Web Performance, Inc. Testing Services
Sample Performance Analysis

Overview

This document contains two performance analysis reports created for actual web testing clients, and are a good example of the extra information provided by our engineers over and above the information generated by our automated How Many Users? report and Advanced Server Analysis.

Customer Analysis 1: Java/J2EE Web Application

Earlier, I hope you have received a link to view the results from last night's test online at

http://www.webperformanceinc.com/clients/XXXX/1-31-test-results/index.html

Additionally, a download link should now be available for the complete results at

http://www.webperformanceinc.com/clients/XXXX/1-31-test-results.zip

In looking at these results, it appears that the server has had a substantial improvement since the last test. Notably, the performance of the system seems quite stable up until 120 users, show no serious signs of degradation. There are a couple dropped connections early on, but since those appear seldom, it seems safe to assume those were just some routing issues in accessing the server from an external network, and are not a serious concern. I have configured the duration compliance to 90% to ignore those anomalies.

The trouble seems to start roughly 16 minutes into the test. At this point, the first errors are observed coming back from the server. In looking through the initial errors, there are a couple which come back with stack traces. The two which had some content have been attached. Similar errors appear to be recorded up until about 24 minutes into the test.

At roughly 24 minutes into the test (180 users), the system comes to a nearly complete stop. At this point, the application servers seem to jump to 12% or 13% CPU time. Further, many users see a loss of responsiveness, and many of the responses received are errors from the server.

At this point the test was allowed to ramp-down. While app servers 1 & 3 appeared to recover, app server 2 appeared to show continuous activity until it was forcibly reset.

The 12 to 13% is interesting for these servers, since the servers report having 8 available processors. Given this, 12% would suggest that only one processing core is being utilized on the system at the time of the error.

I would like to offer a few thoughts regarding the errors that you have seen. Obviously, there is a limited amount that can be said without examining the server's error logs and profiling data, but I will try to offer a few brief guesses and options to consider.

Note that PermGen errors typically come up when multiple classloaders are reloading classes, instead of re-using classes, or Proxy classes are being generated by the same class, but assigned to different classloaders. However, the PermSize for a Sun Java VM can usually be increased by adding

```
-XX:MaxPermSize=128m
```

to the java startup command-line. Note, if my memory is correct, there is a 2 GB limit on a 32 bit VM, where the <data heap> + <permanent heap> + <memory mapped files> <= 2 GB. If you need to utilize more memory, you could try a 64 bit VM (assuming that the VM is running on a 64 bit OS). The downside is that if classes are being reloaded instead of re-used, increasing the permanent heap may only post-pone the errors. Since the test had been running and had many successful repeats before the first of these errors was recorded, it seems that all the classes necessary should have been loaded at this point, and there would not have been much reason to continue loading data into the permanent heap.

That the processors seem to have suddenly jumped to 13% is interesting. If this means that only one core is being utilized, then this could mean clients are not being serviced in parallel on the same VM. This may be worth a check to each app server's thread-pool settings. However, that the CPU usage jumped from 8%-9% to 13% may mean that the cause of the jump is more important to investigate. Given the memory related errors mentioned earlier, this could be an indication that the servers were suddenly force to perform a full GC (Garbage Collection). Additionally, we see that the OS available memory also increased just prior to this point, indicating the VM was forced to grow to it's max heap size. If this is an indication that the memory inside the VM is running out, the GC would block the business logic from running, and force a CPU to full load in order to free up the necessary memory. If that does turn out to be the case, it could be remedied by increasing the VM memory for the app server, and also providing some tuning for the GC process. For Sun's 1.5 VM, a couple good resources available for GC tuning:

http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html
and a more general document
http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp

Eg: for a system with many processors, it may be beneficial to use a multi-threaded GC approach, by adding

```
-XX:+UseParallelGC or -XX:-UseParallelOldGC
```

to the java startup command line.

I hope that this information gives you some good insight, as well as a few places to look at how to get past the mentioned 120 user limit. I think that you should be quite proud, these results clearly show a substantial improvement since the last pass.

If you have any questions, please do not hesitate to ask. I will look forward to hearing back from you, as well as hearing how you would like to proceed!

Customer Analysis 2: ASP/.NET

I wanted to follow up with some comments regarding the load test run yesterday. As noted, the results of this test are available on our website, at

http://www.webperformanceinc.com/clients/XXXX/5-8-08_external/index.html (external test),
http://www.webperformanceinc.com/clients/XXXX/5-8-08_internal/index.html (internal test), and
http://www.webperformanceinc.com/clients/XXXX/5-8_test_results.zip for the complete download.

These tests have been developed around the workflow previously described. The time for the external workflow was set to require approximately 44 seconds, and 35 seconds for the internal workflow. Both tests included a 3 second delay at the end of the test before a new user would pick up the next id. At 500 concurrent external users, this would generate an expected 638 name changes per minute, plus 429 name changes per minute from another 250 concurrent internal users.

In looking through the results, we notice that the system appears to be scaling well for roughly the first 14 minutes of the test. After 14 minutes though, we notice that the hits/second and bandwidth appear to reach a limit for the external users, and noticeably degrade for internal users.

The internal user capacity report shows that 60 users were running on the site at this point in time with page durations staying below 6 seconds. After this point in time, we notice that a substantial percentage of the pages begin requiring significantly more time, and nearly 2/3rds of pages require more than 6 seconds to load at 75 internal users.

The external user capacity report first flags a sudden increase in page durations, in which pages suddenly require more than 6 seconds at roughly 98 external users. We do notice that this appears to recover, and performance is restored around 102 users, so we may speculate that this sudden slowdown was the result of a connection pool establishing more connections. This recovery appears to last up to roughly 121 users, after which users begin to consistently encounter some page durations greater than 6 seconds.

For the internal users, the errors highlights errors starting 41 minutes into the test, which appear in pairs of   “Validation error” followed by an “Extractor could not locate...”. These errors appear to be caused by the browser's connection to the server timing out. Generally, a web browser will retry a failed connection, and this behavior was simulated during the test. However, when the user attempted to retry the transaction, they were unexpectedly rechallenged for authentication by the NTLM authentication configured on the server, thus causing these errors to be mislabeled as Validation + Extraction errors. Since the performance appears to have degraded well before these errors were encountered, these errors are not a significant concern.

Errors encountered by the external users appear to be relatively rare, at only 3 errors encountered during the entire test. This error appears to be a message: “Object reference not set to an instance of an object.” (please see the attached “5-8-external-error.html”). Given that this error appears early during the test, and infrequently afterwards, it looks like this errors is probably only a configuration error with a small subset of users in the provided dataset, and it should be safe to rule this error out as a symptom of load on the system.

Since this makes it look like we can rule out most of the early errors encountered as not being indicative of load on the system, the user capacity reports have been configured not to factor in the errors into these figures. Thus, we can estimate that, assuming a goal of 100% of pages having durations under 6 seconds, the system was able to support 60 internal and 121 external users before reaching it's capacity.

In looking through the pages, it appears that in both tests the “Policy Detail” page appears to show the first signs of degradation, before other pages in the testcases. In both internal and external tests, this page appears twice – when the user is displayed a policy, and again after the name change is completed. Approximately 15 minutes into the test, we notice that instances of this page appear to show an average of 7 to 9 second load times. Under load, we notice that most of the application resources appear to show similar behavior, and page durations increase for all of the .aspx URLs. Static resources (such as the “Menu Elements” page triggered by mouse-hover events) show almost no degradation in performance throughout the test, supporting that the bottleneck encountered was not bandwidth.

The server performance data is included in the internal user report (available under the “Servers” section of the report), as it was captured by the workstation on site with the servers. Please note the users mentioned in the graphs for the server data only represent the users from the internal test, and do not include the external users running at the same time. During our 10 minute discovery test, we noticed that the "tmkhatswebsim" server showed the most significant signs of load, and I would like to examine this server in some additional detail. During the load test, we notice the CPU measurement appears to stay below 40% for the duration of the test, which is not particularly concerning. However, we the captured processor time for the IIS and ASP processes appears to show 100% utilization. These figures seem inconsistent to me, but the reported figures are collected through the same Windows APIs as perfmon, so this is a curious point. If we look at the "ASP.NET Request Execution", and "ASP.NET Application Requests" we note that the ASP server seems to reach it's max executing request limit roughly 15 minutes into the test, and begins queuing requests past this point. It appears that once an opening in the thread pool becomes available, it is able to deplete many of the waiting requests from the queue before encountering another long running request.

From these results, it is not entirely clear if the "tmkhatswebsim" server is the bottleneck, though the metrics reported indicate this is a more likely candidate than the other two servers. It is possible however, that certain ASP requests are contending for a resource lock, tieing up availability in the ASP worker thread pool. Given that the bandwidth limit was not reached during this test, it may be worth-while to consider disabling IIS's compression for dynamic content. This feature was previously disabled before the bandwidth requirements for the test were determined, and it does include a small level of processor overhead. Compression for static content is generally compressed once, so this feature should only affect the bandwidth requirements. Since the bandwidth limit after your optimizations was not reached, trading some bandwidth for processor time may be beneficial for the overall user capacity.

Additionally, since the "Policy Detail" page appears to show the first signs of degradation, we may make another note. Each "Policy Detail" page includes a forward to the actual page. Eg, when an external user requests this page, they request "Policies_registered.aspx", which forwards to "Policy_Detail.aspx". After making the name change, the user posts the change to "NameA_Change.aspx", which forwards to "Policy_detail.aspx". This strategy of forwards is not unusual for .NET applications. What is somewhat unusual however, is that the forwards appear to contain a substantial amount of content which the user never sees. Using the first forward in this example, the forward contains the content:

```
<html><head><title>Object moved</title></head><body>
```

```
<h2>Object moved to <a href="/Policy_Detail.aspx">here</a>.</h2>
</body></html>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="ctl00_Head1"><title>
      Untitled Page
</title>
```
...

And in the second forward, we see the content:

```
<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/Policy_detail.aspx">here</a>.</h2>
</body></html>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="ctl00_Head1"><title>
      Name Change
</title>
```
...

Since the browser never displays the content from the second <html> element in each forward, this like the server may be building two additional excess pages inside the forwards which the user never sees, as their browser will automatically navigate away from them. This may be the result of a redirect exception being caught, when it was intended to cancel the remainder of the page, or it may be a result of a configuration setting on the web server.

I hope this test has helped to give you some insight into your application. If you find that you have any questions, please do not hesitate to ask, and I will be happy to elaborate on these results in further detail! If there is anything further we may do to be of service, please let us know.