Improving Web site performance

by
Susan Florio Barber,
Shawn Harris,
and Carol Zimmet

Level: Beginner
Works with: Domino 5.0
Updated: 08/02/99

Inside this article:

Related links:
Top 10 ways to improve server performance

Optimizing server performance: I/O subsystems

Optimizing server performance: HTTP Threads settings

Expanded Command Caching in Domino 4.61

Optimizing server performance: Transaction logging

Get the PDF:

[Editor's note: This article is the next in our series exposing how the Notes.net site is run. This time, learn about the design of our Lotusphere survey, and how the results will help shape the Notes.net site.]

Improving the performance of your Domino servers is an art. There are no hard and fast rules. Until you understand the intricacies of server performance monitoring and analysis, you can't really understand how to change things on your server to improve performance. This is further complicated because everyone's server environment is different. Even when the experts here at Iris and Lotus come up with a **list of top ways to improve server performance**, they can't guarantee that these things will work for you.

No one understands this better than we do here at Notes.net. About six months ago, we noticed a drop in the performance of the site. At that time, it took several minutes to load a page. This prompted us to tap into the expertise of Carol Zimmet, a server performance expert here at Iris, and our own network administrator, **Shawn Harris**, to try to find ways to improve the overall performance of Notes.net. We went into the project with the same expectations that some of you probably have -- just tell us what to do to improve the performance of our site and we'll do it. We quickly found that it's not that simple. We now know a lot more about monitoring our servers and analyzing the results of what we monitored, and we'd like to share this experience with you to help you improve your own site performance.

This article first provides you with an overview of our server environment and the configurations of our servers. It then describes the various tools and monitoring techniques we used to collect performance data. Next, the article looks at the changes we've implemented to help the performance of our site. Finally, we analyze the impact of the changes and describe possible actions we can take in the future to further improve performance. We hope that this will provide you with some insights into the different techniques you can use to analyze your own Domino Web site.

Our performance goals

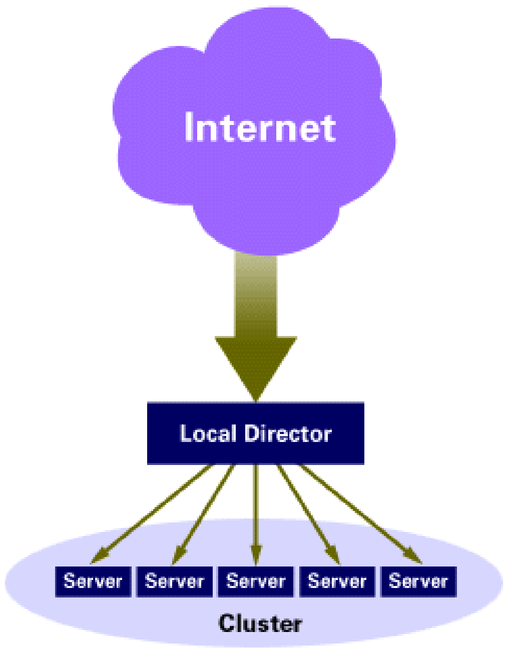
When we began our performance monitoring in January 1999, our main goal was to improve the performance of the Notes.net site. We suspected that the site was slow, in part, because of the increased number of users coming to the site. At that time, we were hosting the Beta downloads of R5. Plus, the general popularity of the site had been steadily increasing. In fact, our log statistics showed that the number of hits to Notes.net had increased from 2.7 million per month in the Spring of 1998, to approximately 10 million per month just one year later. We realized that we could probably improve performance by upgrading our server hardware configurations to better handle this level of traffic.

Our secondary goal was to begin monitoring our servers to collect data about the current level of performance so that we could measure any future change in performance in relation to this baseline data. Also, because we use the Domino servers developed here at Iris, we can easily give the information we gather back to the Domino development team. They can then make the appropriate improvements to the Domino server itself. More importantly, we can share the information with all of you.

Of course, monitoring the performance of Notes.net is an ongoing process. We believe in Domino and we stand behind this belief by using our own product. In fact, how long and how well the newest R5 Beta build ran on our site was one of the criteria used to evaluate how close we were to shipping the final release. You can be sure that we look at whether a new build helps the performance of our site, because our site depends on it.

Our environment

The following illustration shows a general map of the Notes.net environment:



Notice that our site is made up of five servers in a Domino cluster, which sits behind our load balancer, **Cisco Systems** Local Director. When you access Notes.net from the Internet, your request first goes to our Local Director. The Local Director keeps track of the load on each of the servers, so it can direct your request to the server with the least load. We also use a "sticky" setting on the Local Director so that it directs multiple requests from the same client to the same server. This means that you can experience a consistent and constant connection to the same server, which is useful when you are submitting forms to an application. For end-users accessing the site, the Local Director makes our servers appear as a single virtual server with one IP address, and all load-balancing is transparent. (For more information on Local Director and our clustering techniques, see "**Notes.net Exposed: Using Domino clusters for your Web site**.")

Each of our six servers has a slightly different configuration, because different operating systems have different requirements, as do different Domino releases. We also need to make frequent Domino upgrades to our systems. For example, during the time that we monitored the performance of our servers, we moved from R4.6 to the R5 Beta builds and finally to R5 Gold. It is challenging to monitor the performance of your servers using a Beta release, because Beta releases often have a lot of debugging code activated that slows down the server. Performance naturally improves when you move from a Beta release to the Gold release.

The following table shows the configurations of our servers when we started performance monitoring:

Server	Operating System	CPU	RAM (MB)	Disk Structure	Domino
Server 1	WinNT 4.0	Dual 300MHz Dell PowerEdge 4200	512	C: OS, Page file D: Notes Program/Data/Web logs	Beta R5

Server 2	WinNT 4.0	Dual 300MHz Dell PowerEdge 4200	512	C: OS, Page file D: Notes Program/Data/Web logs	R4.6.2
Server 3	WinNT 4.0	Dual 300MHz Dell PowerEdge 4300	512	C: OS, Page file D: Notes Program/Data/Web logs	R4.6.2
Server 4	AIX 4.3.1	IBM F40 2X	700	/opt: OS, Swap file, and Notes Program /local1: Notes Data/Weblogs	R4.6.2
Server 5	WinNT 4.0	Dual 300MHz Dell PowerEdge 4200	512	C: OS, Page file D: Notes Program/Data/Web logs	R4.6.2

The tools we used

When we began monitoring the performance of our servers, we used two tools: the Windows NT Performance Monitor and an internal testing tool with a probe workload. The following sections contain more details on these tools.

To monitor UNIX servers, you can use the tools SAR, VMStat, IOStat, NetStat, or PerfMeter. You can also monitor AIX machines using a tool called *Performance Toolbox*. To find out more about Performance Toolbox, see the IBM Redbook **RS/6000 Performance Tools in Focus**.

Performance Monitor

You can use the Windows NT Performance Monitor to monitor both operating system statistics and Domino statistics. The way it works is that you add the Domino server as a counter within the Performance Monitor. When you run the Domino Install program, you can select to install the Performance Monitor components. If you installed the Domino server without selecting to install Performance Monitor, see the sidebar "**Installing and viewing Domino as a Performance Monitor counter**."

The Performance Monitor lists all numerical Domino server statistics, including those generated by add-in programs. You can choose specific statistics to appear in a report or a chart for analysis. You can also use the Performance Monitor to view the statistics of a remote server. For complete information on using the Performance Monitor, see the Windows NT documentation.

We ran the Performance Monitor in logging mode and collected data at 15-minute intervals. To do this, choose View - Log, and then Edit - Add To Log. We recommend that you add the following objects for logging, because they can be helpful when it is time to analyze your data:

- Cache
- Logical Disk
- Lotus Notes
- Memory
- Paging File
- Process
- System

Then, set up your log options by choosing Options - Log. Specify a Periodic Update interval and click Start Log.

After you've collected data in your logs, you can use the Performance Monitor chart option to view and subset the data. To do this, choose View - Chart; then choose Options - Data From and select your log file. Select the objects, along with the counter and instances that you want to chart. We looked at various system statistics, such as the processor usage, the percentage of disk times (this is the amount of time that the disk was actively servicing requests to read and write data), the average disk queue length, and the amount of memory available over a certain period of time for each server.

We also collected Domino server statistics, including the Domino Web server statistics, by using the Lotus Notes object. The Domino Web server statistics begin with the prefix *Domino.*, and include HTTP Server information. Specifically, we collected information on the number of HTTP requests in a given period of time and the number of active HTTP threads. These statistics helped us build a baseline of statistical information. (You can also gather this type of information by using Domino Statistics and Events. For more information on Domino Statistics and Events, see the **Domino 5 Administration Help**.)

Note: To initialize the Logical Disk object, you need to run the *Diskperf* tool and restart the system. You can run the Diskperf tool using the -Y switch (for example, diskperf -Y \\servername) from the command line. This sets the system to start disk performance counters when the system is restarted. If you are using striped disks, you need to add the -E switch (for example, diskperf -YE \\servername). (On an array of disks, *striping* is a technique for improving disk drive speed. Each file that writes into a striped array spreads, or stripes, over several drives.)

The Probe workload
We used a Probe workload executed by our internal testing tool to measure the response time of our site. The Probe workload opens a page on our site every specified amount of time (in our case, approximately once per minute). We selected our home page, **http://www.notes.net/welcome.nsf**, as the page that the workload should open because that's the first page that users access on the site. This page also has many objects that need to load and therefore, is a page that gives a good indication of a real workload request.

The internal testing tool records the time it takes to open and fully display a page. While using the Probe workload on Notes.net, we realized that we needed to make changes to the tool to make sure that it accurately records the time it takes to open a page. In addition, we enhanced the tool to better distinguish between the initial connection time and the time it takes to retrieve all the objects on the page.

From January until the end of April 1999, we gathered the information that will serve as a benchmark for future Notes.net performance analysis. We created a weekly Lotus 1-2-3 spreadsheet that contained an analysis of the information gathered that week. One thing we compared was the overall site response time with the response time when all the Notes.net servers were active, versus the response time when one or more of the servers was not active. (To see these results, see the sidebar "**Overall response time results**.") We calculated adjusted information by finding out the time during the week each HTTP server was active, and analyzed the information from each of these different perspectives. We also analyzed the difference in the probe response time during the day as compared to night, and during weekdays as compared to weekends.

Once we collected some of this baseline data using the Performance Monitor and the Probe workload, we also tried varying different settings to see if we could detect an improvement in performance. Varying the settings led to interesting observations. We also measured the impact of various changes on the servers, such as hardware configuration changes and Domino upgrades.

The changes we made

The following table summarizes all the changes that we made in our environment. In general, we first made a change on just one server, and watched the effects of the change for a few days. Then, we made the same change on the rest of the servers. To make the table easier to read, we've only included the dates of the initial changes that we made. We've left off the dates of when we implemented the changes on the other servers, and the many upgrades to new Beta builds of R5.

Date	Action
1/22/99	Started Performance Monitor logging
1/26/99	Modified server's disk configuration
1/27/99	Upgraded server to Beta build of R5
2/12/99	Increased the HTTP active threads setting to 80 on all servers
	Added the NOTES.INI setting, DominoAnalyzeFormulas=1
	Looked into whether to raise the Domino cache levels
	Enabled Domino R5 transaction logging
3/3/99	Increased the HTTP active threads setting to 120
3/24/99	Upgraded hardware (faster processors, more memory, disk configuration)
4/5/99	Upgraded server to R5 Gold

The next sections describe these changes in more detail.

Server configuration changes
As mentioned in the section on our performance goals, we suspected that we needed to upgrade our server hardware configurations to better handle the increased traffic on our site. However, we first verified that we did not have a network throughput problem (by using the network sniffer tool, NetXRay, by **Network Associates**). Our results showed an average network utilization of approximately one percent from a given server.

After analyzing our performance statistics in the Performance Monitor, we discovered that our disk I/O was a bottleneck. As a result, we decided to completely reconfigure the disk structures on our servers. (To learn more about why you should distribute I/O among separate devices, see the sidebar article "**Top 10 ways to improve server performance**." We isolated the operating system, Notes program files, Notes data directory, page file (or swap file on UNIX), and transaction logs (a new feature in R5) by placing them each on a separate physical disk or disk array, depending on the needed capacity. (Note: These files need to be on a *physical* disk. If you need to use multiple disks to increase capacity for one logical partition, use RAID5. RAID5 offers fault tolerance while maintaining some level of disk performance I/O.) For more information on I/O performance and RAID levels, see "**Optimizing server performance: I/O subsystems**."

In addition, we increased the system memory on most of the servers to at least 1GB. Plus, you'll notice that we added another UNIX flavor, Sun Solaris, to the site for running Beta builds of Domino R5 and for handling the increased traffic on the site.

The following table shows the current configurations of our servers:

Server	Operating System	CPU	RAM	Disk Structure*	Domino
Server 1	WinNT 4.0	Dual 333MHz Dell PowerEdge 4200	1GB	C: OS D: Notes Data/Weblogs N: Notes Program P: Page file T: Transaction log	R5.0a
Server 2	WinNT 4.0	Dual 333MHz Dell PowerEdge 4200	1GB	C: OS D: Notes Data/Weblogs N: Notes Program P: Page file T: Transaction log	Daily build server (R5.0.1)
Server 3	WinNT 4.0	Dual 450MHz	1GB	C: OS	R5.0a

		Dell PowerEdge 4300		D: Notes Data/Weblogs N: Notes Program P: Page file T: Transaction log	
Server 4	AIX 4.3.1	Quad 400 (IBM F40 2X)	2GB	/: OS /local1: Notes Data/Weblogs /opt: Notes Program /swap: Swap file /tlog: Transaction log	R5.0a
Server 5	SunOS 5.6 Generic	Quad 400	4GB	/: OS /local1: Notes Data/Weblogs /opt: Notes Program /swap: Swap file /tlog: Transaction log	Daily build server (R5.0.1)

Increasing the HTTP active threads setting
Our next changes involved looking at the various Domino parameters that we could tweak. First of all, we began the process of increasing the HTTP active threads settings from 40 (the default) to 80, and then to 120 active 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 on the Internet Protocols/HTTP tab of the Server document. You can then check the peak number of HTTP threads that Domino uses by typing the following at the server console:

```
"show statistic domino.threads.active.peak"
```

As you increase the HTTP active threads, you should see the response time decreasing. If your system spends too much time on overhead tasks, such as swapping memory, you might need to specify a lower number of threads. For more information on adjusting HTTP threads to improve performance, see "**Optimizing server performance: HTTP Threads settings.**"

In our initial monitoring, we noticed that the number of active threads kept growing to 120. By increasing the number of active threads specified in the Server document, we tried to get more system throughput. With the setting at 120, it didn't grow to that number. To see our HTTP Threads settings test results, see the sidebar "**Increasing HTTP active threads**."

However, we had to be careful because, as you increase this number, context switching also increases. This can cause the number of active threads to reach a ceiling. Context switches-per-second is the rate of switches from one thread to another. Thread switches can occur either inside of a single process or across processes. One thread asking another for information can cause a thread switch, or one thread preempting another can cause a thread switch. In the latter case, the higher priority thread becomes the one that is ready to run. (Look for more information on context switching in an upcoming *Iris Today* article.)

Adding the DominoAnalyzeFormulas setting
Next, we wanted to take advantage of the capability for caching Web pages that contain @functions (available in Domino R4.6.1 and later). In addition to keeping an in-memory cache of "static" HTML pages, Domino can also cache pages that contain macro language (@function) formulas. To do this, Domino's Formula Analyzer examines the formula on the page and determines whether the command is too volatile to be cached (for example, if the Web page contains @Today) and,

perhaps more importantly, under what conditions a cached command becomes invalid.

So, we enabled the Formula Analyzer on our servers by adding the NOTES.INI setting, DominoAnalyzeFormulas=1. Our servers then began caching Web pages where @formulas were used. For more information on the Formula Analyzer, see "**Expanded Command Caching in Domino 4.61**".

Changing Domino cache levels
After setting up the Formula Analyzer, we wanted to check out how the caching was actually working on our Domino servers. To do this, we began monitoring the following server statistics:

- Domino.Cache.Command.Count: The actual number of commands that the Command Cache contains
- Domino.Cache.Command.MaxSize: The maximum number of commands that can be cached
- Domino.Cache.Command.HitRate: The percentage-ratio of the number of times a valid cache entry is found in the cache to the number of times the cache was investigated for a cache entry
- Domino.Cache.Command.DisplaceRate: The percentage-ratio of the number of times that a new cached command displaces an aged command to the number of times the cache was investigated for a cache entry.

For more information on these statistics, see "**Expanded Command Caching in Domino 4.61**".

The following table shows some of our results from monitoring the caching statistics. In general, the results show that Domino doesn't cache many of our pages. Notice that the Command.Count and Command.MaxSize results were typically equal. Plus, the Command.HitRate and Command.DisplaceRate results were very low.

Server	Command. Count	Command. MaxSize	Command. HitRate	Command. DisplaceRate
Server 2	128	128	4.00	1
Server 4	128	128	3.69	0
Server 2	128	128	7.08	1
Server 4	119 (with restarts)	128	3.32	0.34

These results caused us to investigate the makeup of our site more. A few areas of the site are user-specific -- that is, we require users to authenticate to participate in the discussion forums, to download Beta releases, and to submit samples to the Sandbox. In order to serve a cached page in these areas, the request for the page must come from a user who already accessed the page using that exact URL. Because the hit rate of these cases is low, you can see a low displacement rate (the value is out of 100, with 100 being 100 percent). Remember that the HitRate statistic shows how many times a match is found in cache. We are always at a low number -- so, a match is not found.

We checked with the Domino developers, who recommended that we make sure that the "front doors" of our site are cached. The "front doors" are the home page and the main launching page for the subsections of the site. (Increasing the MaxSize to a larger number might have allowed us to cache more data and improve performance, but the HitRate and DisplaceRates were likely to remain low because of the dynamic content on the site.)

In response to this, we began focusing on the home page of Notes.net. We learned that the HTML on the home page is never cached because it has an @Today command on it. However, the images on that page are cached. Since most of the page isn't cached, when a user returns to our home page after visiting it once, the

HTML reloads. One solution is to add a $CacheValid field to the document, and set the value of the field to a numeric text string, *N*. This causes Domino to protect the page from validity checks for *N* seconds. The default for the HTTP server is N=0. We did not implement this setting, because we first wanted to evaluate the impact of our other changes.

Domino R5 transaction logging
In our process of upgrading to the Beta builds of R5, and eventually R5 Gold, we also began taking advantage of the various performance improvements in R5. One of these features is Domino R5 transaction logging. With transaction logging enabled, the system captures database changes and writes them sequentially to the transaction log. Then if a system or media failure occurs, you can use the transaction log and a third-party backup utility to recover your database. You also get faster server restart times, greater database integrity, and much higher system availability. For more information on transaction logging, see "**Optimizing server performance: Transaction logging**."

How our changes affected performance

We collected a great deal of data during the months that we monitored performance. However, analyzing the data proved to be more challenging. This is because during the testing time, we continually upgraded the servers from R4.6 to various Beta builds of R5, and finally to R5 itself. As mentioned earlier, Beta builds are not optimized for performance, and they include a lot of debugging code that slows down the server. Plus, we also made all the changes that we've described throughout this article.

Of course, performance testing works best in a controlled environment, where you can change one thing and notice the impact. Also, you need to be able to watch the impact of the change over an extended period of time. Unfortunately, we work in a fast-paced, ever-changing environment (as most Web sites do!) We tried to control the systems for at least a week after a major change, but we discovered that a week really wasn't enough time to evaluate the impact of a change.

However, the important thing is our end-result -- that the site performance improved! We collected feedback about the improved end-user response time from our users (through the **Feedback** link that appears at the bottom of every page on our site). Also, our internal tests showed that the home page now only takes a few seconds to load. We're unable to pinpoint just one factor that caused this performance improvement, but we now have baseline data to use in our future performance analysis efforts.

The future of Notes.net performance monitoring

In the future, we plan to continue monitoring our site and collecting more data for further performance analysis. We hope to take a look at the performance impact of adding a UNIX server to the site. We also want to create a Notes.net workload that we can use for future performance tests in a simulated Notes.net environment. Then, we will be able to control the changes, and test the results before implementing them on our live site.

We hope that you can learn from our experience, and get ideas about ways to improve the performance of your own Web site. We encourage to share your results with us in this month's **Developer Spotlight**, where we feature the Performance team. They can provide you with specific feedback on the performance monitoring techniques you use and help you analyze the results.

ABOUT CAROL
Carol Zimmet started working at Iris in 1994. She is responsible for evaluating performance and performance tool development on the server team. Carol continues to search for the one-step solution to everyone's performance problems. Carol enjoys bicycling with her kids in the trailer and playing racquetball. She has a longing to return to stained glass!

[back to "Notes.net Exposed: Improving Web site performance"]

Installing and viewing Domino as a Performance Monitor counter (sidebar)

This section lists the steps for installing and viewing Domino in the Windows NT Performance Monitor.

Installing Domino as a Performance Monitor counter

To install the Domino Performance Monitor components:

1. Run the Domino Install program again and click the Customize button.
2. Make sure that the install paths are the same as for the original server install.
3. De-select all install options except for "Notes Performance Monitor." This allows you to install only the Performance Monitor components.
4. After the Install program completes, restart the server.
5. Type the following command from the program directory on the server:

   notesreg.bat *directory*

   where *directory* is the full path to the program directory. For example:

   notesreg.bat C:\notes

Viewing Domino statistics in the Performance Monitor

To view your Domino statistics in the Performance Monitor:

1. Open the Performance Monitor by going to the Windows NT Start menu and choosing Programs -

   start perfmon

   Choose View - Chart, and then Edit - Add To Chart.
3. In the Object box, select Lotus Notes.
4. In the Instances box, select the Domino statistics you want to include in a chart, then click Add. Repeat for each statistic you want to add.

   If no Domino statistics appear as instances in the Performance Monitor, initialize the statistics on the server -- for example, type Show Stat at the server console, or let the server run a few more minutes. The Lotus Notes option does not appear until the Domino server loads. Domino statistics do not appear as instances in the Performance Monitor until Domino or an add-in program assigns or updates a statistic.

Note: If you experience difficulty viewing the Lotus Notes object, or if after loading the Lotus Notes object, you cannot find it in the Performance Monitor, remove the installed Lotus Notes object by entering the command *unlodctr notestat*. Then, run the above *notesreg.bat* command again.

Register Here!

About this Site | **Feedback**
Lotus Home **IBM Home** |

[back to "Notes.net Exposed: Improving Web site performance"]

Overall response time results (sidebar)

The following table shows a snapshot of our overall Notes.net response time results (in seconds). Notice that there appears to be a rough correlation between server downtime and the average overall site response time. The results show the probe response time according to the different days of the week.

Week beginning...	Monday	Tuesday	Wednesday	Thursday	Friday	Weekend
2/01/99		5.04/3.54*	2.42*	2.71*		
2/08/99	2.81*	2.42*	2.42*	about.html: 2.14*	2.48*	2.48*
2/15/99	8.96* (two servers down, half-day)	2.55*	2.77*	4.41*	4.49*	8.77* (two servers down)
2/22/99	8.77* (two servers down)	8.96* (two servers down)	2.63*	cont*	4.19* (one server down, plus one server down half-day)	cont.*
3/01/99	4.19*	cont.*	2.74*	5.64* (two servers down)	4.31* (one server down)	cont.*
3/08/99	2.44*	3.56*			2.24*	cont.*

* Represents best case -- retrieving first object only.

[back to "Notes.net Exposed: Improving Web site performance"]

Increasing HTTP active threads (sidebar)

The following table shows how we increased the HTTP active threads over time, and the effect this had in terms of the average number of HTTP requests and the server uptime. We first recorded the Domino release that was running on the server.

The "HTTP Request Count: 1-Day Average" column shows the average number of HTTP requests for one machine during a one-day period. However, not all HTTP requests are equal. When you compare the one-day average numbers with the one-hour averages in the next column, you are really getting an overall indication of the workload on the server. (Remember that the user load helps you determine the best HTTP threads setting.) To truly understand the workload, you should also rely on other system metrics (such as the CPU, response time, memory, and disk I/O). In the third HTTP Request Count column, we multiplied the average hourly rate by 24 to see how close it came to the daily request rate in the first HTTP Request Count column. This was one way that we double-checked our data.

The Server UpTime column shows results for our availability analysis. We kept track of the percentage of time that the HTTP server was available for responding to our Probe workload. For example, in the first monitoring period, the server responded to 353 of the 381 probes, or 93 percent.

Date	Server	HTTP Threads	HTTP Request Count: 1-Day Average	HTTP Request Count: 1-Hr Average	HTTP Request Count: 1-Hr Average * 24	Server UpTime
1/22	Server 2, R4.6	40	332,634	13,049	313,176	353/381 (93%)
	Server 4, R5 Beta	40	365,149	15,784	378,816	363/381 (95%)
1/26	Server 4, R5 Beta	40	205,848	16,547	397,128	473/551 (86%)
2/1	Server 4, R5 Beta	40	251,063	16,684	400,416	627/664 (94%)
2/15	Server 2, R4.6	80	518,947	25,860	620,640	590/594 (99%)
	Server 5, R5 Beta	80	450,644	22,820	547,680	414/594 (70%)
2/22	Server 2, R4.6	80	566,959	24,800	595,200	639/642 (99.5%)
3/1	Server 2, R4.6	80	413,203	16,264	390,336	210/211 (99.5%)
	Server 2, R4.6	120	240,642	13,257	318,168	414/458 (90%)
	Server 5, R5 Beta	80	382,125	17,228	413,472	273/280 (97.5%)