Notes.net

Iris Today

Part 2

by Carol Zimmet
and Amy E. Smith

Level: Advanced
Works with: Domino 5.0
Updated: 09/01/2000

This is the second in a series of articles that identifies and clarifies issues and misconceptions that Domino administrators, users, consultants, and Business Partners often confront. These articles discuss concepts, options, and features whose use or non-use has a direct impact on Domino server performance or deployment and capacity-planning decisions. They attempt to set the record straight and make recommendations on how to proceed.

When it comes to improving Domino server performance and deployment and capacity planning, choosing the best options and making the best choices is a complex task. Server performance and scalability issues span multiple areas (such as mail delivery, database management, and even application design) and multiple platforms, and they are highly dependent on server configurations and the inter-relationships of Domino components.

In **Part 1** of this series, we looked at clearing up misconceptions and erroneous information being promulgated in the user community about the Domino server, as well as some recent new features that are not yet in widespread use (although they ought to be). This installment discusses some proactive steps that administrators can take to improve system performance, including adjusting system parameters, optimizing user files, and implementing sound application development practices.

It is important to note that most of the metrics and information discussed here is not new. However, the Domino Server Performance Team is looking at this information in new ways, in our efforts to make performance analysis more like a science (and so definable) rather than an art (and more like guesswork). For Domino system administrators, implementing definable performance analysis may mean being open to new ideas and trying new things, as well as minimizing risk.

Even large-scale systems need tweaking to achieve higher scalability

In this case, the large-scale system in question is Sun Microsystems' Solaris/SPARC platform. The Domino Server Performance Team achieved its scalability goals on Solaris/SPARC by tweaking a system-tunable parameter found in /etc/system in order to increase the number of file descriptors (fds)/process. Sun recommends that the default limit of 1024 file descriptors (fds)/process can be increased on the Solaris/SPARC platform (release 2.6 onwards; 2.8 is the current release) for applications that use the poll system call, like Domino. (The default value should be left at 1024 for applications that use the select system call.) The Performance Team tried this and increased the Domino system-wide file descriptor (fd) limit to 64 K. As a result, we now recommend this as one of the required steps in achieving a highly scalable configuration.

To increase the number of file descriptors limit to 64 K, append the following line to the file /etc/system (please see the caveat below):

```
set rlim_fd_max=65536
```

Caution: After doing this, you must reboot your Solaris/SPARC system for the

change to become effective. Be aware, however, that if you make a mistake and enter a very large number, your system may not come back up.

The number of file descriptors required by different server protocols varies. NRPC, for example, is a persistent connection protocol; each user requires about 4 to 5 file descriptors (for each database open and for a connection at the network layer), so 10,000 NRPC users requires about 50 K of file descriptors. IMAP, another persistent connection protocol, requires 3 to 4 file descriptors per user.

Note: As of R5.0.5, the use of AIOThreads is the strategy for effective system usage (that is, the server will indicate at startup if it detects thread pools that are disabled).

Considerably fewer file descriptors are needed for HTTP, which is a connectionless protocol. In fact, Sun recommends 8 K in their NotesBench R5Webmail reports. You can, however, set the value to 64 K, as this is merely a limit and no resources are allocated for it.

See the Sun Solaris and Domino tuning documentation available on the **Lotus IT Central Performance Zone** (click on the Technical Library graphic) and Sun's **docs.sun.com**, which covers additional parameters and provides greater detail about what has worked successfully in other scenarios.

To increase the number of users, spread their data files on multiple drives

Administrators can increase the number of active users and improve disk access performance on their servers by taking advantage of the fact that the files can be distributed across multiple drives. The Performance Team recently did some testing that included evaluation of a mail workload from the browser (a variation of the traditional Webmail workload) on single, two-drive, and four-drive configurations. This testing validated that more users can be added on multidrive systems. (Note that the results presented here apply to all mail configurations, as well as application file configurations.)

First, the team tested a configuration where all mail files existed on the same drive (I:). Testing was successful for 1,000 users and their associated mail files. Attempts were made to increase the number of users (in increments of 500) to determine the upper limits for the number of users on a single-drive system. However, the team was only able to increase the number of users to 1,500; at this point, we observed disk I/O bottlenecks (see the disk I/O metrics in the table below). Attempts to increase the number of users to 2,000 users failed totally, as the system was saturated and could not maintain that user count. In addition, variation in stats like response time increased as **Mail.Waiting** increased. (The user count achieved with this evaluation effort is a number only applying to this test configuration.)

Note: You can also review fields from the Domino console command, sh dbs, as an alternate source of database utilization information. See the *Iris Today* articles **Optimizing server performance: Semaphores Part 1** and **Part 2** for more information on this.

Next, the team distributed the user mail files across two drives—actually on two physical drives on the same controller port. As a result, they were able to increase the number of users, leverage more of available processor power, and accomplish more **Domino.Requests.Per1Minute.Total**. The resulting disk I/O metrics indicated that the access rate was within a reasonable range (based on production level standards).

The team then took the evaluation one step further, by measuring the impact of distributing the data files across four drives. The following tables show

some of the key system metrics from these tests:

1,500 Users -- 1 Drive		
Key Metrics Observed	**Average**	**Maximum**
% Total Processor Time	46.90	84.92
Available MBytes	1,516,109.97	1,692,352.00
Average Disk Queue Length (I:)	1.62	5.53
Average Disk Queue Length (J:)	n/a	n/a
Mail.Waiting	0.94	4.00
Server.Mailboxes	4	
Domino.Config.ActiveThreads. Max	100	
Context Switching	2,298.12	8,591.47
Domino.Requests.Per1Minute. Total	823.28	
Domino.Requests.Per1Minute. Total (average)	6.53	
Mail.Delivered (average)	53.51	

2,000 Users -- 2 Drives		
Key Metrics Observed	**Average**	**Maximum**
% Total Processor Time	53.75	58.61
Available KBytes	934,232.30	972,044.00
Average Disk Queue Length (I:)	2.39	3.93
Average Disk Queue Length (J:)	2.05	3.46
Mail.Waiting	1.13	3.00
Server.Mailboxes	4	
Domino.Config.ActiveThreads. Max	125	
Context Switching	2,402.08	2,905.43
Domino.Requests.Per1Minute. Total	1,124.93	
Domino.Requests.Per1Minute. Total (average)	4.69	
Mail.Delivered (average)	79.92	

2,000 Users -- 4 Drives		
Key Metrics Observed	**Average**	**Maximum**
% Total Processor Time	52.75	55.54
Available KBytes	990,532.10	1,037,996.00
Average Disk Queue Length (I:)	0.95	1.12
Average Disk Queue Length (J:)	0.71	0.80
Mail.Waiting	1.06	3.00
Server.Mailboxes	4	
Domino.Config.ActiveThreads. Max	125	
Context Switching	2,356.45	2,605.77
Domino.Requests.Per1Minute. Total	1,114.92	
Domino.Requests.Per1Minute. Total (average)	4.74	
Mail.Delivered (average)	79.34	

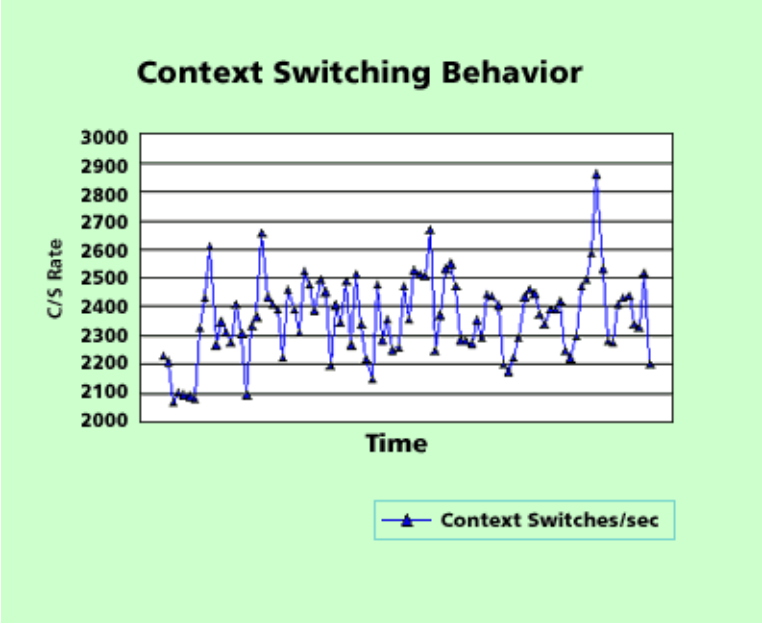
The metrics shown above are those typically used for evaluating system performance, and in these tests, their numbers proved interesting in the following ways:

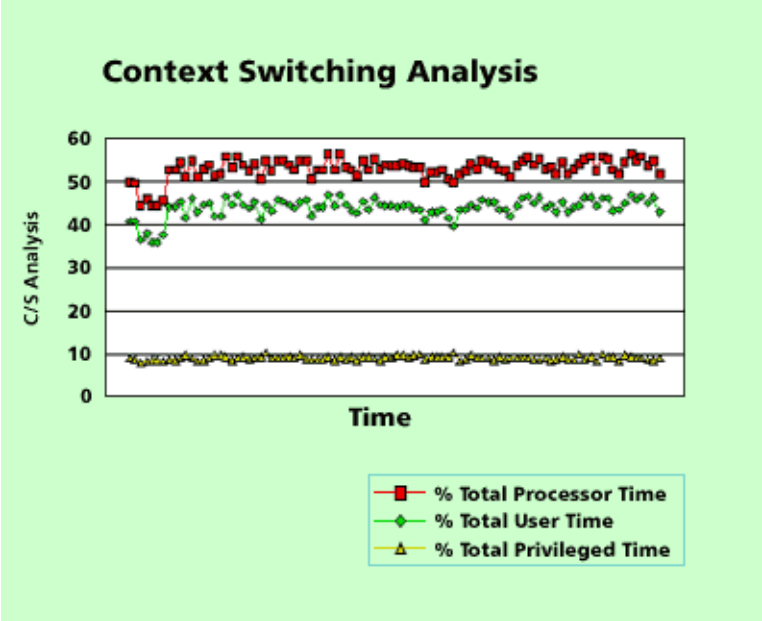
- **Memory utilization** (the reverse interpretation of **Available KBytes**) increased as the number of data drives increased. Most likely memory utilization improved going to a four-drive configuration due to better disk I/O and less memory buffering.
- **HTTP active thread count** (reported through **Domino.Config.ActiveThreads.Max**, and not the same as threadpools for Notes NRPC clients) increased along with the number of simulated users in order to support the additional workload. This is not a problem, however, as there is still plenty of headroom left on the system for additional memory usage.
- R4.x Domino statistics included entries for **Max**, **Allocated**, and **Currently Used** values for HTTP Threads. Domino R5.x-supported statistics now include **Max** and **Allocated**. If needed, the **Max** value will reach the allocated amount, and the value does not decrease. The logic is coded to use the **Max** amount whenever possible, so there is no longer a need to analyze the **Currently Used** relationship to **Max**, as they are now seen as equal.
- **Average disk queue** length for I: drive over the different evaluation configurations indicates that the disk was heavily utilized (in fact, the average queue length increased), even when going from a single drive to a two-drive configuration. More headroom, however, was achieved going to the four-drive configuration. This seems to be the only metric that improved appreciably by going to a four-drive configuration; however, administrators need to understand their own server performance issues and assess any gains to be made by increasing the drive configurations on an individual basis.

- **HTTP server activity** increases going from the 1,500 user workload to the 2,000 user workload; it is measured by the amount of mail delivered and the total requests processed by the HTTP server. This reflects how the Domino server was able to process the additional workload as the disk I/O bottleneck was minimized. If you divide the HTTP request rate (total processed) by the number of users, you can see that the average rate did decrease as the number of users increased from 1,500 to 2,000. This could be an indicator that the number of threads allocated for HTTP processing are all fully utilized and that the additional requests that came in are queued for processing. These HTTP requests include a request to send mail, and you can see how the average **Mail.delivered** rate increased. Using the distributed disk drive strategy resulted in the removal of a bottleneck and enabled the Domino server tasks to operate more efficiently.

- **Context switching** reflects the rate of switches from one thread to another, which can occur for several reasons. When increasing the number of users on the system, it is important to be sure that the context switching rate doesn't approach a danger zone, indicating that the system is spending a lot of time switching among tasks as opposed to executing on behalf of the tasks. This can be confirmed by comparing system time with user time (see the graphs below). The values from the testing, as listed in the previous tables, do not show appreciable growth in the context switching rate.

  The data shown in the graphs was taken from the 2,000 user, two-drive analysis data. The first graph shows the context switching rate on its own; you can see the range of values and number of spikes. The second graph allows you to compare the context switching rate in relation to the processor behaviors. Additionally, processor work effort is broken down in the second graph by total privileged, processor, and user times.





- **E-mail throughput** was acceptable. As described later in this article, the **Mail.Waiting** statistic is a good indicator of whether the router task is keeping up with requests. The average value listed for the **Mail.Waiting** statistic is in a very reasonable range—an average of approximately 1 for all the different configurations—indicating that mail routing is able to deliver the mail as soon as it is received. It is doing a good job of keeping up with the demand; in this case, mail was being generated at approximately 400 mail items per minute.

As you can see from the test results, any additional effort involved in distributing databases is worth it. Not only does it facilitate better drive usage, but it successfully supports as much as 33 percent additional users, making it very worthy of administration and TCO consideration.

Configuring disk storage on the AS/400
By default, the AS/400 spreads data equally among its disk units, which is one of its advantages. This improves system performance by balancing disk utilization across all of the disk arms. You can override this by setting up Auxiliary Storage Pools (ASP). (For details on creating ASPs, see ***OS/400 Hierarchical Storage Management V4R4***, QB3A0Z01.)

By default, Domino servers run within the AS/400's base storage pool,a shared system pool in which many operating system functions and some system jobs are run. The base storage pool contains all of the main storage not allocated to all the other pools in the system. Generally, performance will be best using this default, because the AS/400 can allocate resources where they are needed. There may be some cases, however, where you may want to put one or more servers into their own storage pools. For instance, you may want to place a Domino server in its own storage pool to allow you to specify specific priorities, memory, and so on. If you isolate servers into separate pools, the system can still move resources as needed using the automatic performance tuning feature, but you have more control of what can be moved. (For more information about creating storage pools, see ***Work Management for Version 4 Release 4***, SC41-5306.)

Note: Although not explored here, the MAIL*n*.BOX file is under heavy contention during the mail delivery process. It is worth investigating the benefits of moving the MAIL*n*.BOX onto its own disk drive to avoid disk

contention situations. (See **Part 1** of this series of articles for information about assessing this metric using Domino console commands sh dbs and sh stats.)

Database cache setting: if the Domino system vendors are taking advantage, shouldn't you?

One area in which appreciable performance benefits can be gained is in the adjustment of the NOTES.INI parameter NSF_DbCache_MaxEntries, which allows users to set the number of databases that a server can hold in its database cache at one time. It is tied to the metric **Database.DbCache.MaxEntries**, which shows the maximum number of entries that the cache can hold.

Note: This parameter, as well as the general concept of the Domino server database cache, is described in the **Domino 5 Administration Help** documentation. However, note that some of the information listed here supersedes what is written in the documentation, reflecting additional knowledge that was gained after the manual was published.

The database cache is a defined area of memory that holds key information about a recently opened database file. When the database information is stored in memory, the response time for the end user improves when the database is referenced. This is because key information is retrieved from memory quickly, as compared with the slower medium of disk I/O. As this number increases, more memory is used and more entries are stored in the cache (and there is a greater chance of finding a match in memory).

There's always a trade-off between the total amount of available memory and the amount of memory reserved for a cache. If too much memory is allocated, this may rob other server tasks of critical memory. Hence, the ability to adjust the database cache size allocation correctly and efficiently hovers somewhere between an art and a science. Too many entries reserved in the database cache will cripple the server, as the amount of total available memory is decreased and not available for other types of processing.

The minimum number of entries allowed in the cache at one time is 25. That is actually not a practical number for most larger scale configurations. Most of the time, the value will be calculated using NSF_Buffer_Pool_Size value divided by 300 K and that is the value that is used. If the result is less than 25, the number will appear as 25 (1 percent case). The Domino logic will use the greater value. Depending on the server platform, the maximum amount possible is 10,000. AS/400, RS/6000/AIX, Windows NT, Windows 2000, and Solaris/SPARC all support a maximum of 10,000 entries. If NSF_DbCache_MaxEntries is not defined, it assumes a default value based on the available memory on your system. The NSF_Buffer_Pool_Size is extracted (check out the value **Database.Database.BufferPool.Maximum.Megabytes** found on your Domino server console interface) and used to calculate the default. For this reason, setting values for NSF_Buffer_Pool_Size is discouraged, as you need to make sure that maximum and minimum values are not violated. If you do set it, make sure that it is set properly for your environment. In addition, setting the buffer pool size might have a ripple effect on the values set for other parameters.

The following are some examples from our analysts' systems:

- System Blade, which has 4 GB RAM, had the following values set in a single partition configuration:

  Database.Database.BufferPool.Maximum.Megabytes = 747
  Database.Database.BufferPool.MM.Reads = 0
  Database.Database.BufferPool.MM.Writes = 0

Database.Database.BufferPool.Peak.Megabytes = 7
Database.Database.BufferPool.PerCentReadsInBuffer = 97.12
Database.DbCache.CurrentEntries = 0
Database.DbCache.HighWaterMark = 0
Database.DbCache.Hits = 0
Database.DbCache.InitialDbOpens = 33
Database.DbCache.Lookups = 0
Database.DbCache.MaxEntries = 2241
Database.DbCache.OvercrowdingRejections = 0

- The values were set by default on Hades2K, which has 512 MB RAM, which had the following values set for a single partition configuration:

  Database.Database.BufferPool.Maximum.Megabytes = 171
  .
  .
  .
  Database.DbCache.MaxEntries = 513

In general, you can achieve improved database cache performance by overriding the **Database.DbCache.MaxEntries** value and adjusting it instead to the number of files accessed by your user population. For mail server configurations, you can achieve performance gains by setting the value **Database.DbCache.MaxEntries** to be approximately equal to the number of active users. (This doesn't hold true for an application server, where typically many users are accessing a very small number of files.) As some additional files are generally used, the Server Performance Team usually sets the value for **Database.DbCache.MaxEntries** to be about 50 over the active user count.

For example, if you have 2,700 mail users, each with their own mail file, a value of 3,000 is fine; however, if you have 2,700 application users who all open and use the same application database, the default value is valid. (A word of caution: recommendations made here may result in increased memory requirements. Hence, you should not implement them unless your system has sufficient unused memory.)

You should also monitor the **Database.DbCache.CurrentEntries** metric in your production environment, as this tells you how much of the database cache is actually being used. If you see that the value **Database.DbCache.CurrentEntries** equals **Database.DbCache.MaxEntries ,** and you have enough extra memory available, the NOTES.INI value should be used.

Note: Do not use **Mem.Free** and **Mem.Available** to see if there is extra memory. See **Part 1** of this series of articles for more information about these metrics.

You can also use **Mail.DBCacheEntries** to determine cache usage. Note that the router database cache size defaults to (NSF_BUFFER_POOL_SIZE_ MB * 3). You can monitor **Mail.DBCacheEntries** to see if your system is using the maximum, and compare **Mail.DBCacheHits** with **Mail.DBCacheReads** to see how effectively your cache is being used.

If your server usage profile indicates that your system is more of an application server, where there are many users accessing a smaller number of files, this type of change probably won't yield many benefits. In this case, the databases accessed are already available through the database cache and do not get swapped out. Increasing the database cache size will not improve the access time to your database information.

If you are considering how this setting will work to your advantage (and we

hope that you are), you should also review the reports posted by vendors to the **NotesBench Consortium** Web site, and see how the vendors are making use of this NOTES.INI setting. They will take advantage of every possible setting and adjustment that will maximize response time for their users.

In summary, we do not recommend setting a value for DbCache, as every configuration and Domino server are different. The database cache is memory-dependent and by default, is three times the size of the NSF_Buffer_Pool, if not already set in NOTES.INI. The first thing an administrator should check is the number of data files on the server as well as the amount of server memory. For instance, if there are 2,500 mail files, you could set the NSF_DbCache_MaxEntries to 2,550 (the additional 50 are for system files such as log.nsf, names.nsf, and so on). In reality, there could be fewer such files.

For more information about how the Database Cache operates, see Lotus Customer Support Technote #176417 **How Does the DbCache Work?**

Can Domino failover be supported in your clustered environment?

Suppose you want to take advantage of Domino's failover capability and have successfully enabled the appropriate settings. This doesn't necessarily mean that your current systems and Domino usage pattern can support it. You need to work with the **Server.Availability** statistic, which is used to determine when failover should occur. You also need to have a system with sufficient capacity to support failover in the cluster. Designing for a cluster configuration is a challenge, and even after all the planning and analysis has been done, you need to keep monitoring the configuration to make sure that it is performing as expected. A good starting point is to keep track of the **Server.Availability** values for all the systems defined for failover.

Note: If all the systems in a cluster are heavily utilized (a situation to avoid in any case), it's not going to help to have a failover strategy implemented.

There are some system controls that you can use to fine-tune your implementation. Again, if the original system has not been sized correctly, the fining-tuning will not correct the larger issues. There are the two NOTES.INI variables, Server_Availability_Threshold and Server_Transinfo_Normalize, that are used to ensure that cluster failover is graceful. Graceful failover means that the Domino server can anticipate the cutoff point for rejecting new user connections, before the server maxes out. Such prudent planning ensures that the currently attached users are guaranteed good response times, and the new users trying to attach will also have quality response times.

The first NOTES.INI parameter, Server_Availability_Threshold is set and works directly with the Domino statistic **Server.Availability**. When **Server.Availability** reaches the threshold set in the Server_Availability_Threshold parameter, the server begins rejecting user requests. The threshold may need to be set high (95-97) for the best failover characteristics.

In addition, Server_Transinfo_Normalize will probably need to be set. This allows you to "normalize" the response times experienced by your Domino server and insures that the failover processing will occur only when it's supposed to. This value can be tailored for your environment.

The key message here is that using server performance parameters and statistics not only helps you with up-front planning, but also helps with ongoing monitoring and fine-tuning of your system after implementation. Any configuration changes (such as user workload change, user count change, hardware configuration change, and so on) means that you will need to check

the values. Such due diligence means improved server performance and better reliability for your users.

For more information on clustering, see the *Iris Today* articles **Optimizing server performance: Domino clusters Part 1** and **Part 2**. For more information on load-balancing strategies, see the *Iris Today* article **Workload balancing with Domino clusters**.

The importance of sound application development practices

In the past year, a good deal of information about development techniques that can be used to optimize your Notes/Domino applications has been made available to the user community. The tactics described have appeared in:

- ***The View*** magazine.
- The Lotus White Paper *Maximizing Application and Server Performance in Domino* (Available in the Technical Library on the **Lotus IT Central Performance Zone**).
- Presentations made by the Performance Team and the Web Server Team at Lotusphere, DevCon, and other large conferences, which you can find in the **Iris Sandbox**.
- The IBM Redbook on **Performance Considerations for Domino Applications** (SG24-5602-00).

As an example of the kind of information disseminated through these vehicles, *The View* magazine published an article that featured a comparison of GetNthDocument code and GetNextDocument code. This information has since been referenced numerous times and has made an impact on the Domino developer community. Both practical usage and test analysis indicate that the GetNextDocument code should be used, especially when working with a large number of documents.

This is only one example of an efficient application development practice that has an impact on server performance. Countless others exist. If you are a developer, it behooves you to think about how your development practices affect the servers on which you work. This requires developing some insight into what could be going on "behind the scenes" in your code, such as the implications of what is included in a loop or the type of operation that implies a disk access.

Better coding practices will be also benefit your applications. Application performance isn't solely dependent on LotusScript, the Domino back-end classes, or any of the other components that are invoked as part of your application. Improper coding practices can often be the culprit, and the time spent on analyzing the application to that end is time well spent. Sound application development practices can help server performance by not allowing the applications to be the source of any bottlenecks on the server.

If the server is performing well, there's no a guarantee that the applications will do so; however, this does not necessarily point to the server as the culprit. Analyzing the application might be a better and quicker approach to achieving better performance.

If an application is performing poorly, and you think that it needs some performance tuning, here are some things to look for. If views are slow to open or scroll, check for:

- The volume of data. Perhaps it's time to archive.
- Time/date-sensitive view formulas. Devise alternate strategies.
- ReaderNames or AuthorNames fields on the documents displayed in these views, as these force extra analysis on a per document basis.

If documents are slow to create, save, or open, or even in Read mode, check for:

- An abundance of @Db formulas, especially in keyword fields.
- (Web)QueryOpen or (Web)QuerySave agents.
- Scheduled agents, especially if they execute against newly created or modified documents, or are otherwise scheduled to execute hourly or more frequently. Such agents should have a very fast execution time, and you can pretty easily log enough information to see if this is the case. A good rule of thumb is that no scheduled agent should take longer than a few seconds, unless there are extenuating circumstances.
- Poor logic or coding, or using an inappropriate method to get a collection of documents, which can also lead to poor execution times.

An article in *The View* magazine, "Performance Testing LotusScript Code Using Object-Oriented Design Techniques" by Burke LaShell, describes a technique that developers can use to isolate parts of their applications that might have performance problems, and provides a utility that shows you exactly where your bottlenecks are. You can download the utility from the **article abstract page**.

Client-level decisions do have an impact upon server performance

As mentioned in **Part 1** of this series, client-level activities can have a big impact on server-level performance. In doing the research for this series, the Server Performance Team came across several client issues that Domino server administrators can address in their own installations to help increase their server performance.

One such issue is that of image resource storage. Designers should take advantage of image resource storage within the database, which was introduced in R5. Although there are other methods for using images in a database, the use of image resources is the most efficient as it requires that you maintain images in only one location, and refer to that location wherever you want the image to be used. If there are any changes to the image, changing and refreshing the source file distributes the changes to wherever the image is referenced. Based on the experiences of the Lotus Professional Services Technology Team, designers and developers could take better advantage of this feature from both a performance and administration viewpoint.

Understand more about the mail delivery rate

There is a long list of Domino server statistics, and their meaning is often unclear. There are some hidden jewels of information there though, that can help you better understand the Domino server. With this knowledge in hand, you can better appreciate the information that's been generated about certain functional areas and take proactive steps as necessary. This section deals specifically with NRPC Mail Delivery. (Note that the router operates for all types of mail delivery, such as IMAP and WebMail, and this information can be leveraged for alternate mail protocols.)

From a performance viewpoint, the statistics described below are the ones to concentrate on for Mail Delivery analysis. Note that the scope of the analysis is restricted to those stats found in the Mail section. For the big picture of mail processing, the Platform and Database stats should also be included in the analysis.

Note: It may take a few minutes for some of the Domino stats listed below to appear at initial server startup, as Mail Routing gets going. Also, if there is no activity for a given feature area, the associated Domino stat will probably not appear in the stat list. For example, if there is no SMTP delivery performed, the SMTP stats will not appear in the Domino stat list.

Mail.Waiting

This is the count of mail items waiting to be delivered. Specifically, it is the

count of mail items that have been seen by the router and then dispatched to a transfer or delivery destination. This value is cumulative for all MAIL.BOX files.

This metric has proven to be the most beneficial in tuning analysis efforts in the benchmarking environment. In performance analysis, an increase in this value is an indication that the Mail Router might not be keeping up with the incoming requests for delivery. However, this is not necessarily true in the customer production environment, as the sizes of the mail items are very different and the time of day (which includes peak periods) may factor into the analysis of how the mail router is performing. More importantly, if the destination or next server hop is not available, the mail item is queued on the local server to wait—a situation that is not typically tested for in a performance analysis environment.

Mail.WaitingForDNS is an indicator as to whether DNS is performing well. The Domino router logic looks up the target domain in DNS to acquire the host names to use for the SMTP connection. If DNS is down or performing poorly, this value grows. This is an indication of a bottleneck in your network configuration, and further troubleshooting should be performed at this point.

Mail.WaitingRecipients is the count of the total number of recipients for each mail item held on the Mail.Waiting queue. Note that mail may have been delivered to some of those recipients. For example, if a message came in with ten recipients, and nine were delivered, but one was to be delivered to someone on a downed server, the Mail.WaitingRecipient count would still be ten.

Mail.TotalPending
This value (new for Domino Release 5.0.4) reflects the number of mail messages currently resident in all active MAIL.BOX files. If multiple MAIL.BOX files are enabled on the server, this value includes all messages in all of the currently enabled mailn.box databases. This count includes all Dead (non-deliverable) and Held mail, as well as mail that is pending delivery.

Mail.TotalPending is updated on a 5-minute interval (to minimize impacts on the server's performance), regardless of other processes that are updated instantaneously. The server process maintains and monitors this statistic. Since the stat is maintained by the server process and not the router, the **Mail.TotalPending** value will still grow whether or not the router is running. Practically, **Mail.TotalPending** keeps track of the total number of messages in all MAIL.BOX files. As messages are processed by the router and removed from MAIL*n*.box, the statistic is decremented.

Mail.Delivered
The **Mail.Delivered** statistic is a count of the number of recipients whose inbox was updated on the local server. For example, a message with three recipients all being delivered on the local server would count, for this statistic, as three.

Mail.TotalRouted
This indicates the number of recipients for whom messages were transferred or delivered. A message with three recipients physically transferred once to a destination would count as three.

This metric is also broken down by protocol:

Mail.TotalRouted = Mail.TotalRouted.NRPC + Mail.TotalRouted.SMTP

Mail.Deliveries
This is an indication of how many trips to the mail file were taken for a given user. It's equivalent to number of times a user's mail file was opened and

closed. It will always be less than, or equal to, **Mail.Delivered**. For example, if there are two messages pending for a user, and the router opens the mail file once, delivers both messages, and then closes the mail file, that counts as two *delivered* and one *delivery*.

Mail.Transferred
This is the total number of messages transferred from the Domino server to a given destination (or the next hop). A single message with three recipients, all going to the same destination, would count as one message transferred. If that same message had two recipients going to one destination and one going to another, it would be counted as two messages transferred.

This metric is also broken down by protocol:

Mail.Transferred = Mail.Transferred.NRPC + Mail.Transferred.SMTP

The metric Mail.Transferred.TotalKBT is also calculated at this time, which keeps track of the total KB transferred to another destination.

Mail.Hold
This is a stat reflecting an administrator-requested action. It's a count of the messages with RoutingState held. For configurations where the administrator has selected "hold undeliverable mail," this should be the count of the messages in that state.

Other mail statistics
Other Domino stats that are useful to track from an administration viewpoint (and whose names are more self explanatory) include **Mail.AverageDeliverTime**, **Mail.MaximumDeliverTime**, and **Mail.MaximumSizeDelivered**.

Examples
Here are some examples to analyze:

		Mail.Waiting	Mail.WaitingFor DNS	Mail.Waiting Recipients
01/28/00	12:54:31 PM	3	0	9
01/28/00	12:55:31 PM	0	0	0
01/28/00	12:56:31 PM	4	0	12
01/28/00	12:57:31 PM	3	0	5
01/28/00	12:58:31 PM	2	0	6
01/28/00	12:59:31 PM	2	0	6
01/28/00	01:00:31 PM	1	0	1

		Mail. Hold	Mail. Delivered	Mail. Deliveries	Mail.Total Routed
01/28/00	12:54:31 PM	0	21972	21956	21972
01/28/00	12:55:31 PM	0	22248	22222	22248

01/28/00	12:56:31 PM	0	22511	22486	22511
01/28/00	12:57:31 PM	0	22750	22725	22750
01/28/00	12:58:31 PM	0	23025	23003	23025
01/28/00	12:59:31 PM	0	23252	23231	23252
01/28/00	01:00:31 PM	0	23503	23477	23503

See the Lotus Customer Support Technote #145928 **How to Interpret Notes R4 Mail Routing Statistics** for more information about some of the mail stats discussed above.

When planning for that next server

The suggestion to plan ahead for the next server isn't so much a tip or a clearing up of a misconception as it is a plug for our Performance Product Managers, who are out there gathering information about various platforms. We have already shared much of this information and will continue to include it in our presentations and Notes.net articles and postings. If you are planning to invest in a new or additional server, you should check out the sources and links listed below. This list is an excellent summary of cross-platform tools and information; it was developed in response to Lotusphere feedback about the difficulty of finding this kind of information.

Note: The list does not include links to individual vendors; see the **NotesBench Consortium** Web site for links to specific vendor sites.

Platform	Capacity Planning Tools and White Papers
IBM Netfinity NT	• **IBM Netfinity sizing tool**
Compaq NT	• **Compaq ProLiant Sizer for Domino**
Dell NT	• See your Dell representative
Sun Solaris/UNIX	• Internal tool - See your Sun representative • **Lotus Domino R5 for Sun Solaris**, IBM Redbook (SG24-5969-00)
RS6000/AIX	• Internal tool - See your IBM AIX representative • **RS/6000 resources** • **Resource Tuning of Lotus Domino on AIX: Quick Reference Guide**
AS400	• Lotus Domino for the AS400 **Performance page** includes links to sizing information, including the Workload Estimator for the AS400 • White papers: **Evaluating Appropriate workloads for the AS400e Dedicated Server for Domino**, **AS/400 Performance Tuning for Domino**, **Domino for AS/400 Performance Tuning**
S/390	• Internal tool - See your S390 representative

- **Domino for S390 Performance Overview** page
- **Lotus Domino for S/390 Performance Tuning and Capacity Planning**, IBM Redbook (SG24-5149-01)

Conclusion

Domino administrators have an ongoing challenge when it comes to optimizing the performance of their servers. Not only do they need to understand the information that their servers provide, but they also need to apply that information properly for their particular configurations. Administrators also have the opportunity to be proactive when it comes to performance; to take all this information and use it to fine-tune their servers to achieve even greater performance. On a higher level, administrators can also takes steps to ensure that application development practices in their organizations enhance server performance rather than hinder it.

For additional information on server performance, see the **NotesBench Consortium** and the **Lotus IT Central Performance Zone**.

If you have benefitted from any of the tips or information presented in this series, the Domino Server Performance Team would like to hear from you. Use the **article feedback form** to send us your success stories.

ACKNOWLEDGEMENTS

Special thanks to the various individuals and teams that have contributed their insights and datapoints to make this article more successful. In particular, the Domino Server Performance Team, James Grigsby, Maria Krylova, Lotus Professional Services Technology Team, Lotus Support, and the Mail Routing Team have all made valuable contributions to the points and the content in these articles. We would also like to acknowledge Louis Bradbard and Richard Kanosky, of the Server Performance Team, for the data generation and analysis they provided for this article.

ABOUT THE AUTHORS

Carol Zimmet started working at Iris in 1994. She is the co-lead on the Domino Performance Team, and responsible for evaluating performance and performance tool development. Carol continues to search for the one-step solution to everyone's performance problems. She is also interested in a "white box" approach towards improving the quality of the product. Carol enjoys bicycling with her kids and playing racquetball. She has a longing to return to stained glass!

Amy E. Smith is a principal user assistance writer for Lotus. She writes and maintains functional specs for Domino and Notes. She also is a member of the Notes UA Web team.

