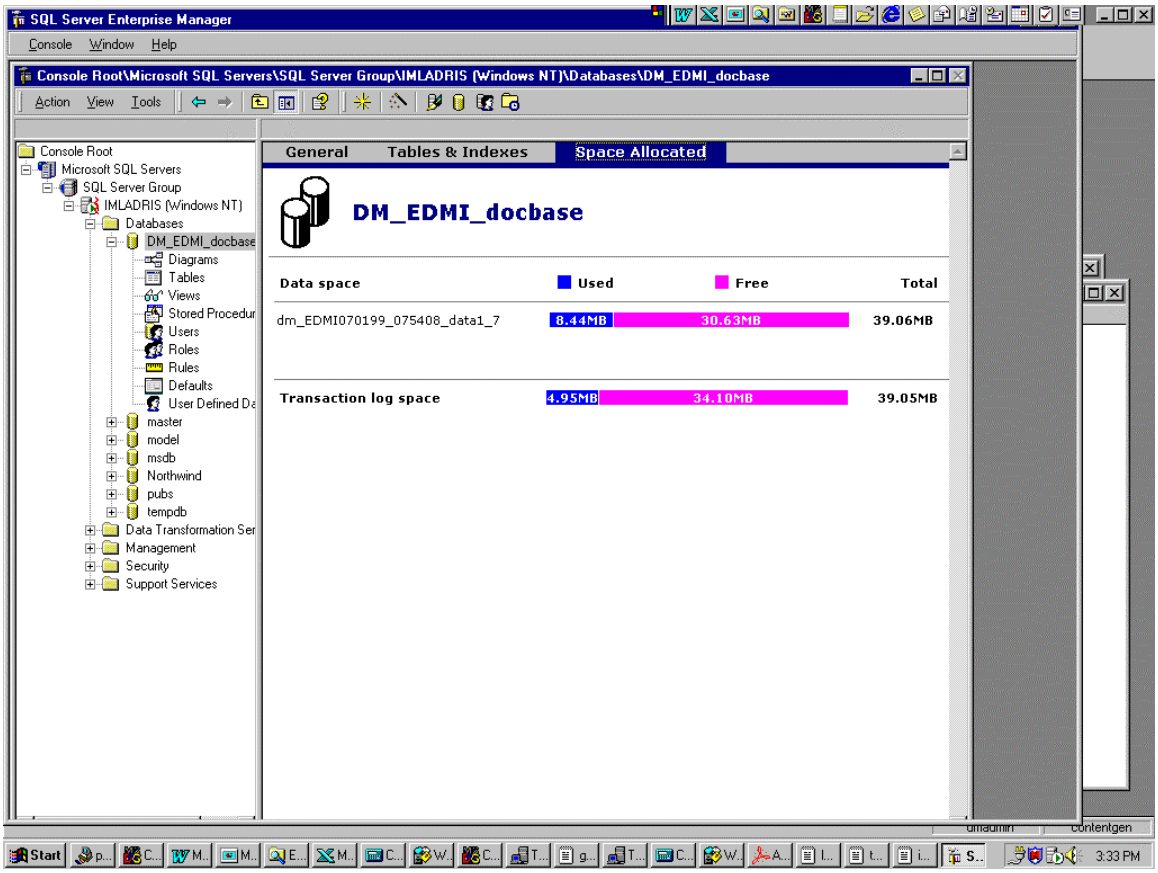


Figure 18: Showing DBMS file space usage using the SQL 7 Administrator

How do I obtain DBMS queries from Documentum?

Documentum DQL queries are parsed and translated into the target SQL for the DBMS. These translated queries can be can be viewed when the DMCL trace is enabled (as described earlier).

Once the trace level has been set to 10 the log file for the session will also contain the translated SQL queries. For example, if we execute the following api commands:

```
trace,c,10,'c:\temp\trace.trc'
id,c,dm_sysobject where object_name = 'hello'
```

Then within %DOCUMENTUM%\dba\log\000*\ there will be a subdirectories for each user. Within one of these sub-directories will be a text file with contents that would, in this case, look like:

```
Mon Aug 23 12:43:24 1999 501000 [DM_SESSION_I_SESSION_START]info:  "Session
010026ac80000422 started for user EDMI."
Mon Aug 23 12:43:24 1999 812000 [DM_QUERY_T_SYNTAX_BEGIN]info:  "Begin syntactic parse
(call yacc)."
Mon Aug 23 12:43:24 1999 942000 [DM_QUERY_T_SYNTAX_COMPLETE]info:  "Syntactic parse is
complete."
Mon Aug 23 12:43:25 1999 082000 [DM_QUERY_T_SELECT_BEGIN]info:  "Begin SELECT statement."
Mon Aug 23 12:43:25 1999 132000 [DM_QUERY_T_SQL_SELECT]info:  "SELECT statement generated
by the query is: select all dm_sysobject.r_object_id from dm_sysobject_sp dm_sysobject
where (dm_sysobject.object_name='hello') and (dm_sysobject.i_has_folder = 1 and
dm_sysobject.i_is_deleted = 0).  Begin Database processing."
Mon Aug 23 12:43:25 1999 893000 [DM_QUERY_T_CLOSE_BEGIN]info:  "Begin database cursor
close (CreateResultObject())."
Mon Aug 23 12:43:25 1999 893000 [DM_QUERY_T_CLOSE_COMPLETE]info:  "Database cursor close
is complete."
Mon Aug 23 12:43:25 1999 893000 [DM_QUERY_T_SELECT_COMPLETE]info:  "SELECT statement
semantic checking and setup is complete."
Mon Aug 23 12:43:26 1999 003000 [DM_SESSION_I_SESSION_QUIT]info:  "Session
010026ac80000422 quit."
```

We've marked the translated SQL query as italics in the trace above. Once this query has been obtained, then it can be fed into the appropriate DBMS command to get the plan information. Note also that this trace can be useful in determining which queries are taking up the time since the trace contains timing information as well.

However, note that not all is accomplished from DQL queries. Some Documentum operations generate SQL directly.

How do I Capture Query Plan Information for the RDBMS?

Oracle

The following is a step-by-step procedure in figuring how out to get plan information for Documentum queries when running Oracle. This may perhaps be useful when trying to understand if an additional index is required for some application-defined DQL query or to evaluate the benefits/disadvantages between two different optimization strategies.

The basic procedure is to enable a flag called sql_trace (along with another flag called timed_statistics). This will cause trace files to be generated that can be later converted into a file that provides the plan for each query plus the time needed to execute it.

Enabling Global Tracing From the init.ora File

When the sql_trace flag is enabled in the init.ora file it will cause trace files to be generated for each Oracle session that is established. Each session will have its own trace file. The trace is enabled by having the following two items in the init.ora file and restarting the Oracle instance:

```
sql_trace = true
timed_statistics = true
```

To turn off the tracing, change the value of sql_trace to false and restart the Oracle instance.

The advantage of this approach is that there are times when it is difficult to enable SQL tracing from the application. The disadvantages are that firstly, the Oracle instance needs to be restarted [hence this may only be useful prior to deployment] and secondly, that this will result in many trace files (since all sessions are traced) potentially leading to confusion as to which file is the desired one.

To help locate the desired trace file (among the many generated) one can “grep” for the user name of the account that is being used to run the test scenario. If this user name is used only for that test scenario then only its file will contain that name embedded in some of the Documentum generated queries.

For example, suppose our test user is called tuser8 and there has been lots of trace files generated. Then if we grep through the following files:

```
F:\Oracle\ADMIN\ORCL8\udump>ls *.trc
ORA00044.TRC        ORA00163.TRC        ORA00239.TRC        ORA00309.TRC
ORA00047.TRC        ORA00164.TRC        ORA00240.TRC        ORA00310.TRC
ORA00049.TRC        ORA00165.TRC        ORA00241.TRC        ORA00311.TRC
ORA00052.TRC        ORA00166.TRC        ORA00242.TRC        ORA00312.TRC
ORA00062.TRC        ORA00167.TRC        ORA00243.TRC        ORA00313.TRC
ORA00065.TRC        ORA00168.TRC        ORA00244.TRC        ORA00314.TRC
ORA00066.TRC        ORA00169.TRC        ORA00245.TRC        ORA00315.TRC
ORA00067.TRC        ORA00171.TRC        ORA00246.TRC        ORA00316.TRC
ORA00068.TRC        ORA00172.TRC        ORA00247.TRC        ORA00317.TRC
ORA00069.TRC        ORA00174.TRC        ORA00248.TRC        ORA00318.TRC
```

We can find the one that corresponds to the test session for tuser8:

```
ORA00044.TRC:select all dm_folder.r_object_id from dm_folder_sp dm_folder where
(dm_folder.r_object_id in (select r_object_id from dm_folder_r where r_folder_pa
th='/System/DFC Customizations/Startup Items')) and (dm_folder.i_has_folder = 1
and dm_folder.i_is_deleted = 0) and  ( dm_folder.owner_name = 'test user - tuser
8' or exists (select * from dm_acl_sp, dm_acl_rp where dm_acl_sp.r_object_id = d
m_acl_rp.r_object_id and dm_folder.acl_domain = dm_acl_sp.owner_name and dm_fold
er.acl_name = dm_acl_sp.object_name and dm_acl_rp.r_accessor_permit >= 2 and (dm
_acl_rp.r_accessor_name = 'dm_world' or dm_acl_rp.r_accessor_name = 'test user -
 tuser8' or (dm_acl_rp.r_is_group = 1 and dm_acl_rp.r_accessor_name in ('tusers'
)))))
```

Enabling SQL_TRACE from PL SQL

If the desired Oracle query is known (perhaps obtained from the Documentum session log after turning DMCL tracing to level 10), then it could be executed via a PL-SQL script. The sql_trace can be enabled via such a script. For example, suppose we have a query that we’d like to trace. We can create a script that looks as follows:

```
alter session set sql_trace = true ;
select * from dmi_change_record_sv DG where DG.r_object_id= -2147400647;
alter session set sql_trace = false ;
```

This can be executed on UNIX (for Oracle 7.X and 8.X) or Windows NT 8.1.5 as:

```
sqlplus docbaseowner/password < script.sql
```

Or on Windows NT with Oracle 7.3.4 as:

```
plus33 docbaseowner/password < script.sql
```

This will cause the trace file to be generated for that session. The advantage of this approach is that only a single session is traced. The disadvantage is that it is not always possible to obtain the Documentum generated query from the DMCL trace.

Enabling SQL_TRACE from Documentum DMCL

A third option for enabling the tracing of Oracle plan information is to enable it programatically through DMCL. This is accomplished in the following manner:

```
qid = dmAPIGet("apply,c,NULL,EXEC_SQL,QUERY,S,'alter session set sql_trace true'")
if ( qid = "" ) then
        print "exec sql failed exiting"
        print dmAPIGet("getmessage," & session)
        exit sub
}
rc = dmAPIExec("next," & session &  "," & qid)
while ( rc <> 0 )
        bool_val = dmAPIGet("get," & session & "," & qid & ",result")
        rc = dmAPIExec("next," & session & "," & qid)
Wend
if ( bool_val <> "T" ) then
        print "exec sql returned false (" & bool_val & ") exiting"
        exit sub
}
```

A superuser can also do “ad hoc” sql_tracing from the interactive message tester in Workspace or the integrated desktop client by issuing the following command:

```
execsql,c,alter session set sql_trace=true
```

Simply perform the actions or queries that you want to collect statistics for, then issue:

```
execsql,c,alter session set sql_trace=false
```

The trace file will contain the information for that session only.

Converting the Oracle Trace File to See The Query Plan

Once a trace file has been generated and identified, then the next step is to convert it so that the plan information is made available. This is accomplished using the tkprof utility. Suppose the trace file name is mysession.trc, then on UNIX (or NT with Oracle 8.X):

```
tkprof mysession.trc mysession.out explain=docbaseowner/password
```

On Windows NT with Oracle 7.3.4:

```
tkprof73 mysession.trc mysession.out explain=docbaseowner/password
```

The file mysession.out will contain the Oracle trace information. The plan information for a query should look something like the following:

```
select all dm_user.user_privileges
from
 dm_user_sp dm_user where (dm_user.user_name='test user - tuser3')
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.01       0.02          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0          6          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        6      0.01       0.02          0          6          0           2

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 37  (TUSER8)

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  TABLE ACCESS BY INDEX ROWID DM_USER_S
      2   INDEX RANGE SCAN (object id 12310)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
      1   TABLE ACCESS (BY INDEX ROWID) OF 'DM_USER_S'
      2    INDEX (RANGE SCAN) OF 'D_1F246FA680000017' (NON-UNIQUE)
```

Useful CPU, response time, and disk statistics

Query plan

Oracle trace plan information is described in other sources, see the reference section at the end of this document for other sources that help to explain the Oracle query plan output.

Sybase

The main tool for displaying query plan information with Sybase is the showplan command. Once a suspicious query is located it can be fed through the Sybase isql command to display the query plan.

```
use db_EDMI_docbase
go
set statistics io on
go
set statistics time on
go
set showplan on
go
select all dm_user.user_privileges
from
 dm_user_sp dm_user where (dm_user.user_name='test user - tuser3')
go
```

This generates the following output:

```
1> use db_EDMI_docbase
1> set statistics io on
Total writes for this command: 0

Execution Time 0.
SQL Server cpu time: 0 ms.  SQL Server elapsed time: 36726 ms.
1> set statistics time on
Parse and Compile Time 0.
SQL Server cpu time: 0 ms.
Total writes for this command: 0

Execution Time 0.
SQL Server cpu time: 0 ms.  SQL Server elapsed time: 0 ms.
1> set showplan on
Parse and Compile Time 0.
SQL Server cpu time: 0 ms.
Total writes for this command: 0
```

Query

Query plan for the query

```
Execution Time 0.
SQL Server cpu time: 0 ms.  SQL Server elapsed time: 0 ms.
1> select all dm_user.user_privileges
2> from
3>   dm_user_sp dm_user where (dm_user.user_name='test user - tuser3')

QUERY PLAN FOR STATEMENT 1 (at line 1).


    STEP 1
        The type of query is SELECT.

        FROM TABLE
            dm_user_s
        Nested iteration.
        Index : d_1f003a6780000017
        Forward scan.
        Positioning by key.
        Keys are:
            user_name  ASC
        Using I/O Size 2 Kbytes for index leaf pages.
        With LRU Buffer Replacement Strategy for index leaf pages.
        Using I/O Size 2 Kbytes for data pages.
        With LRU Buffer Replacement Strategy for data pages.

Parse and Compile Time 0.
SQL Server cpu time: 0 ms.
```

Execution time information

```
Table: dm_user_s  scan count 1,  logical reads: (regular=2 apf=0 total=2),
physical reads: (regular=0 apf=0 total=0),  apf IOs used=0
Total writes for this command: 0

Execution Time 0.
SQL Server cpu time: 0 ms.  SQL Server elapsed time: 0 ms.
 user_privileges
 ---------------

(0 rows affected)
```

Disk IO Information

Microsoft

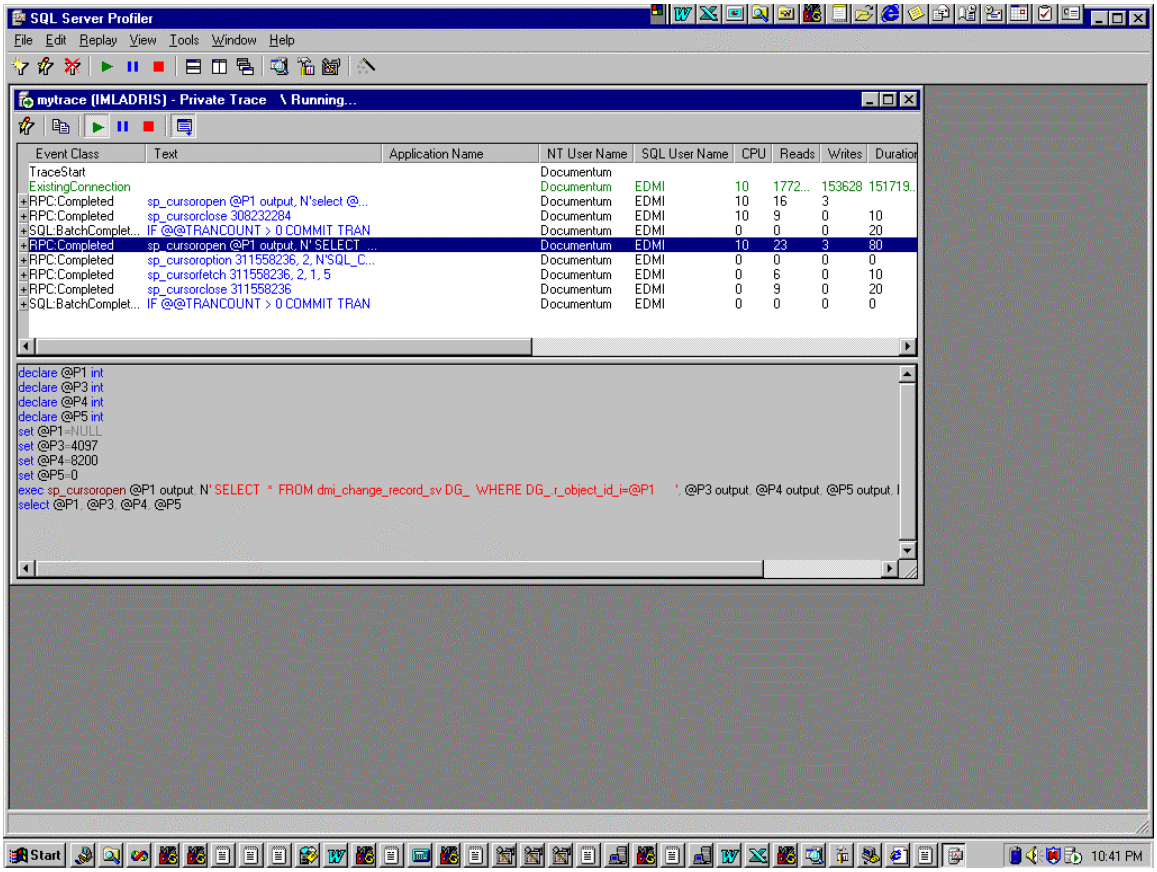
There are several tools needed to generate plans for Microsoft SQL 7.

Capturing Queries Using the SQL Profiler
The SQL profiler is a tool to capture certain events generated by the SQL Server. One of the default events provides the text of the queries to the server. This text can be extracted and fed back to the server to view the corresponding plan.

The SQL Profiler is started by selecting (from the NT start menu):
Start→Programs→Microsoft SQL 7.0 → Profiler.

Once this is started then a trace has to be enabled by selecting File→New→Trace. Provide a filename for the trace file and click start. This will start a trace session, as shown below.

FIGURE 19: SQL 7 Profiler



These traces can be saved to a file (File→Save) for viewing later. In addition, the queries can also be dumped to a file using File→Save As→SQL Script. A particularly long running query can also be copied from the profiler window and pasted to a script file.

Obtaining Plans using the Query Analyzer

Once a query has been singled out, it can have its plan displayed using the SQL 7 Query Analyzer. This tool can be found from Start→Programs→SQL Server 7.0→Query analyzer. Once started you can copy and paste the desired query into the command window and display the command . For example the plan for the query shown above is shown below.

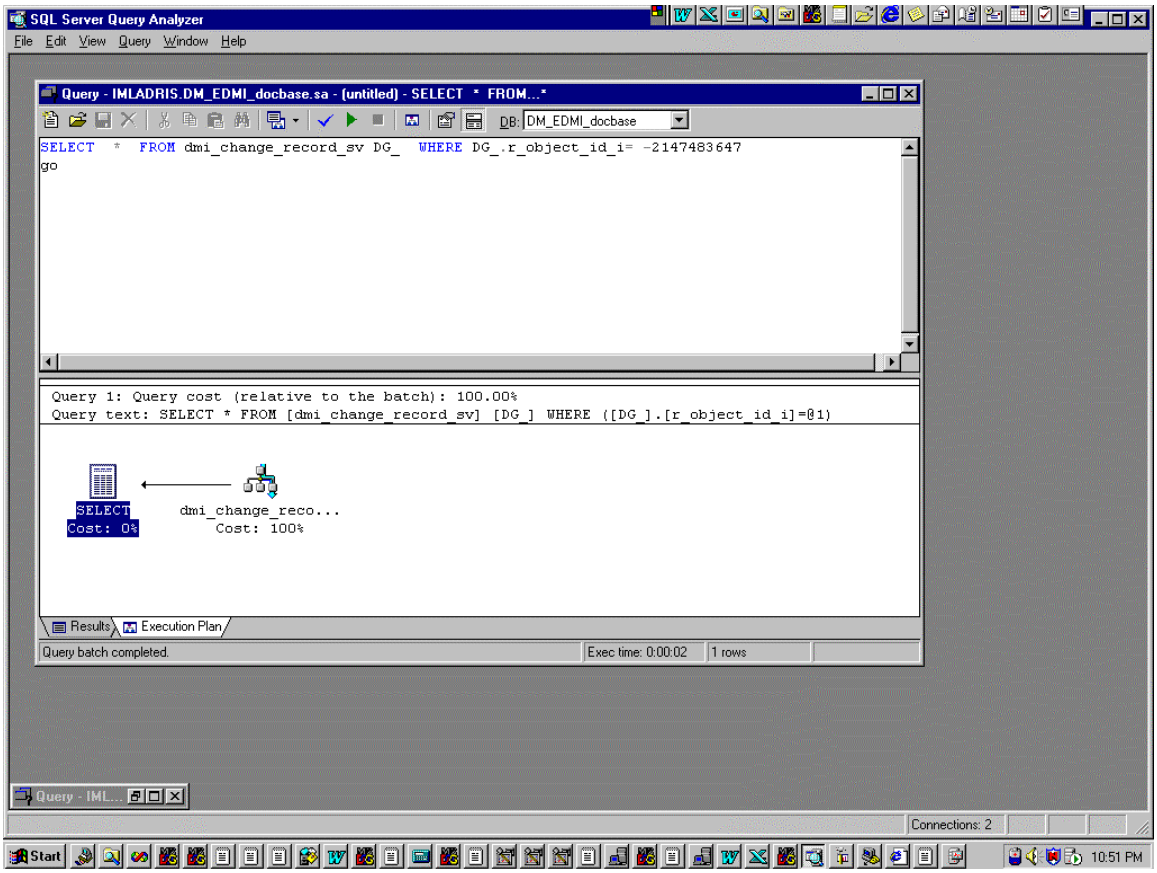


FIGURE 20: SQL 7 Query Analyzer

The disadvantage of using the Query analyzer is that you can only view or print the output. You are unable to dump it to a file for later use.

Obtaining Plans Using Showplan

Another approach at getting the query plan is the enable showplan output. For example while within isql (or the query analyzer it self) one could run:

```
set showplan_all on
go
select * from dmi_change_record_sv DG where DG.r_object_id = -2147400647
go
```

to get the following:

```
StmtText               StmtId      NodeId      Parent      PhysicalOp
LogicalOp                     Argument DefinedValues EstimateRows            EstimateIO
EstimateCPU             AvgRowSize  TotalSubtreeCost         OutputList Warnings Type
Parallel EstimateExecutions
---------------------- ----------- ----------- ----------- -----------------------------
------------------------------ -------- ------------ ---------------------- ----------
-------------- ------------------------ ----------- ------------------------ ---------- -
------- ------------------------------ -------- ------------------------
set showplan_all on    4           1           0           NULL
NULL                          NULL     NULL          NULL                          NULL
NULL                    NULL        NULL                       NULL       NULL     SETON
0        NULL

(1 row(s) affected)

StmtText
StmtId      NodeId      Parent      PhysicalOp                     LogicalOp
Argument
DefinedValues
EstimateRows             EstimateIO               EstimateCPU              AvgRowSize
TotalSubtreeCost         OutputList
Warnings                                       Type                             Parallel
EstimateExecutions
---------------------------------------------- ------------------------------ -------- -
-----------------------
select * from dmi_change_record_sv DG where DG.r_object_id = -2147400647
5           1           0           NULL                               NULL
NULL
NULL
NULL                    NULL                     NULL                     NULL
NULL                    NULL
NULL                                          SELECT                           0
NULL
  |--Clustered Index
Scan(OBJECT:([DM_EDMI_docbase].[dbo].[dmi_change_record_s].[d_1f0026ac80000144]),
WHERE:([dmi_change_record_s].[r_object_id]=Convert([@1])))  5           3           1
Clustered Index Scan          Clustered Index Scan
OBJECT:([DM_EDMI_docbase].[dbo].[dmi_change_record_s].[d_1f0026ac80000144]),
WHERE:([dmi_change_record_s].[r_object_id]=Convert([@1]))
[dmi_change_record_s].[r_object_id_i], [dmi_change_record_s].[r_object_id],
[dmi_change_record_s].[type_change_count], [dmi_change_record_s].[storage_change_count],
[dmi_change_record_s].[reinit_change_count], [dmi_change_record_s].[cache_change_count],
[d 1.0                    3.7578501E-2            7.9600002E-5             51
3.7658099E-2             [dmi_change_record_s].[r_object_id_i],
[dmi_change_record_s].[r_object_id], [dmi_change_record_s].[type_change_count],
[dmi_change_record_s].[storage_change_count],
[dmi_change_record_s].[reinit_change_count], [dmi_change_record_s].[cache_change_count],
[d NO STATS:([dmi_change_record_s].[r_object_id])  PLAN_ROW                        0
1.0

(2 row(s) affected)
```

The output from showplan is very wide (which is why the above text looks so jumbled), so we recommend using the –w column_width option when using isql and showplan. We've highlighted the most important part of this output above.

Other useful commands

As with Sybase, SQL Server provides commands that display information on how long queries take (STATISTICS TIME) and how many I/Os are needed to serve them (STATISTICS IO).

Is that Disk I/O Fast Enough?

In many cases on UNIX it is not possible for the RDBMS server to employ asynchronous I/O on normal file system files. In those cases its necessary to use raw devices. Asynchronous I/O can significantly improve the performance of the RDBMS.

On Windows NT asynchronous I/O is supported on NTFS-based files for both Oracle and Microsoft SQL Server.

BEQUEATH might be very helpful for Host-based Oracle Installs

Oracle has a client connection protocol mechanism called the BEQUEATH driver that could provide good performance when the Oracle RDBMS and the Docpage (or EDM) server are running on the same host. This “networking” driver uses native OS IPC mechanisms rather than TCP/IP. This can be helpful to alleviate the “stress” on the TCP/IP layer especially when RightSite and Netscape are also running on the same machine. This “driver” cannot be used if the Oracle is on a separate machine from the DocPage (EDM) server.

An example setup for the Oracle tnsnames file (using BEQUEATH) is shown below:

```
dmbeq=
(DESCRIPTION=
  (ADDRESS=
   (PROTOCOL=BEQ)
   (program=/export/nfs2-1/app/oracle/product/7.3.4/bin/oracle)
   (argv0=oracleDMNOTES)
   (args='(description=(local=yes) (address=(protocol=beq)))')
  )
  (CONNECT_DATA=
   (SID=DMNOTES)
  )
)
```

An example entry in the Documentum server.ini would look like:

```
[SERVER_STARTUP]
docbase_id = 1
docbase_name = dm_EDMI
concurrent_sessions = 300
database_conn = dmbeq
```

Rule-based and Cost-based Optimization with Oracle

The biggest demands on the disks typically come from large table scans. A table scan occurs when a DBMS reads all of the data for a table looking for a few rows. That is, in a table scan the DBMS does not use an index to shorten the lookup effort. A database index is very similar to an index in a book. It is a data structure that allows the DBMS to locate some records efficiently without having to read every row in the table. Documentum maintains many indexes on tables and even allows the Administrator to define additional ones on their underlying tables. Now table scans are not always a bad thing (in some cases using an index can actually hurt performance). However, if the table is large enough or the physical RAM is small enough, then table scans generate enormous amounts of disk I/O.

In general, these large table scans should not appear in your workload, but there are certain operations in EDMS 98 that are likely to lead to a table scan. For example, if a case insensitive attribute search occurs then the DBMS cannot use an index and this leads to a table scan. When large table scans do occur, and can't be tuned away, then its important to size the server's disks to get best performance.

Some table scans are unavoidable (case insensitive search), others are the result of query optimization problems by the DBMS vendors. Some vendors, like Oracle, actually support multiple modes of query optimization. One popular one that has been used early on at Documentum relies on the Oracle *rule-based* optimizer. This optimizer will pick a data access plan based on a sequence of rules, not on the statistics of the tables (i.e., the number of rows). As DocBases get larger this, at times, will lead to some costly mistakes that cause table scans. On large Docbases a cost-based optimizer can, and might, deliver better access plans because it can determine table size and column value distributions. However, cost-based optimizers are not guaranteed to pick the best data access plan. Even with the table statistics they can mistakenly pick an access plan that leads to more disk I/O.

In addition, the cost-based optimizer allows you to specify a "goal". For example, if you wanted the optimizer to select the best path such that the the first row is retrieved as quickly as possible, you would set your optimizer_mode parameter in the init.ora file to "FIRST_ROWS". If you wanted the optimizer to choose the path that would return all results as quickly as possible, you would set optimizer_mode to "ALL_ROWS". Documentum performs best when ALL_ROWS is used as the optimizer mode.

If a cost-based optimizer is picked and all of the tables involved in the query have no statistics, then the rule-based optimizer is used instead. If a single table involved in the query does have statistics, then Oracle will try to guess the information for the other tables, and use that to formulate the execution plan.

To switch between the rule-based and cost-based optimization simply set the optimizer_mode in the init.ora file and generate statistics via SQL or the Update Statistics job. *Please note that cost-based optimization is generally not recommended for pre-4i docbases.*

Databases like Sybase and the Microsoft SQL Server always use cost-based optimization and need to have statistics on the various tables.

Row Level Locking in Sybase

Sybase 11.9.2 provides support for row level locking, which in many situations will provide a dramatic improvement in performance. It is strongly recommended that all Documentum tables be converted to row level locking. Row level locking is not enabled by default (the default value is ALLPAGE, which uses page-level locking).

There are two methods to ensure that row-level locking will be used:

- Run sp_configure 'lock scheme' This will ensure that any table created will be row-level locked. To have the desired effect this must be run prior to creating the Docbase. sp_configure will display the current value:

```
1> sp_configure 'scheme'
2> go
 Parameter Name        Default     Memory Used Config Value       Run Value
 --------------------- ----------- ----------- ------------      -----------
 lock scheme           allpages            0    datarows           datarows
```

```
(1 row affected)
(return status = 0)
```

- Run the “ALTER TABLE ..... LOCK datarows” command for each table that needs to be row level locked.

To check if a table has been row-level locked, execute the sp_help command on the specified table. For example:

```
1> sp_help dm_sysobject_s
 Name                           Owner
         Type
 ------------------------------ ------------------------------
        ----------------------
 dm_sysobject_s                 dbo
         user table
       :
       :
Object is not partitioned.
Lock scheme Datarows
```

Toward end of sp_help output for table will be some information on the locking scheme for that table

documentum

Networking

This section focuses on some performance issues associated with communication and networking. It is quite common for networking to account for a portion of performance issues in client/server and multi-tier environments. Most Enterprise deployments involve users with remote facilities that may (or may not) have access to high-bandwidth communication facilities.

In this section we cover several important components of communication and networking. We provide more background into latency and bandwidth issues and the Documentum network architecture. We also cover some important issues on tuning the Documentum inactivity timer for remote users.

Latency and Bandwidth

This section will outline some of the communication aspects of Documentum deployments. It is meant to give a high level summary. The networking characteristics between the Docpage server and RDBMS and the client /docpage server will be discussed.

Latency and bandwidth are some key concepts necessary for understanding the issues in network sizing.

The latency of a data transfer can be thought of as the time it takes to send some data between two hosts. Latency is affected by:

- Processing time on the respective host machines,
- The propagation delay across the transmission media,
- The available bandwidth of each network between the two hosts, and
- The processing time of all routers and gateways between the two hosts.

Bandwidth, on the other hand, covers the available throughput of the various components in the network. It is usually measured in Bytes/sec or bits/sec. It is affected by the transmission speed as well as the amount of external traffic that is also using the same media.

Using concrete examples, one could think of a single lane interstate freeway with a speed limit of 60 miles/hour as having a bandwidth of that matches the speed limit (that is, a car could cover 60 miles in an hour). The actual bandwidth (especially due to rush hour traffic) is likely to be far less (e.g., LA freeway with an average speed of 20 miles per hour). That is, the available bandwidth is likely to be diminished by external traffic forces (i.e., additional cars).

Following this example, the latency would be the time it takes to get from one place to another. The distance from Pleasanton, California to the Golden Gate bridge in San Francisco might only be about 40 miles, but the delays caused by the toll bridge between Oakland and San Francisco and the various traffic lights in San Francisco are likely to make that trip take longer than 1 hour.

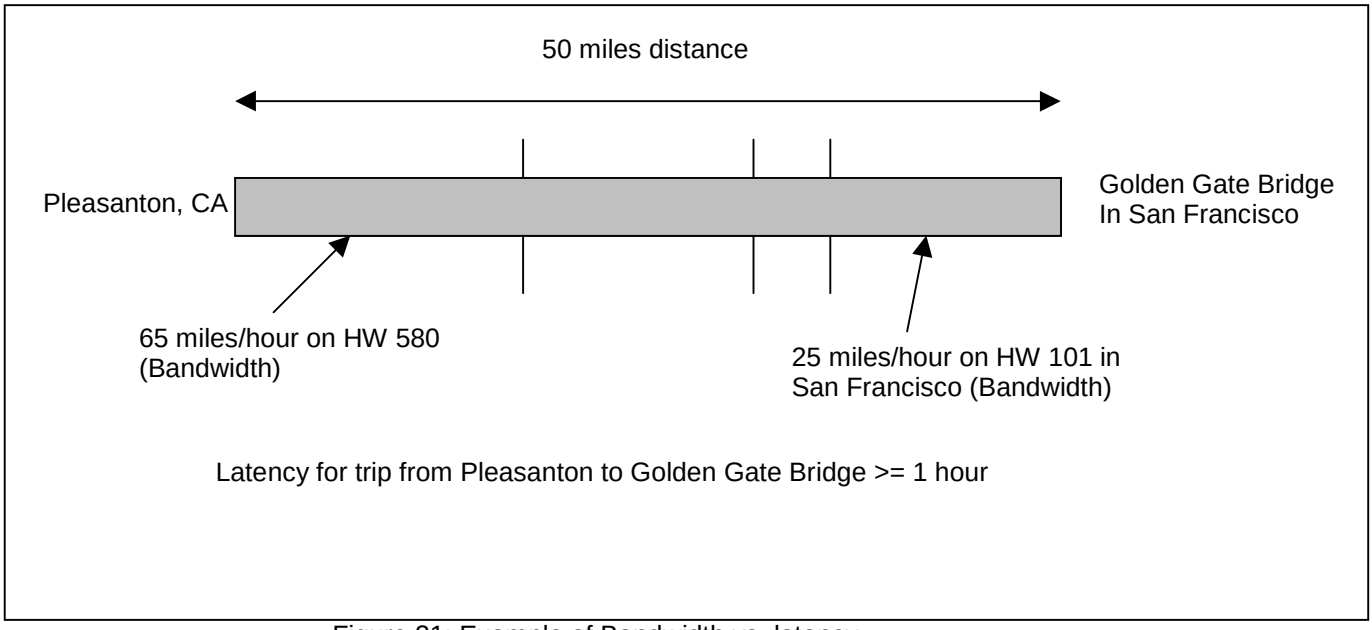


Figure 21: Example of Bandwidth vs. latency

In applying these concepts to real-world networking one must be sensitive to the various components in the network, and how they affect response time of a Documentum implementation.

Overview of Documentum Network Load

Documentum supports a flexible, multi-tier, parallel server architecture that can be mapped onto single large systems all the way to many small systems depending on the business needs and administrative preferences of the customers. In addition, distributed and replicated Docbases can be configured to meet these business and user response needs.

It is best to first understand some of the basic networking issues for a non-distributed environment. The issues associated with distributed Docbases are variations of those basic items.

The following diagram below summarizes the network load (or demand) that is generated by a Documentum installation. The Documentum users communicate to the Documentum servers using either HTTP to a web-server or by using the DMCL library directly (for example, by using the Documentum Windows Explorer Integration or Workspace). The amount of this network traffic is typically about 25% (with respect to the number of packets) of that between the Docpage (or EDM) server to the RDBMS (ignoring content transfers). This is why it is important to keep as high bandwidth as possible between the Documentum Server and the RDBMS. (An exception to this, of course, are Documentum Content servers which are optimized to minimize the interactions they have with the RDBMS. A Documentum Content server is a special Docpage (EDM) server that provides content to local users.)

One of the most likely contributors to poor response time is the lack of bandwidth relative to the amount of data that needs to be exchanged. If users perceive a difference between Web client and client/server applications (e.g., SmartSpace Intranet vs. Workspace) then the extra bandwidth taken up by HTTP/HTML are the most likely cause.

It is also important to re-enforce the notion that the network traffic between the DocPage server and the RDBMS is characterized by many small message exchanges. Hence, the available network capacity is influenced greatly by latency and reduced bandwidth.

Measuring Network latency

Its advisable to try to determine what the latency of the networking infrastructure is for the Documentum environment. One simple way to get an indication of this is to “ping” the various components of the network. Ping is a utility available on all Operating Systems and provides an end-to-end latency measurement. For example, the ping output on Windows NT looks as follows:

```
C:\Documentum\dba\config\EDMI7>ping camelot

Pinging camelot.documentum.com [172.32.0.19] with 32 bytes of data:

Request timed out.
Reply from 172.32.0.19: bytes=32 time=170ms TTL=253
Reply from 172.32.0.19: bytes=32 time=160ms TTL=253
Reply from 172.32.0.19: bytes=32 time=160ms TTL=253
```

The above example shows that there is a latency of 170 ms between the local system and the remote system (camelot) located in the United Kingdom.

Test Response time over high-latency facilities

As mentioned earlier, we strongly recommend testing custom built user interfaces over low-bandwidth, high-latency facilities prior to deployment. This will allow the consultants/developers to tune their application to meet the remote response times prior to deployment.

These tests can easily be performed over a dial-up line or by utilizing a network delay generator, if available.

Tuning the Inactivity Timeout for remote users

Both the DocPage server and the RightSite server free up inactive connections. By default the DocPage server will do so after 5 minutes of inactivity and the RightSite server will do so after 30 minutes of inactivity.

The DocPage server will disconnect from the DBMS and kill the process (UNIX) or thread (NT) that corresponds to the now inactive connection. RightSite will kill the process or thread associated with the connection.

This feature reduces the memory demands on the system and minimizes the number of concurrent DBMS sessions. The Docpage server session is re-established transparently once another command is initiated from the client. RightSite forces named user sessions to login again once the session has been timed out.

However, in either case when a session is restarted there is a startup cost that must be paid. The restart up cost includes things like reconnecting to the DBMS, resetting caches, etc. Hence, this is a tradeoff of memory (and DBMS concurrent sessions) for CPU time. For remote users the network delay in restarting the connections might be noticeable. Hence, if this becomes an issue for enough users then its advisable to increase the time for the timeout in an effort to minimize the number of session restarts that occur.

This timeout is set in the server.ini file and can be set in the following fashion:

```
client_session_timeout = 5
```

Which sets the timeout period to five minutes (the default). Remote users might want to have a client_session_timeout of one or 2 hours. Setting this to a large value is also recommended for Content servers.

Documentum EDM Server Administration

This section covers some Documentum EDM (or DocPage) server Administration monitoring and tuning techniques. These are items under control of the administrator (rather than the application developer).

Monitoring Documentum Sessions

In large multi-user scenarios it is useful to monitor the number of Active Documentum sessions. A detailed listing can be generated from the Documentum Administrator. Sometimes all that is required, however, is to know the number of active sessions. The following code fragment illustrates how to do this:

```
Sub Main1( docbase$, docbase_user$, ospasswd$, loopcnt )

  s$ = dmAPIGet("connect," & docbase$ & "," & docbase_user$ & "," & ospasswd$ )
  m$ = dmAPIGet("getmessage," & s$)
  cnt = 0
  active_cnt = 0
  for j=0 to loopcnt
       query_id$ = dmAPIGet("apply," & s$ & ",NULL,SHOW_SESSIONS")
       rc = dmAPIExec("next," & s$ & "," & query_id$)
       while rc <> 0

         result_count = dmAPIGet("values," & s$ & "," & query_id$ & ",session")
         for i=0 to result_count
            status = dmAPIGet("get," & s$ & "," & query_id$ &_
                         ",session_status[" & i & "]")
            if ( status = "Active" ) then
                    active_cnt = active_cnt + 1
            end if
            cnt = cnt + 1
         next i
         rc = dmAPIExec("next," & s$ & "," & query_id$)

       Wend
       status = dmAPIExec("close," & s$ &  "," & query_id$)
       print "there are " & cnt & " sessions " & active_cnt & " Active sessions"
       Sleep 12000
       cnt = 0
       active_cnt = 0
   next j
End Sub
```

An example output from this program:

```
there are 290 sessions 20 Active sessions
there are 288 sessions 20 Active sessions
there are 288 sessions 20 Active sessions
there are 288 sessions 19 Active sessions
```

Documentum DocPage Server caches

The Docpage server has several caches used to conserve on operations like DBMS interactions, CPU cycles, and network operations. Most of the caches are small (less than 1M byte) and bounded by the number of objects. By far the most dominant memory usage by the DocPage server lies with its “global type cache”. This cache holds structures used to enable fast access to the DBMS tables that make up a type’s instances. The size of this cache is limited by the number of types in the system. The amount of real memory consumed is determined by the number of instances/types that are accessed. Although this cache is called the “global” cache, it primarily

functions as a per-session access structure. Each Docpage process/thread will have its own copy.

Getting around Docbroker Bottlenecks

In certain large multi-user environments the Docbroker can become a bottleneck. This is nearly always manifested in slow connection times under high multi-user load. The recommended course of action in such cases is to employ multiple Docbrokers and partition the user community to use one or the other as their primary docbroker. For example, suppose we have 3000 users that normally use a particular Docbroker for their various Docbases and this Docbroker runs on a machine called 'Broker1'. It has been determined that this Docbroker cannot service this entire user community with acceptable response time. There are several options possible to add additional Docbrokers. In general all of these solutions require running multiple Docbrokers and partitioning the user communities to use one or the other as their primary Docbroker. Changing the Docbroker that a user employs is a matter of changing their dmcl.ini to point to the desired Docbroker.

Multiple Docbroker hosts

The Documentum Administrator manual describes how to setup a separate Docbroker on a different machine. Once this Docbroker has been spawned it is necessary to update the server.ini files for each Docpage (or EDM) server to ensure it projects to that Docbroker.

Pre-4i Multiple Docbrokers on a Single host

Prior to Documentum 4i the only way to spawn multiple Docbrokers on the same host was to have multiple IP addresses on that host. Each Docbroker could bind on the separate IP addresses. Each separate Docbroker needs to be spawned with a Docbroker init file which specifies the IP address (or corresponding symbolic name). The format of this Docbroker init file is as follows:

```
[DOCBROKER_CONFIGURATION]
host=IP address (or symbolic name)
```

To spawn the Docbroker to use this file execute:

```
dm_docbroker.exe –init_file filename
```

Where *filename* is the name of the init file (it must be the full path name on NT).

Several IP addresses can be configured for a single host by either having separate Ethernet boards or by multi-homing a single board. That is, if supported by the OS, it is possible to have a single Ethernet board support multiple IP addresses. This technique is called mutli-homing. This can be accomplished in the following way on Solaris using ifconfig (for example):

```
Ifconfig le0:1 192.170.120.240 up 0xffffff00

Ifconfig le0:2 192.170.120.241 up 0xffffff00
```

Where le0 is the name of the interface.

4i Multiple Docbrokers on a Single host

In Documentum 4i the Docbroker configuration has been enhanced to allow the [DOCBROKER_CONFIGURATION] section of the Docbroker init file to support multiple ports. This makes it easier to support multiple Docbrokers per host, so that multi-homing is not required.

To setup multiple Docbrokers on the same host with Documentum 4i do the following:

- Define a service name for each Docbroker on the host and assign each with a different port number. On UNIX this services file is located at */etc/services*. On Windows NT the file is located at *\winnt\system32\drivers\etc\services*. The formats for both are the same. For example you might add the following two lines in that file:

  ```
  Docbroker1                1496/tcp
  Docbroker2                1497/tcp
  ```

- Then for each Docbroker init file specifiy the port number (or the service name) that it should use:

  ```
  [DOCBROKER_CONFIGURATION]
  host=IP address (or symbolic name)
  service=service name in services file (defines the port)
  ```

  For example:

  ```
  [DOCBROKER_CONFIGURATION]
  host=myhost
  service=Docbroker1
  ```

- In the EDM Server side add a port number to the [DOCBROKER_PROJECTION_TARGET_n] clause in the server.ini for each Docbroker that you want this to project to. Following our example:

  ```
  [DOCBROKER_PROJECTION_TARGET_1]
  host=myhost
  port=1497
  ```

Parallel Full Text builds

Normally, updating a full text index involves indexing only a few documents that had been added during the course of the day. However, for some environments there is a need to index many documents per day, and at times it may be necessary to re-index all of the documents. In these cases the best way to speed up the process is to index several groups of documents in parallel.

The indexing task can be run in parallel as long as the content has been partitioned into multiple filestores. There are several ways to do this (see the EDM Server Administrator's guide), one in particular is to turn specific filestores on and off-line as documents are being loaded.

Note that the benefit one derives from parallel full text indexing depends much on the server hardware platform that is used. There needs to be sufficient disk access capacity and CPU to take advantage of the parallel build.

Keeping Docbase “trimmed” with dm_clean

The dmclean utility is used to remove orphaned content and annotation objects, and unused internal ACLS. It should be run periodically to delete content in the filestores for deleted objects, which then frees disk space up for new object content.

When an object is created, it is initially assigned an “internal” ACL. These are system-generated ACLs based on the default ACL for the user and/or object type. Most applications will replace these auto-generated ACLs with system ACLs during the lifecycle of the document. The internal ACLs are no longer used, and simply take up space in the system. Over time, it is not uncommon to have tens of thousands of these ACLs in the system that will never be reused. These extra rows have been known to dramatically increase the response time for certain types of operations.

Setting the batch_hint_size appropriately

The batch_hint_size parameter determines the flow of data between:

- The server and the RDBMS in fetch operations; and
- The DocPage Server and the client on “next”

This value can be set in the the client’s dmcl.ini file

```
batch_hint_size = n
```

or programmatically

```
set,c,sessionconfig,batch_hint_size,n
```

While the default value of 20 is appropriate for most applications, you may want to increase or decrease this parameter. For example, clients accessing the DocPage Server over a WAN often see dramatic performance improvements by increasing the batch_hint_size to 50.
For queries that return a large number of rows, but very few attributes, you may want to temporarily set the batch_hint_size to a large value, then reset it to the default after the operation:

```
set,c,sessionconfig,batch_hint_size,100
execquery,c,T,select object_name from dm_document
…
set,c,sessionconfig,batch_hint_size,20
```

For queries that return a large number of attributes, you can reduce the batch_hint_size prior to the operation:

```
set,c,sessionconfig,batch_hint_size,100
execquery,c,T,select * from dm_document
…
set,c,sessionconfig,batch_hint_size,10
```

documentum

Documentum Application Tuning Issues

The following section covers some application tuning issues. That is, these are tips for avoiding some performance pitfalls when customizing and/or developing applications based on Documentum.

Using the ID function to ensure index usage (pre-4i)

Each object in a docbase can be uniquely identified by its object id. Therefore, in your pre-4i applications you will issue statements such as:

```
select object_name, title
from sop
where r_object_id='09000001801cca30';
```

Time: 11.647 sec

or:

```
objID = dmAPIGet("id,c,sop where i_chronicle_id=' 09000001801cca30' and
any r_version_label='EFFECTIVE'")
```

Time: 11.366 sec

However, these are by no means the most efficient method to perform this query. This is due to the underlying table and column structure for the objects in the database.

The primary key for the dm_sysobject_s table is not the r_object_id column. In fact, the r_object_id_i column is the primary key for the dm_sysobject_s table, and the r_object_id_i/i_position columns are the primary key columns for the dm_sysobject_r table. The r_object_id column is not indexed in either table.

In order to use the index on the r_object_id_i column, you must use the ID function.

The DQL query above should be rewritten as follows:

```
select object_name, title
from sop
where r_object_id_i=ID('09000001801cca30');
```

Time: 1.312 sec

The API call should be written as:

```
objID = dmAPIGet("id,c,sop where i_chronicle_id_i=ID('09000001801cca30')
and any r_version_label='EFFECTIVE'")
```

Time: 1.342 sec

This same principle applies to any dmID type attribute, such as i_folder_id, parent_id and child_id.

Avoiding upper() and other Functions that disable index Usage

It is possible to disable index usage when retrieving rows for a query, update or delete, based on the way that the WHERE clause is formed.

For example, using a function on an indexed column disqualifies that index:

```
select r_object_id, object_name
from sop
where upper(dept) = 'SALES';
```

Other examples of statements that will prevent the optimizer from using an index include:

```
where number_col * 2 = 50
where name like '%son'
where status != 'Approved'
```

Avoiding Repeating Attribute display in Workspace window view

The Workspace client allows user to customize the attributes that are displayed in the window. It is best to avoid adding a repeating attribute to the default display, as this will greatly increase the amount of time that it takes to display folder contents.

For example, opening a folder with 1000 objects took nearly 60 with the repeating attribute r_version_label displayed. Without the version label, it took only 7 seconds.

4i – Indexing Attributes for Faster Full-Text Searches

Pre-4i full-text searching with a WHERE clause and a SEARCH clause required that two separate operations be performed: a database attribute search and a Verity Topic search. The results of the Verity search was then loaded into a temporary table and merged with the results of the attribute search. Finally, the ACLs of each document would be checked to ensure that the user had access to the matching objects.

4i now allows printable attributes to be indexed and searched by full-text indexing. This can dramatically improve performance in combination attribute/text searches.

To mark an attribute for full-text indexing:

1. Start the Documentum Administrator and connect to the docbase.
2. Under Docbase Management, click Types.
3. Select the type for which you want to add or drop indexed attributes.
4. Select the attribute you want to add or drop from the list.
5. Select or de-select the Full-Text indexed checkbox as needed.
6. Update the index to incorporate the changes you have made.

4i – Using non-restartable dumps

Documentum 4i now allows the user to configure a dump to be non-restartable. By default dumps are restartable and this is achieved by keeping a record in the RDBMS of all items dumped to the log. The non-restartable dump feature turns off this record keeping. This improves performance by eliminating the network traffic and database operations associated with keeping this record.

4i – Full Docbase Dump

Dump performance for a full docbase can be enormously improved using the full_docbase_dump parameter. This is set in the following fashion (within the dump script):

```
set,c,l,dump_operation
full_docbase_dump
```

This will cause all of the types in the docbase to be dumped, but dumped in an efficient manner. However, the full docbase load occurs at the same speed regardless of how fast it was dumped.

4i - DQL SETFILE

Prior to Documentum 4i the only way to perform a SETFILE action (I.e., storing a file from the file system into the Docbase) was through using DMCL API. An example setfile operation looks as follows:

```
Rc = dmAPIExec( “setfile,c,” & doc_id & “,c:\foo.text,text” )
```

A setfile operation is usually done within the context of a large number of other DMCL API operations to set other attributes associated with the document. When executing over a low bandwidth WAN, the round trip network packets associated with the content transfer could contribute some significant delay.

It has always been possible to set attributes using the DQL UPDATE … OBJECT statement, but in addition, now with 4i it is possible to also perform the SETFILE from within this UPDATE … OBJECT statement. When operating over a low bandwidth connection this can improve performance by allowing the content transfer to occur in some other fashion that might be more optimized for low bandwidth lines.

In the graph and diagram below, we illustrate the performance improvement of using an all DMCL api method for setting numerous attributes and performing a setfile vs. transferring the files via FTP and using a single DQL UPDATE … OBJECT statement to set the results.

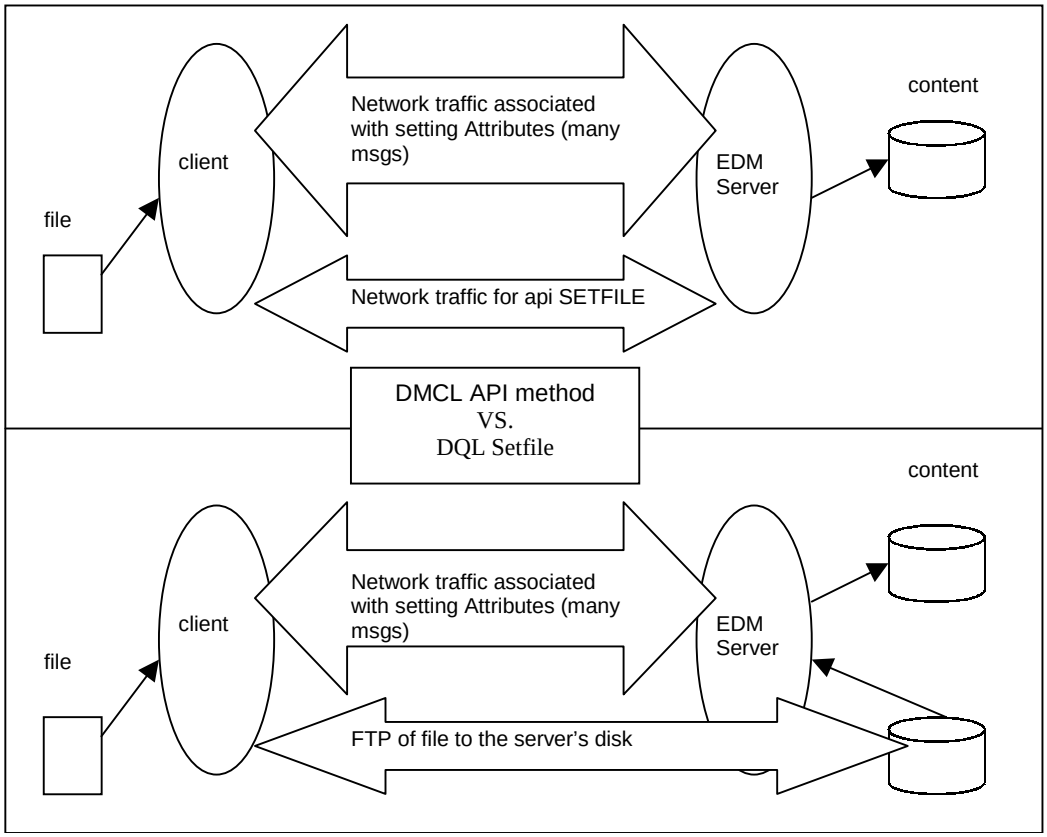


Figure 23: DMCL API setfile vs. DQL Setfile

In practice, the performance difference between protocols like FTP and Documentum for large files is small. However, one can imagine some file transfer protocols that are especially tuned for some low bandwidth or noisy lines that can out perform the Documentum content transfer. These could be employed in this case to transfer the documents to the server machine and loaded using the SETFILE within UPDATE … OBJECT.

We have found, however, that on high bandwidth networks that the SETFILE within the DMCL outperforms the SETFILE within UPDATE … OBJECT.

Avoid Enumerating All user Names/Cabinets in Menu-Pull downs

One common problem with user interfaces over low-bandwidth, high-latency communication facilities is that they tend to put exhaustive lists (like all user names or all cabinet names) into menu pull downs. The penalty for transferring all of the cabinet or user names can be high and can cause a noticeable delay in response time.

Miscellaneous Hints

When using the execsql API, try to avoid using subqueries. If possible, rewrite the query to use a join for improved performance. When using the execquery API, ensure that the Read-Only flag is set appropriately. Setting it to F for a select can dramatically slow down the execution of that query.

Documentum RightSite Server

This section covers some tuning and monitoring issues associated with the Documentum RightSite server.

RightSite, an isapi integration with IIS (and an nsapi integration to Netscape) provides web-access to the Documentum Docbase. RightSite will execute the Documentum DMCL on behalf of the Web-user to access content and attribute information.

It is important to understand that as RightSite is executing DMCL on behalf of the users, it concentrates their activity on a single server. That is, environments based on RightSite use the HTTP “thin-client” or Documentum 4-tier paradigm. In a non-Web or 3-tier environment the Documentum DMCL (client library) processing occurs on the client machine. In this HTTP “thin-client” or 4-tier software architecture, this client library processing is moved from the user PC’s onto the host that is running RightSite and the Internet Server. Very little work actually happens on the client machine (all users are assumed to be using browsers sending HTTP). That is, RightSite performs those operations that would have traditionally been done on the hundreds (or even thousands) of client PCs. This is illustrated below.

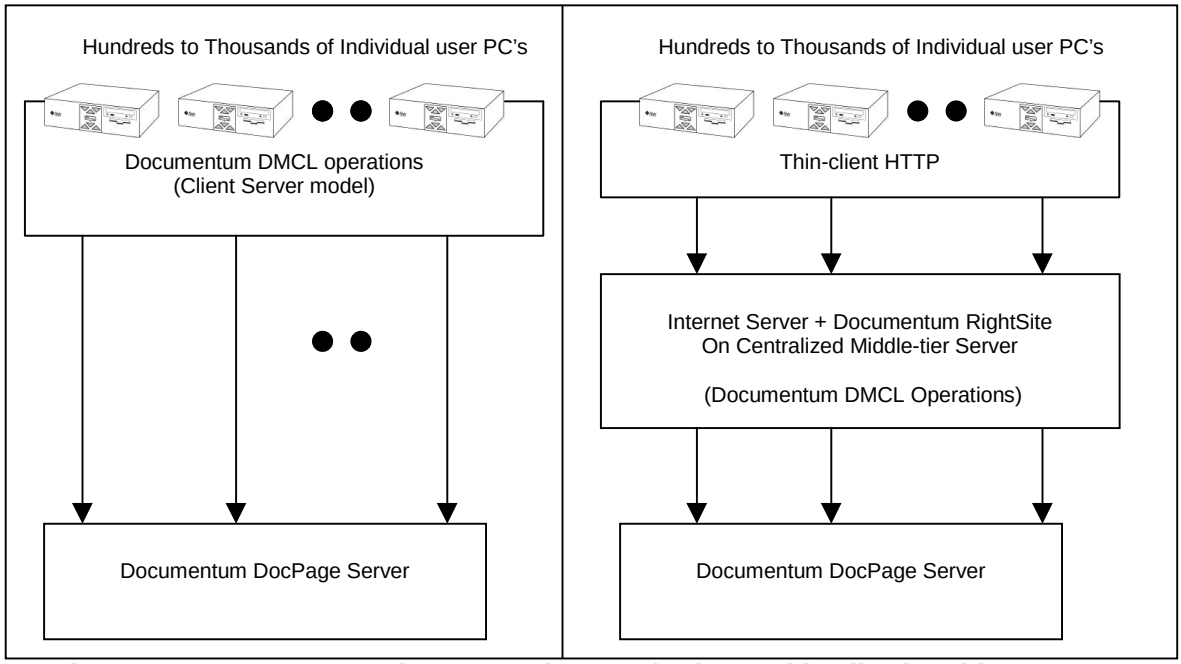


Figure 24: Documentum 3-tier, non-Web vs. 4-Tier (HTTP thin-client) architectures

This has implications on CPU consumption and on file system usage (discussed below).

NT Performance Monitor Issues

This section will go into more depth on how to monitor RightSite performance with the NT performance monitor. RightSite employs a multi-process server architecture on UNIX and Windows NT. Since the NT performance monitor does not handle multiple process-object instances with the same name well, it is typically necessary to determine the total RightSite CPU consumption indirectly (assuming it is the left-over CPU consumption once the rest of the system and process CPU consumption has been factored out).

Support Many Named Users on NT

As mentioned above RightSite will create a separate process for each named session. On Windows NT the RightSite root process (dwvsm) runs as an NT service. By default, Windows NT limits the amount of memory a service-based process can use for creating new processes. All Windows NT processes run within a “desktop” (which should just be understood as a collection of OS resources). Services run within a different “desktop” than normal processes spawned by the user. To ensure that RightSite has sufficient OS resources to spawn processes in a large multi-user environment it is necessary to ensure that “Service desktops” have sufficient memory to spawn many processes.

The following registry change needs to be made. In the following key

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\SubSystems
```

The following part of the entry for Windows as changed from:

```
SharedSection=1024,3072,512
```

To:

```
SharedSection=1024,3072,6144
```

This third memory parameter controls the amount of memory from a global heap that each desktop can use to spawn processes. By default, service-based desktops (e.g., like RightSite) have only enough of this global heap to spawn about 100 processes (the default value is typically 512K bytes or 1M byte). We increased it to 6M in the example above. However, the maximum size of the global heap that all desktops share is 48M.

If this parameter is not changed then the RightSite service will have problems spawning dwserver processes. This problem shows up as long connection or operation times (on the order of 10 – 20 minutes). It may also show up (using the Netscape server) as a failure to call some DLL like COMCTL32.DLL.

In most cases this parameter need not be changed RightSite anonymous sessions. RightSite anonymous sessions are served from a pool of servers and typically don’t need 100 servers to serve up even a large amount of requests.

Temporary File Spaces

RightSite has several temporary file space areas on disk that are good to understand to ensure excellent performance: RightSite’s DMCL file cache and its file-checkout staging area. These caches are the same as those created for normal Documentum clients. These require a bit more attention in this case because RightSite supports many users. For high-capacity environments we recommend installing RightSite on a RAID 1+0 stripe set, rather than using a single drive or a RAID5 stripe set. This will ensure that as users access content (and the content is brought over to the temporary file cache) they won’t experience any unnecessary delay.

HTTP server log growth

By default the HTTP server will log the activity that it has with RightSite. If this logging is allowed then a policy for handling these logs needs to be determined. The logs will take up disk space and could generate a noticeable amount of disk I/O.

RightSite System Parameters

Some of the system parameters that can be used to tune the performance of RightSite are as follows:

- MAX_ANON_REQS: This variable is set in the CONFIGURATION section using the RightSite administrator and requires a restart of RightSite. By default RightSite allows upto 5 requests to be queued up at each anonymous session server before a new one is started up. When this parameter is set to a smaller number, more anonymous session servers are started as requests come in, up to the maximum allowed. For example, if this parameter is set to 1, each new anonymous request results in the starting of a new session server if all existing servers are busy. Lowering the setting of this parameter essentially involves trading off response time for individual requests against overall load on the RightSite host machine since more session servers processing requests in parallel means that more load is created.

- DW_USE_GETFILE: By default RightSite uses the server API "getcontent" to retrieve content from the server and ship it down to the browser since it uses efficient in-memory content transfer as opposed to copying files around on the RightSite side. This is most useful if the underlying drives supporting RightSite are slow or overwhelmed because this feature allows RightSite to by-pass those drives.

  However if the same content is accessed by many anonymous users or the network penalty of fetching content from the Documentum Server is high, then it will be beneficial to set this parameter to "TRUE" in the ENVIRONMENT section using the RightSite Administrator (this is the default value). When the content is retrieved using the "getfile" method it will be put in the DMCL cache and won't have to be fetched across the network if accessed again.

4i – RightSite: Cache Control Attributes

Cache control attributes, when set in WebQL templates, allow a developer to control when a browser's cache should be used for certain dynamic HTML pages. This feature allows a dynamically generated page cached at the browser to be considered static. In addition, the valid lifetime of the cached page in the browser's cache can be specified. In these cases a request to the page will be made and RightSite will send an HTTP NOT_MODIFIED status to the browser for the page if it has not expired, rather than regenerating it. This feature is described in more detail in the RightSite Reference Manual.

4i – RightSite: Page Caching

This feature enables a developer to tell RightSite to cache dynamically generated pages on the file system for a period of time and keyed by a certain profile. Within this period, if another request comes in for the same page by a user with the same profile, the cached page is sent out from the file system, drastically improving performance. The profile is based on a list of variables sent to the dynamic page. This feature is described further in the RightSite Reference manual (which also contains detailed examples).

4i – RightSite: Site Caching

This features enable the site-wide caching of fairly static, computed data such that it is accessible across sessions. This can be used to store, for example, site navigation data that does not

change often but is expensive to compute. This data might be used in multiple web pages. As above this feature (with examples) is documented further in the RightSite reference manual.

4i – RightSite: Paging Through Result Sets

This feature exploits the "cachequery" feature of the DocPage server to provide web users with the ability to page through query result sets similar to AltaVista or other search engines. In the past the only way to do this was by re-executing the whole query and skipping through a portion of the result set, which was expensive. With this feature all of the results are cached at the RightSite side and a new clause (START_ROW) is provided for the DMW_REPEAT WebQL macro which allows RightSite to track where it is in the cached result set. Again, this is documented further in the RightSite Reference Manual.

documentum

References and Further Reading

Operating Systems

"*Inside Windows NT (2nd Edition)*", Solomon, Microsoft Press, 1998

"*Windows NT Resource Kit* " documentation, Microsoft Press

"*Configuration and Capacity Planning for Solaris Servers*", Wong, Sun Microsystems Press, 1997

"*Solaris Performance Administration*", Cervone, McGraw Hill Publishing, 1998

Relational Database

"*Oracle Performance Tuning*", Corrigan and Gurry, O'Reilly Publishing, 1996

"*Inside SQL Server 7.0*", Soukup and Delaney, Microsoft Press, 1997

"*Sybase Adaptive Server Enterprise Performance and Tuning Guide*", Sybase, Inc