Case Study: Load Testing and Tuning to Improve SharePoint™ Website Performance

Abstract: Initial load tests revealed that the capacity of a customized Microsoft Office SharePoint™ Server (MOSS) website cluster was far below the required level. With 200 concurrent users, average page durations ranged from 10-30 seconds. The cluster of 3 SharePoint™ servers and one SQL Server could serve only 4 pages/sec. Several iterations of load tests and tuning of the SharePoint™ configuration, custom code, and SQL Server resulted in steadily improving performance of the system. Limited only by bandwidth, the system can now support ~2000 users with average page durations of 2-5 seconds. At that level, system throughput is over 45 pages/sec with all servers running below 25% CPU utilization.

Introduction

When we (Web Performance) first discussed the project with the customer (SHRM – Society for Human Resources Management), they expressed concerns over the performance of the system due to slow pages seen during system development. However, the website had not undergone any stress testing to date. The new site would be replacing the customers existing website – which is the primary interface between SHRM and their 250,000 members. The new site was scheduled to go live in 4 months and would bring a measurable benefit to the organization through improved work-flow and publishing features as well as an enhanced customer experience. However, the project leadership needed assurances that the new site would handle the anticipated load (1500 simultaneous users during peak hours).

Testing Environment

The system, as tested, consists of a cluster of 4 SharePoint™ web servers, an application server and a database server. One of the SharePoint™ servers is reserved for search indexing. The application server is used for a few distinct functions, such as authentication. Each web server is an HP VL360 G5 with 2 quad-core processors and 16G of RAM. The database server runs on the same hardware with 32G of RAM. The servers sit behind a Cisco CSS load balancer on a DS3 line (45Mbps). The database storage is on an EMC Clarion CX-500 SAN and the web servers use only local disk storage. All servers run Windows Server 2003 64-bit Enterprise SP2. The web servers run Microsoft Office SharePoint™ 2007 64-bit. The SQL server runs SQL Server 2005 (roll-up 8).

Test case Design

Working with the customer, we proposed a handful of test cases to exercise ~500 pages in their site using our load testing software, Web Performance Load Tester™. This was a relatively small subset of the entire site, which is content-rich with over 15,000 articles. Considering a looming project deadline, this was agreed to be a reasonable compromise for getting rapid results. The result was 5 test cases that exercised various navigation paths through the site.

Initial Tests

A few days later, we were ready to execute the first tests of the site. Initial tests were not promising. Under a simulated load of 100 simultaneous users, the system returned pages, on average, in less than 3 seconds – but that was only after the first group of users had passed the homepage and login steps. During the ramp-up, the average page durations (APDs) peaked over 12 seconds. After the second group of users was added (for a total of 200), average page durations exceeded 20 seconds, as shown in this chart:

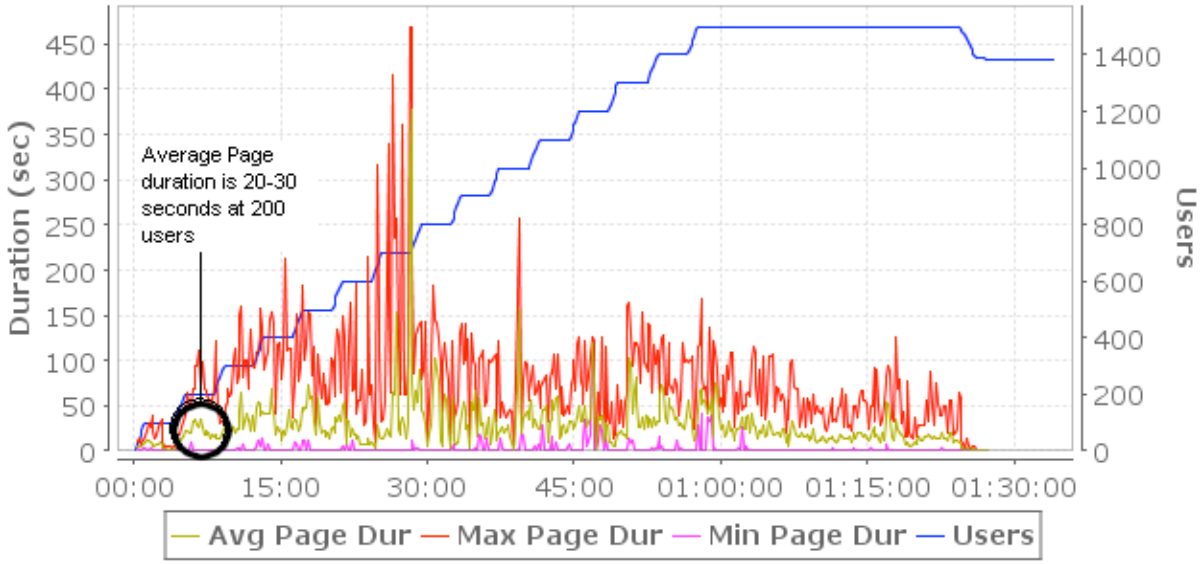

figure 1: Initial testing shows poor performance - 20-30 second page durations at 200 users

The metrics gathered during the test (via Load Tester's Server Monitoring Agents) indicated that hardware was not the bottleneck - neither CPU, memory or disk were taxed during the tests. A series of tests and subsequent investigations indicated that the network and load balancer were not the limiting factor either.

Next, we isolated each SharePoint™ web server in the cluster and tested them individually. The tests revealed a number of differences between the servers. For instance, one server was not compressing the page content. More importantly, we found that running with only a single SharePoint™ web server resulted in better performance (average page durations under 6 seconds) up to 300 users – three times the capacity of the system with 3 web servers (note that this test ran for a shorter period – thus the change in scale on the Users axis and the time axis).

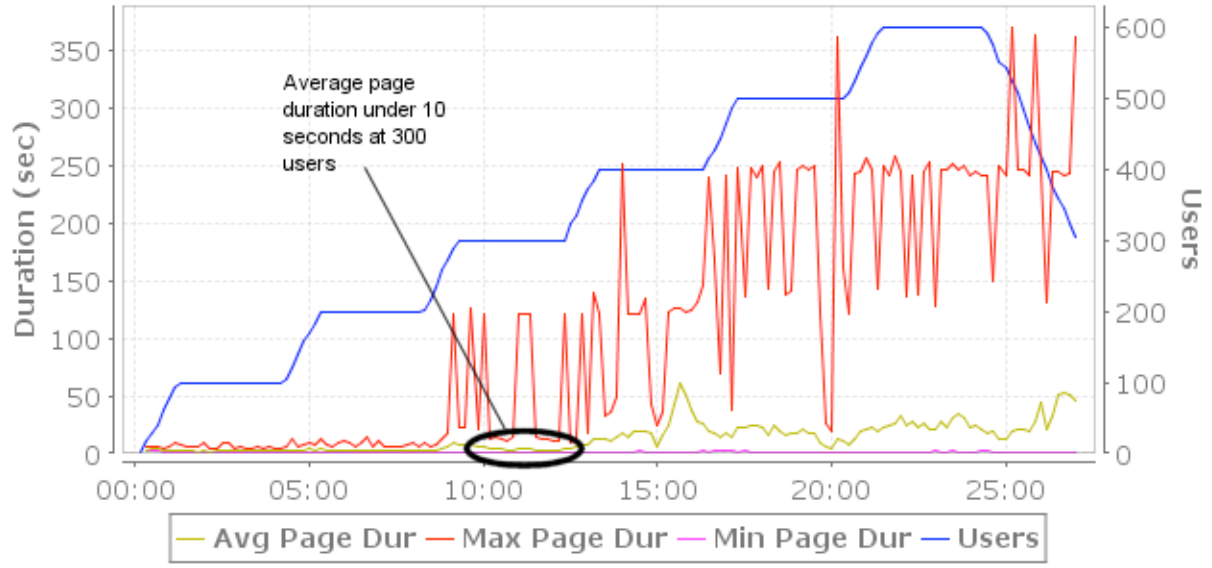

figure 2: A single server performed better, but performance is still not acceptable

We also noted that CPU utilization was not scaling linearly with the applied user load. At ~400 users, the CPU utilization peaked on the web and database servers at ~60% and ~30% respectively:

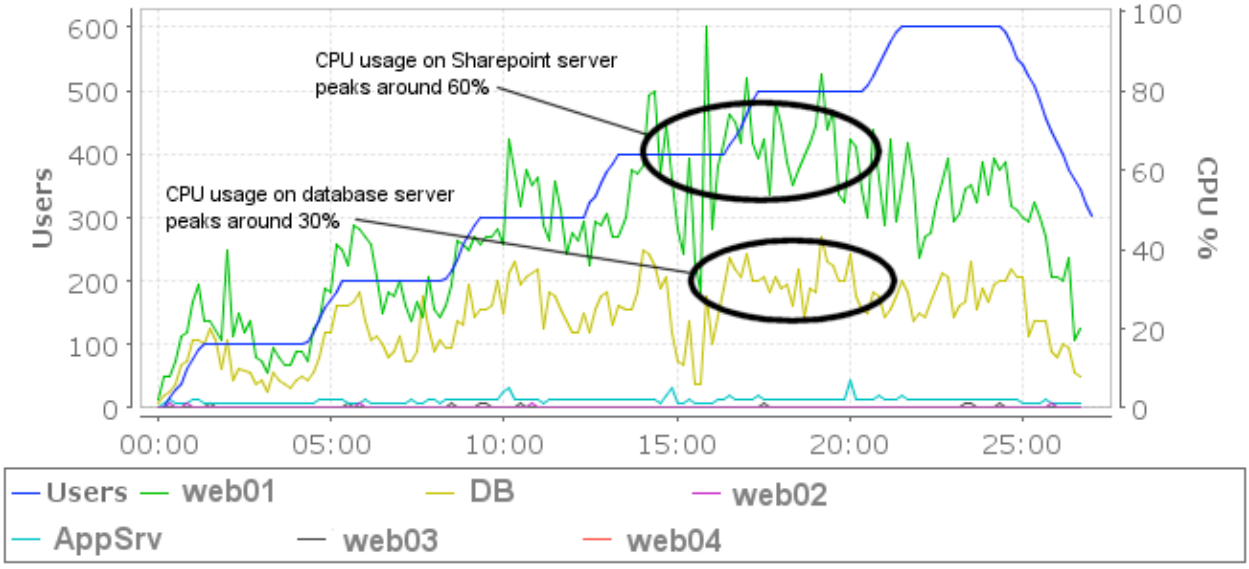

figure 3: CPU utilization levels off after 400 users

Additional user load did not raise these levels – in fact, CPU usage declined as more load was added. After the peak, additional load did not raise the key throughput metrics, such as hits/sec, pages/sec and bytes/sec. The server metrics did not indicate a bottleneck in any other hardware category (network, memory or disk). This left software or software configuration as the most likely limiting factor. The most common culprits in this situation are connection pools, thread pools, resource contention and database locking.

SharePoint™ Tuning

During the next series of tests, we focused on testing a single server, since there is little point in load testing and tuning a cluster of servers when the individual servers are not operating up to their potential.

A number of optimizations to the SharePoint™ configuration were suggested and implemented, including:

- Move static resources (images, etc) to an image library to facilitate caching of the resources in the browser
- Change SharePoint™ cache settings to *Extranet Publishing Site*
- Change the custom role provider to use *Role Provider Caching*

In addition, the Content Query Web Part was changed to handle taxonomy more efficiently. After each change was implemented, the system was tested to measure the change in performance. In each test, an improvement in bandwidth utilization was observed (particularly between the SharePoint™ servers and the database). However, the end-user performance was unchanged.

Next we tried to determine if the entire SharePoint™ installation would share this performance profile, or only the instance that was being tested. The customer created a new *out-of-the-box* SharePoint™ site using one of the example sites. This site was tested to 1500 users with only slight degradation seen at the peak. The test was very near (or past) the bandwidth limits of the network connection (a 45 Mbps DS-3).

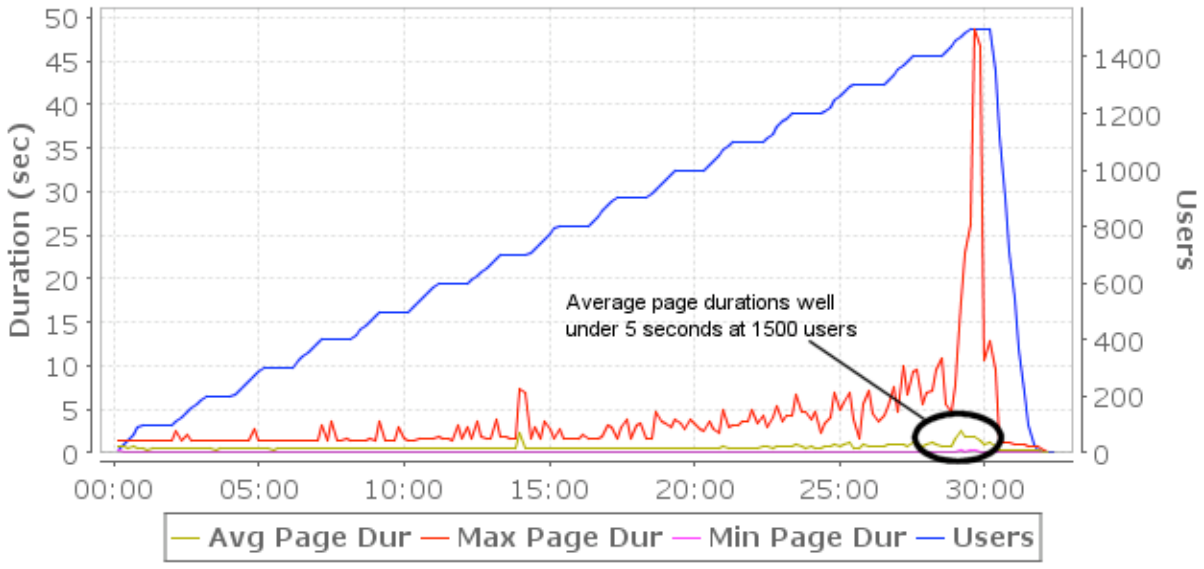

figure 4: Average page durations are greatly improved - under 5 seconds up to 1500 users

We were now convinced that the OS, hardware and SharePoint™ installation were healthy. We returned to the original site and targeted authentication. A new test case was designed that visited 6 public pages as an unauthenticated user. The system was tested and scaled to 1000 users, but performance was poor with average page durations in the 10 second range. The system was stable, but performance had degraded rapidly by 1200 users (when we again hit the bandwidth

limits).

Curious if the improved results of previous test were due to a lower number of unique pages visited, rather than authentication, we next designed a test case that visited a larger number of pages, both authenticated and not. This included more pages than the first unauthenticated test, but a lot less than the original test scenario. This load test produced better performance, but was unstable – exhibiting a stalling behavior when under load. For example, the system was able to ramp up to 1300 users with the system serving ~30 pages/sec. As the test added more load, the system throughput suddenly dropped to less than 5 pages/sec. In multiple test runs, the stalling behavior was observed at varying load levels.

figure 5: System throughput scaled with load, then dropped to very low levels

The test case was dissected in several iterations to determine if any particular group of pages performed better or worse than others, but no offenders were found. We again returned to a set of pages that did not require authentication – this time picking a larger set of pages containing a variety of features (27 pages total). Load tests revealed the system could service these pages with average page durations under 1 second at 1500 concurrent users with consistent throughput (~39 pages/sec) for 2 hours. Further experimentation revealed that the addition of one relatively simple test case caused the system to become unstable. This gave us an easy way to demonstrate good and bad performance of the system under the same configuration (with different usage patterns). We hoped this would allow Microsoft Support Engineers to diagnose the problem.

During some of the previous tests, we also noticed that system performance sometimes degraded consistently from one test to the next. We subsequently discovered that rebooting the database between test runs temporarily improved performance. To help get consistency from the test results we began regularly rebooting all the servers prior to each test. This was good test practice to ensure a consistent testing environment. We came to realize that there is a larger significance to this particular symptom – although the realization did not come until later in the process (discussed later in this paper).

After looking at our test results as well as collecting their own data, Microsoft SharePoint™ Support indicated that SharePoint™ was apparently unable to make use of such large hardware (8 processors with 16G of RAM). In an effort to validate that the problem was indeed caused by the large hardware, they recommended that we reduce the number of processors to 4, and then later suggested reducing it to 2. In each case, this resulted in a surprising performance improvement but the stalling behavior remained. Reducing the number of processors moved the point of failure, allowing the system to run longer before stalling, but did not cure the problem. This proved that we had a problem unrelated to the size of the hardware that warranted more detailed, low level analysis.

Database Tuning

Early in the testing, Web Performance suspected that the database was the bottleneck. However, an analysis of database performance during the tests by both the customers in-house DBAs as well as Microsoft DBAs determined that locking contention was at low levels and the database was performing well. This had put the focus on the SharePoint™ servers. It now seemed prudent to return our attention to the database.

After additional testing and data gathering, Microsoft Support engineers found that contention on *tempdb* allocations within SQL Server was causing delays processing queries from SharePoint. This problem is described in the Microsoft Knowledge Base (#328551).

The fix requires creating additional *tempdb* databases within SQL Server (1 for each processor) and enabling a startup parameter (-T1118) that instructs SQL Server to use a round-robin *tempdb* allocation strategy. This will reduce resource allocation contention in the *tempdb* database to improve performance on complex queries.

After making this change, load tests indicated that the system was able to sustain 15 pages/sec (650 users) for 2 hours on a single server. Web page performance had improved - average page durations were down to the 2-4 second range. Specific changes to custom SharePoint™ components and some additional database optimizations suggested by Microsoft Support brought average page durations under 1 second.

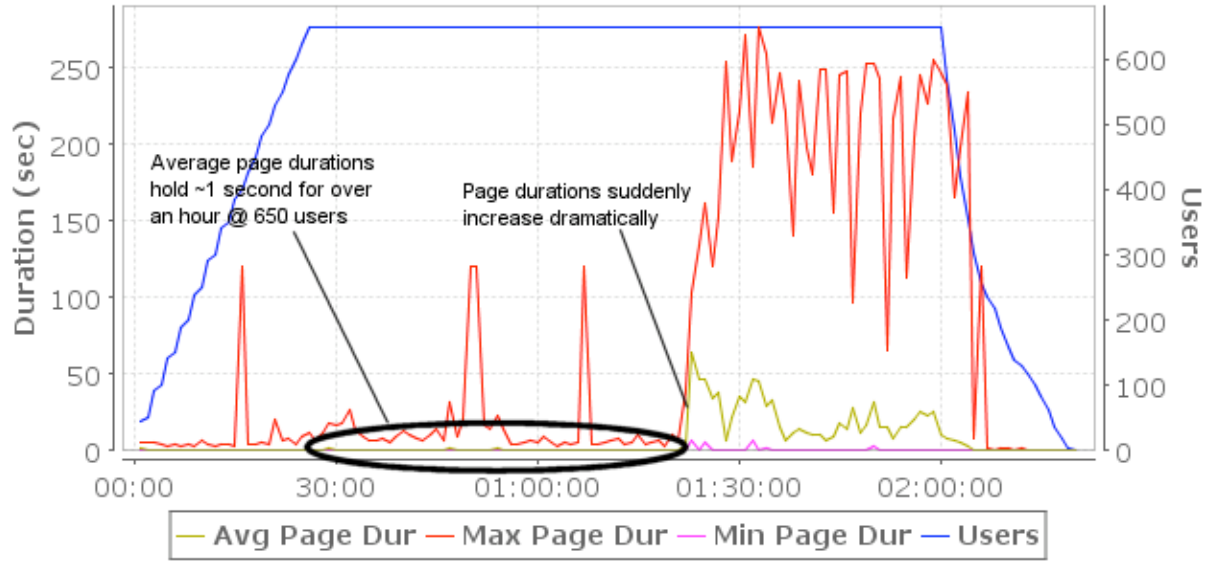

figure 6: Page durations were greatly improved, but performance was not stable

While we had achieved a fast, stable system on a small subset of pages, the instability re-appeared when we re-introduced the remaining 3 test cases into the mix. The behavior appeared after roughly 80 minutes of operation at load. It was not as bad this time – rather than stalling, throughput would suddenly drop by 30-50% and then oscillate up and down wildly:

figure 7: System throughput is good, but degrades severely and unpredictably

There was some debate at this point if the SharePoint™ Server or the SQL Server was the culprit. We recalled the "rebooting the database fixes the problem" discovery from previous testing and brought this to the attention of the Microsoft Support engineers. We also found that if the load test was stopped when the servers were in a degraded state and restarted within a few minutes, the degradation would continue, even at very low load levels. Further diagnostics around these symptoms revealed that once the system performance had degraded significantly,

clearing the query plan cache in SQL Server (via DBCC FREEPROCCACHE) would restore system performance almost immediately. Unfortunately the fix was not permanent – the performance degraded again within a short period of time.

This led the Microsoft engineers to a Microsoft Knowledge Base article (#927396) that indicated problems with the size of the TokenAndPermUserStore cache in SQL Server. When the server has a large amount of physical memory (in this case 32G) and the rate of random dynamic queries is high, the number of entries in this cache grows rapidly. As the cache grows, the time required to traverse and cleanup the cache can be substantial. Because access to this cache is single-threaded, queries can pile up behind each other waiting for the cleanup to complete. This slows performance and prevents a multi-processor system from scaling as expected. This was remedied by starting SQL Server with a "-T4618" parameter (note this is not one of the listed solutions for this issue – it was provided by a Microsoft Support Engineer) which limits the TokenAndPermUserStore cache size.

After the above fix was applied, the next load test of the system showed steady performance with 15 pages/sec and APDs under 1 second, supporting 650 concurrent users for 10 hours. However, in a subsequent load test, errors started appearing in the pages indicating that SharePoint™ was unable to render many web parts on the page: "Arithmetic operation resulted in an overflow". Microsoft quickly traced this to a bug in a SharePoint™ cache implementation that was fixed by reducing the SharePoint™ Security Token Cache size. Search the knowledge base for the article "Object cache throws Integer Overflow exceptions when cache size > 2000" for more details.

With the above fix applied and tested, the system was ready for a longer stress test to judge the stability of the system over longer periods. The next load test ran for 48 hours at 650 users. The system performed well – easily satisfying the performance requirement with only a single SharePoint™ web server. No degradation of performance was observed. Further testing with all three SharePoint™ servers and higher load levels showed similar success.

Final Results

Prior to stress testing and tuning the website, it could handle only 100 users (4 pages/sec). It can now handle 2000 users (45 pages/sec, nearly 800 hits/sec) with low CPU utilization (~20%) on the servers. For reference, if held for an entire day, this rate would result in ~3.9 million page hits per day.

At 2000 users, CPU utilization of the servers is below 25% -- the customer's Internet connection is now the only factor limiting total capacity. With a higher bandwidth connection, it is possible that this site could now service up to 8000 users.

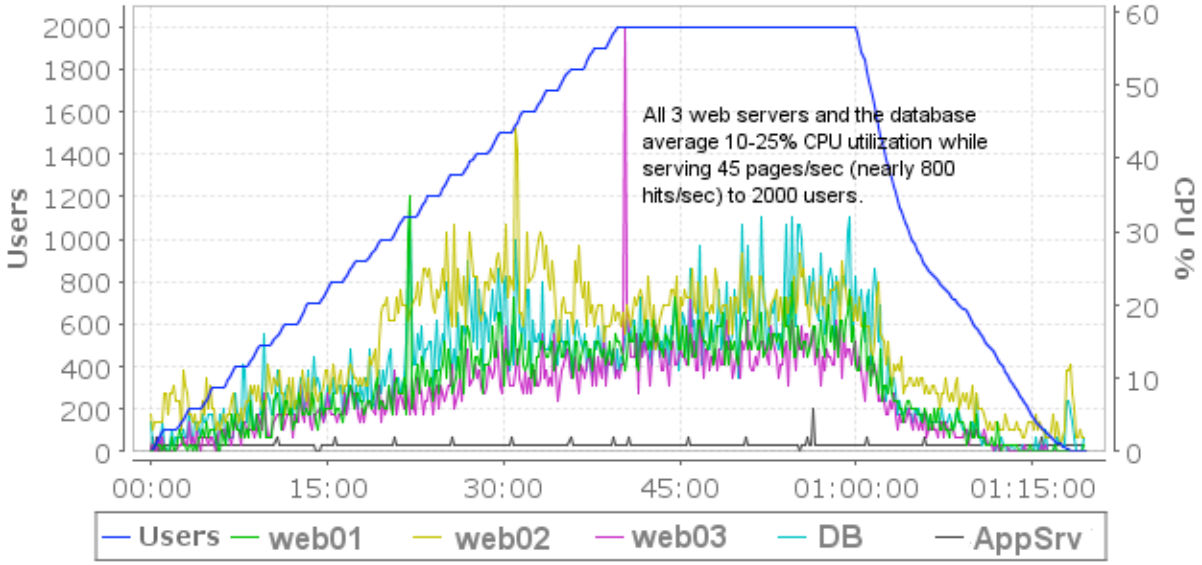

figure 11: Servers show low utilization at 2000 users

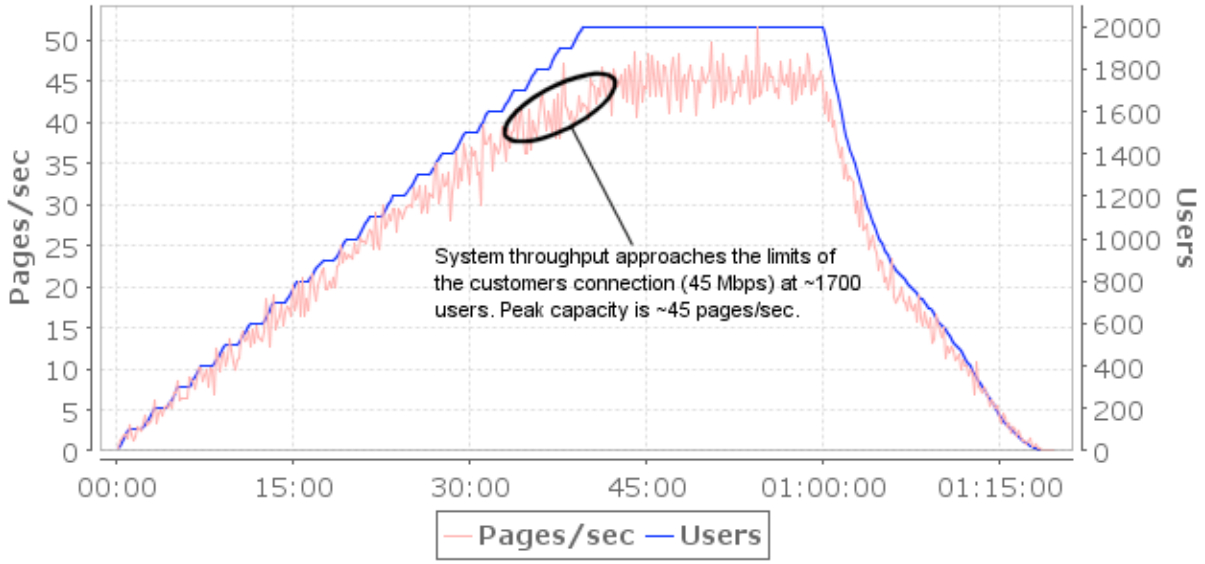

figure 9: 2000 user test shows high throughput and steady performance

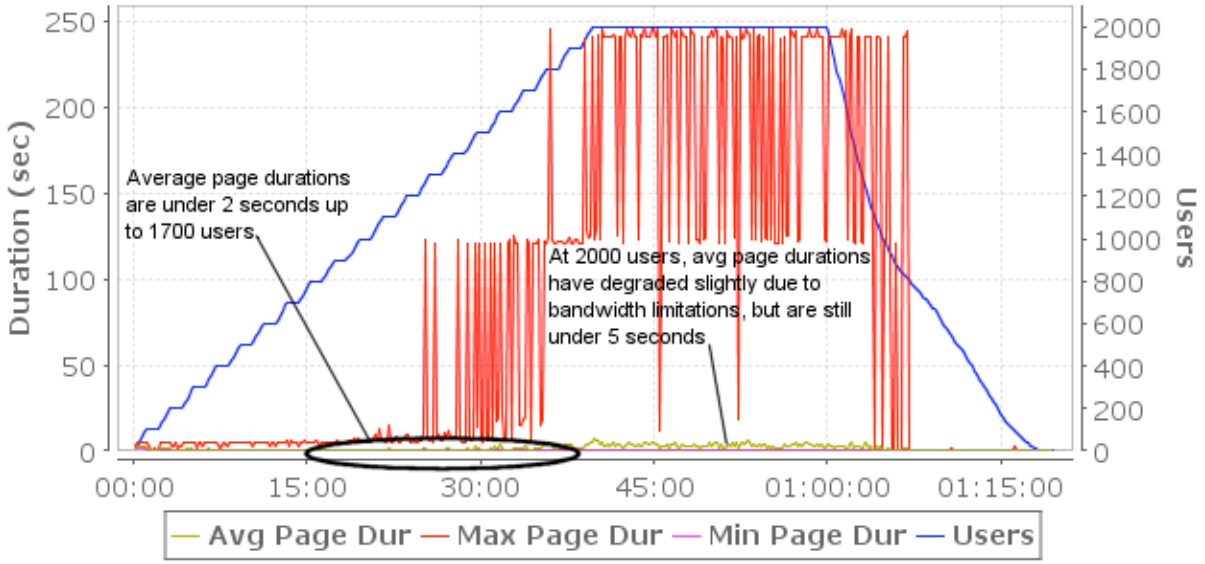

figure 10: 2000 user test shows low page durations

Additional Resources

- How to optimize a SharePoint™ Server 2007 Web Content Management Site for Performance
- SharePoint™ Development Capacity & Performance Planning 2003 & 2007 What you need to know...
- http://blogs.technet.com/wbaer/archive/2008/03/01/bill-s-5-min-sharepoint-performance-recommendations.aspx
- SharePoint™ performance demands finely tuned SQL Server, storage