Optimizing server performance: HTTP Threads settings

by George Demetriou

[Editor's note: This article resides in "Iris Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino and Notes.]

So, you've got your Domino server all set up to "work the Web," but want to make sure that you get the most out of the system? You may not realize that you can adjust the number of HTTP threads running on your server, and improve both the server's response time and resource utilization.
In this article, we'll take an in-depth look at a performance analysis of Domino Web server resource utilization on Windows NT. The test shows the impact of changing the HTTP threads setting on server performance. We'll start by defining what HTTP threads are, then describe the test methodology and test data, and finally summarize what the results mean to you. This can help you decide how you want to set up your environment in the future.

For background information on how we conduct performance analyses here at Lotus/Iris, or an introduction to the tools we use, see "**Optimizing server performance: Port encryption & Buffer Pools**." To read more recommendations for improving server performance, see the "**Top 10 ways you can improve server performance**."

What are HTTP threads?

HTTP threads are threads of execution for handling incoming HTTP requests. To specify the number of threads that you want active on your Domino server, use the "Number of active threads" field in the HTTP section of the Server document in the Public Address Book. The default setting is 40. When the HTTP server task initializes on the Domino server, the defined threads are created and occupy approximately 20-40Kb of memory each. These threads are fixed in number until you change the value in the Server document, and then restart the HTTP task.

Our expectation for this performance analysis of a Domino server functioning as an HTTP-based messaging server was that the HTTP active threads setting should be equal to the number of Web users on a particular Domino server. For example, if you are anticipating 200 Web users to use a Domino server, you might assume that you should set the HTTP active threads to 200. However, as you will see from our test results, this is *not* the case.

Test methodology and test data

To run the test scenarios, we set up one client that could simulate Web browser users running the new **NotesBench** WebMail workload with the following configuration:

- CPUs: One Pentium II processor
- Memory: 256MB RAM
- OS: Windows NT 4.0 Workstation
- Notes: Release 5 (based on Beta 1)
- NotesBench WebMail workload (available with Release 5 of NotesBench)

We set up a Domino server with the following configuration:

- CPUs: Two Intel Pentium II/300MHz with 512K Level 2 Cache
- Memory: 512MB RAM
- Single SCSI controller
- Three hardware arrays created across six disk drives
- Hard Drives:

  --Logical C: One RAID0 7200rpm drive for OS

  --Logical D: One RAID0 7200rpm drive for page file and Domino executables

  --Logical F: One RAID5 7200rpm with four drives for user mail files, logs, and mail.box (total 36GB storage for \data directory)

- OS: Windows NT Server 4.0, Service Pack 3
- Domino: Release 4.62 for Windows NT

In particular, we wanted to test the relative impact (the number of users, the response time, and the resource utilization) when varying the following HTTP threads settings: 10, 25, 50, 100, and 200. We applied each setting to a NotesBench WebMail multirun test with 25, 50, 100, and 200 users. This test scenario compared average user response times, probe response time, system CPU utilization, memory used, and disk utilization at various loads. We ran each test for approximately 60 minutes in a steady state, with a ramp-up period of 5 minutes.

The workload we used for all the tests is the new NotesBench WebMail workload, which will be available when R5 ships. This workload enables each simulated user to access their respective mail file, built using the R4.6 Mail - Combined template (mailc46.ntf), via the HTTP protocol. Each user iteratively executes a 15-minute script that consists of:

1. Preparing and sending a 10K message to three recipients dynamically selected from the server's directory every 15 minutes.

2. Reading the Inbox and the first five Inbox documents, and deleting the first message.

All messages are sent to and received by other simulated WebMail users on the same Domino server.

In addition, the WebMail workload requires the following NOTES.INI settings on the client:

- ThreadStagger = 5 (seconds)
- NormalMessageSize=10000
- NumMessageRecipients=3

The ThreadStagger setting specifies when each user logon begins, so in this case, each user logon begins at five seconds apart. This setting helps the server ramp up smoothly, without having connection timeouts during the ramp-up phase. The NormalMessageSize and NumMessageRecipients settings specify the size of the message (10K) and the number of recipients (three) that are used in the 15-minute WebMail script.

To see the results of these tests, see the sidebar "**HTTP Threads Settings Test Results**." For our conclusions, see the following section, "What did we find out?"

What did we find out?

Our basic discovery was that *more* HTTP threads is *NOT* necessarily better. In fact, as indicated by the data and graphs, there is a degradation in performance *and an increase in memory consumption,* if there are more HTTP threads defined than are needed. *Therefore, you should only define the minimum number of HTTP threads necessary for your server's load.*

The best way to determine the optimal HTTP threads setting for a given user load is to first run with the HTTP threads setting approximately equal to the user load. Then, at the server console, check the peak number of HTTP threads used by typing the following:

"show statistic domino.threads.active.peak"

Since this value indicates the maximum number of HTTP threads in use, it is much more appropriate to use this value for the HTTP threads setting in the Server document.

If you've just installed a Domino server and you have a general idea of how many Web mail users it will be supporting, then a good starting value for the number of HTTP threads setting is 10% of the number of Web users.

For example, if you anticipate 200 Web mail users, a suitable initial setting for the HTTP threads is 20. Once you have your users running, you can "fine tune" the setting based on the "domino.threads.active.peak" statistic. If you constantly see that the Peak value is the same as the value you defined, you should increase the value of defined threads in the HTTP section of your Server document.

What if you don't use your server for HTTP-based messaging? You can still monitor the "domino.threads.active.peak" statistic to determine if you've started more threads than you need. By reducing the number of threads, you can decrease the amount of memory used by the HTTP server and this memory will be available for other activity on the server.

ABOUT THE AUTHOR

George Demetriou started working at Iris in 1997 from Eastman Software. He is a Performance Engineer who has spent his time working on Domino Web server performance measurement and evaluation.

HTTP Threads Settings Test Results (sidebar)

These are the results we found when testing how the HTTP threads setting affects server response time and resource utilization. We measured the impact of changing the HTTP threads setting by monitoring Domino transactions (NotesMarks), response time, CPU utilization, memory utilization, and disk response time.

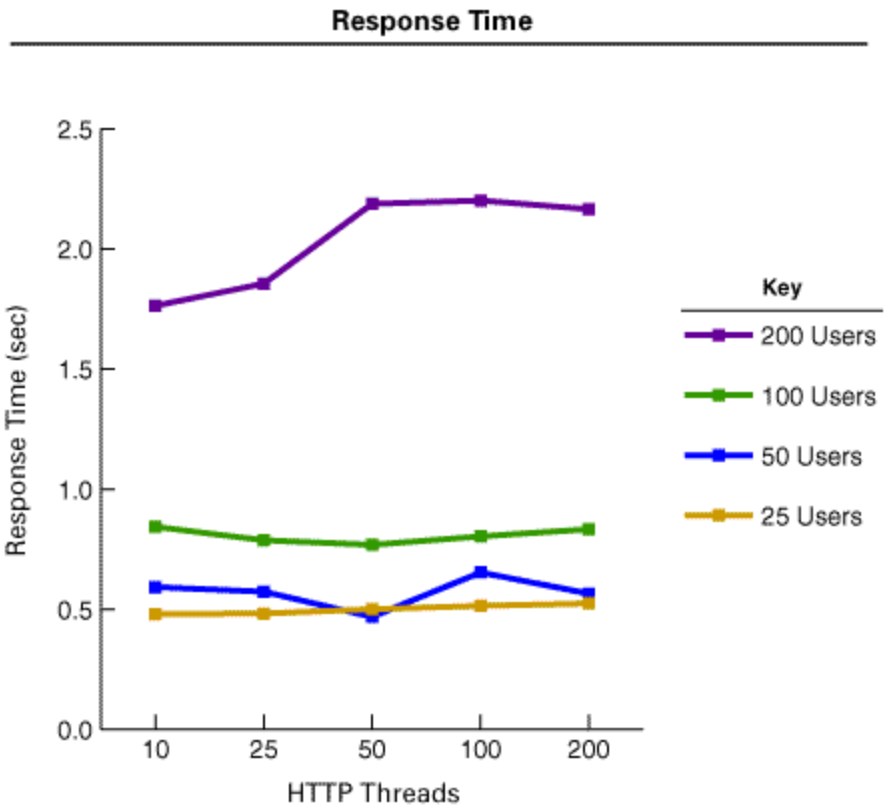
WebMail user response time (seconds)

This chart displays the average WebMail user response time that results when we vary the number of WebMail users as well as when we vary the number of HTTP threads. We can view the effect of increased user load for a given HTTP thread setting by reading the appropriate column. Similarly, we can view the effect of increased HTTP thread settings for a given user load by reading the appropriate row.

As expected, for a given number of HTTP threads, if we increase the user load, response time will increase (that is, worsen). For example, if you read the first column (10 HTTP Threads), the response time steadily increases from 0.481 to 1.764 seconds as we increase the user load from 25 to 200 users.

However, for a given WebMail user load (read across a row), the response time is not improved, and in some cases, worsens, if you increase the number of HTTP threads. For example, consider the last row of the chart (200 WebMail users). As you read across the row, the response time worsens, from 1.764 seconds at 10 HTTP threads to 2.166 seconds at 200 HTTP threads.

WebMail Users	10 HTTP Threads	25 HTTP Threads	50 HTTP Threads	100 HTTP Threads	200 HTTP Threads
25	0.481	0.483	0.501	0.515	0.525
50	0.593	0.574	0.469	0.654	0.566
100	0.844	0.788	0.769	0.804	0.834
200	1.764	1.856	2.188	2.202	2.166

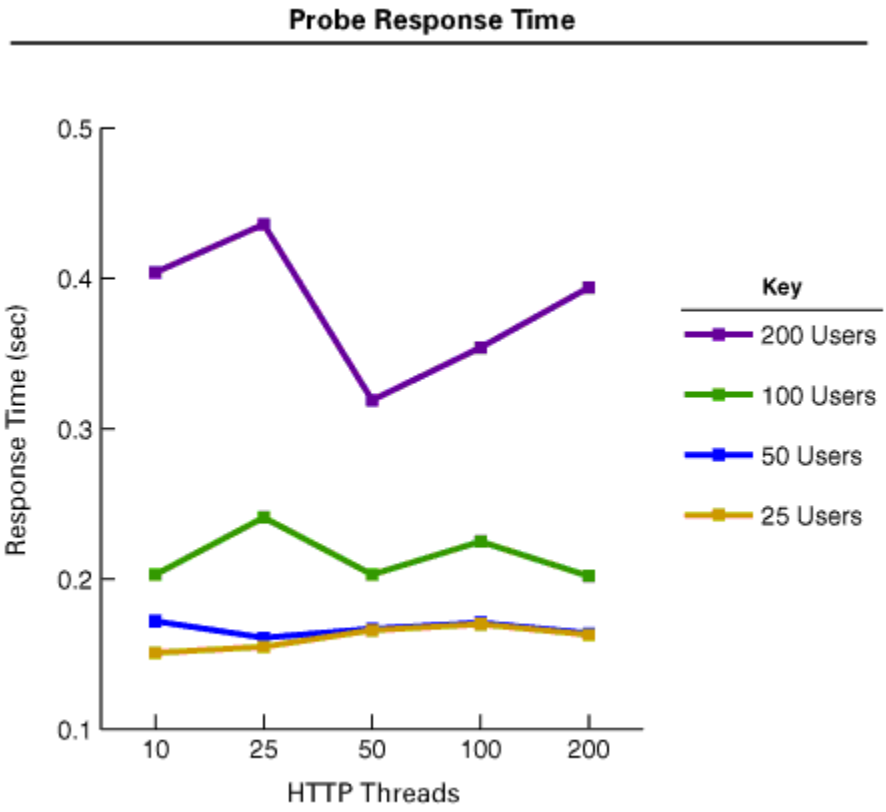

Response Time
Response Time (sec)
2.5
2.0
1.5
1.0
0.5
0.0
10
25
50
100
200
HTTP Threads
Key
200 Users
100 Users
50 Users
25 Users

NotesBench probe response time (seconds)

This chart displays the average NotesBench probe (operations over the HTTP protocol) response time that results when we vary the number of WebMail users as well as when we vary the number of HTTP threads. The probe response time differs from the user response time in that it represents the average response time for a specific request, in this case, opening the Inbox. The user response time is the average time for all the requests that comprise the WebMail workload (opening the Inbox, sending mail, and deleting mail).

Again, as expected for a given number of HTTP threads, if we increase the user load, response time will increase (that is, worsen). Also, like the WebMail user response time behavior, the probe response time is not improved when you increase the number of HTTP threads.

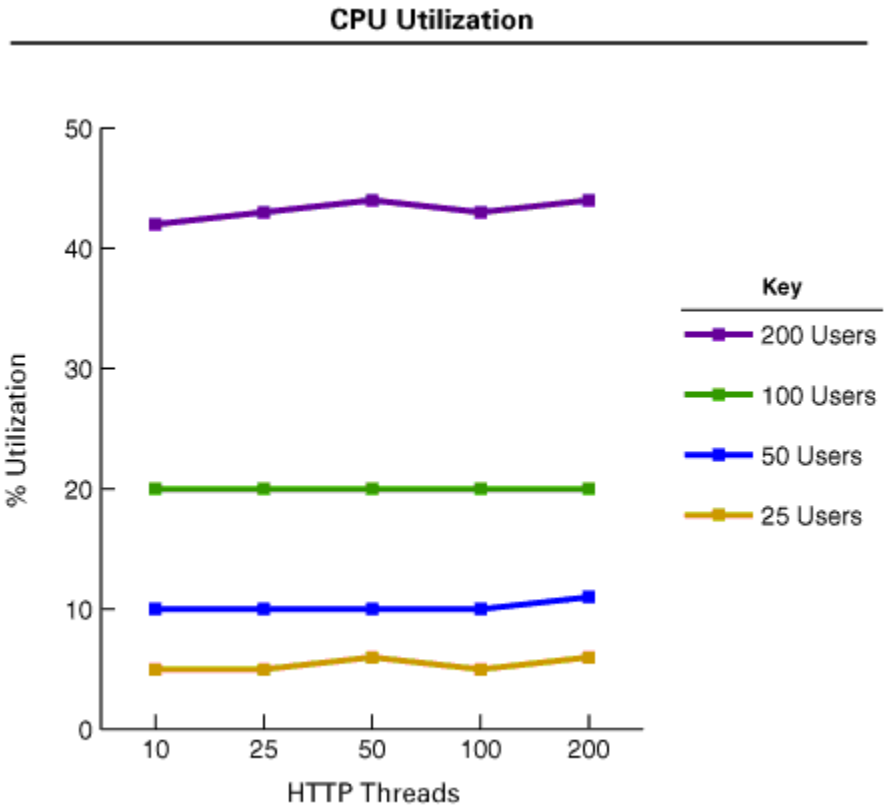
WebMail Users	10 HTTP Threads	25 HTTP Threads	50 HTTP Threads	100 HTTP Threads	200 HTTP Threads
25	0.151	0.155	0.166	0.170	0.163
50	0.172	0.161	0.167	0.171	0.164
100	0.203	0.241	0.203	0.225	0.202
200	0.404	0.436	0.319	0.354	0.394



Server CPU utilization

This chart displays the CPU utilization on the server when we vary the user load as well as when we vary the number of HTTP threads. As you can see, for a given user load, the CPU utilization is not affected by an increase in HTTP threads.

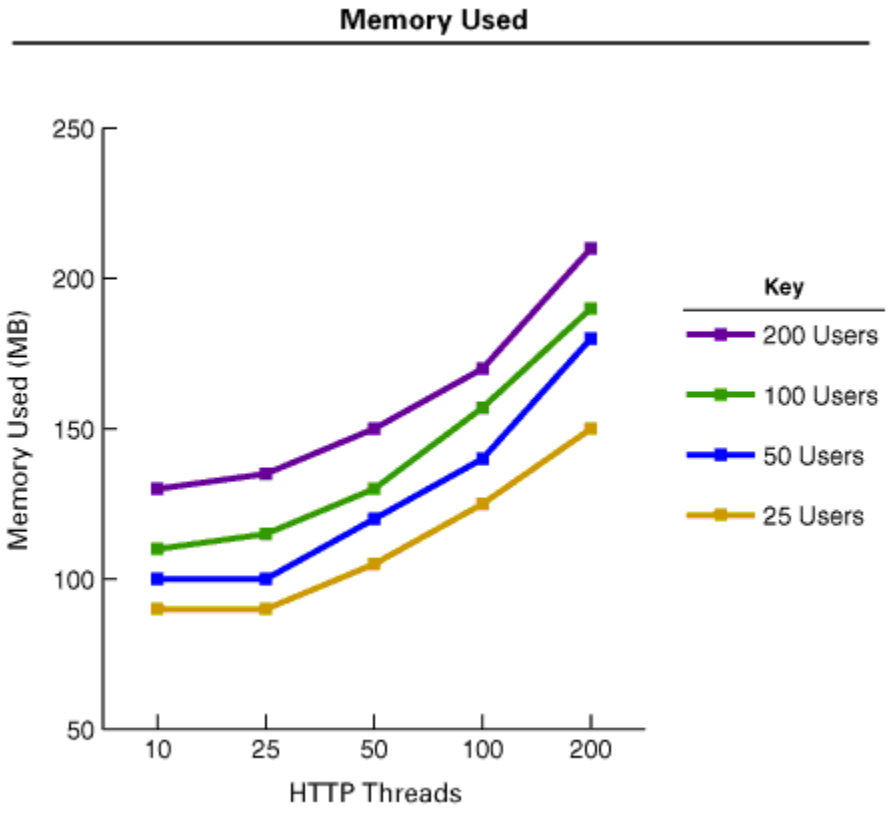
WebMail Users	10 HTTP Threads	25 HTTP Threads	50 HTTP Threads	100 HTTP Threads	200 HTTP Threads
25	5	5	6	5	6
50	10	10	10	10	11
100	20	20	20	20	20
200	42	43	44	43	44



Memory used (committed bytes in MB)

This chart displays the memory used on the server when we vary the user load as well as when we vary the number of HTTP threads. As you can see, for a given user load, the memory used increases when the HTTP threads setting is increased. Since we obtain no performance improvements with the increased HTTP threads, this additional memory utilization is essentially "wasted."

WebMail Users	10 HTTP Threads	25 HTTP Threads	50 HTTP Threads	100 HTTP Threads	200 HTTP Threads
25	90	90	105	125	150
50	100	100	120	140	180
100	110	115	130	157	190
200	130	135	150	170	210



Logical average disk queue length

This chart displays the average disk queue length of the server volume containing the \data directory when we vary the user load as well as when we vary the number of HTTP threads. Disk queue length is the number of both read and write requests that were queued. The larger the value, the more I/O-bound the server is. In this case, we see that increasing the HTTP threads results in an increase in the average disk queue length and therefore, increased I/O overhead.

WebMail Users	10 HTTP Threads	25 HTTP Threads	50 HTTP Threads	100 HTTP Threads	200 HTTP Threads
25	0.04	0.05	0.05	0.06	0.07
50	0.10	0.09	0.08	0.08	0.10
100	0.20	0.18	0.14	0.16	0.16
200	0.40	0.35	0.40	0.44	0.62