So Much Static

Whenever someone asks, “How does Tomcat perform?” A significant percentage of the public believes Tomcat isn't good at serving static files. Although this was true with Tomcat3, Tomcat 5.0.19 and 5.5.4 have made great strides. Rather than attempt to prove Tomcat beats server X, I feel it's more important to understand the performance characteristics.

What does “performance characteristics” mean? To put it plainly, it's how Tomcat performs under different conditions. I used Jmeter to run the a series of benchmarks using Tomcat 5.0.19, 5.5.4, Apache 1.3.3 and Apache 2.0.50. To verify the results, I also used Apache AB.

Test Scenario

Jmeter
Test Series 1
File sizes: 1k, 5k, 10k, 20k, 40k, 80k, 160k
Threads: 5, 10, 15, 20, 25, 30

Test Series 2
Apache AB
File sizes: 1k, 10k
Threads: 100, 150, 20

Tomcat5.0.19
jdk1.4.2
jdk5
jdk5 + “-server”

Tomcat5.5.4
jdk5
jdk5 + “-server”

Hardware environment #1

Server:
Redhat Fedora Core 1
AMD2ghz
1Gb RAM
100Mb ethernet

Client:
Gateway 450 Laptop
Windows XP Pro
1.4ghz Pentium M/Centrino
1Gb RAM
Jmeter nightly build
jdk1.4.2
100Mb Ethernet

Sun Netra X1
Solaris 8
400mhz Ultrasparc 2
768Mb RAM
Apache ab 1.3
Dual 100mb Ethernet

Network:
Linksys 16 port 10/100 Switch

CAT5 cables

Tim's Environment #2

Dual 400mhz RISC HPUX servers
Tomcat 5.5.7, jdk5
100mbit network

Methodology and Environments

The primary test environment is my home development environment. Although it is getting a bit old, the results are still valid. Tim Funk was kind enough to run the Apache ab tests at work.

There two sets of tests. The first set attempts to measure the performance in terms of requests per second and kilobytes per second as the file size and concurrent clients increase. The data from these tests shows how performance degrades. The second set of tests shows how tomcat performs with a large number of concurrent clients. This measures how well the server responds to a sudden spike in load.

Results

Unless stated explicitly, the graphs and tables are for the environment #1. The first graph shows how tomcat performs using different VM's and settings. One interesting difference between 5.0.19 and 5.5.4 is “-server” option does not improve the throughput for 5.5.4. I had a discussion with Remy about this and my guess is the additional optimizations and ehancements in 5.5.4 negate the benefits of “-server”. With tomcat 4.0.x and 4.1.x, the improvement in performance was approximately 25-30% with “-server” in 2002. Although I don't know the exact changes in catalina and coyote responsible for the improvement, the results indicate 5.5.4 is more efficient in terms of memory and thread usage.

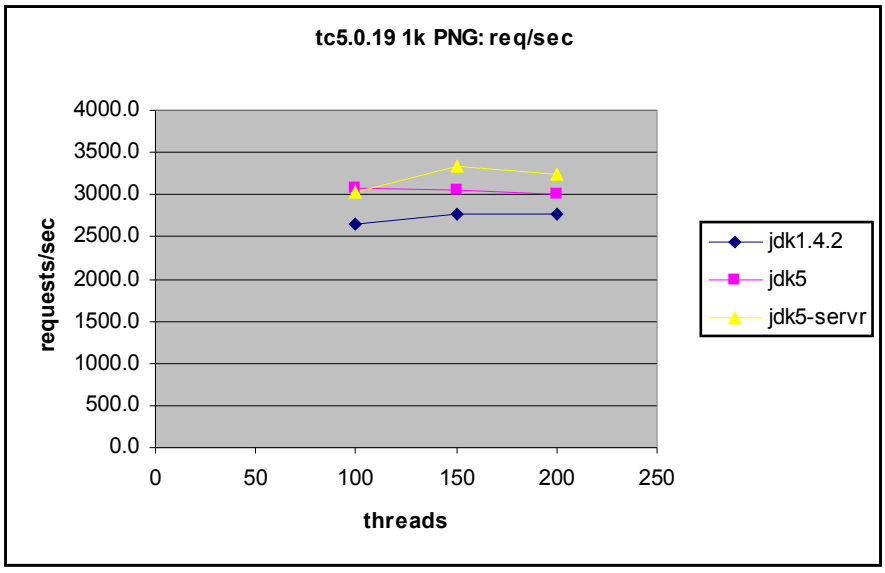

Chart A: ab test series 2

	100	150	200
jdk1.4.2	2652.0	2761.2	2762.6
jdk5	3076.6	3058.6	3006.6
jdk5-servr	3041.1	3348.3	3234.2

Table A: ab test series 2

Running TC5.0.19 with jdk5 and “-server” is roughly 15% faster than running jdk1.4.2 in client mode. That's pretty good news for those running 5.0.x release and want a boost. There's a catch though. You have to make sure your webapps run under jdk5.

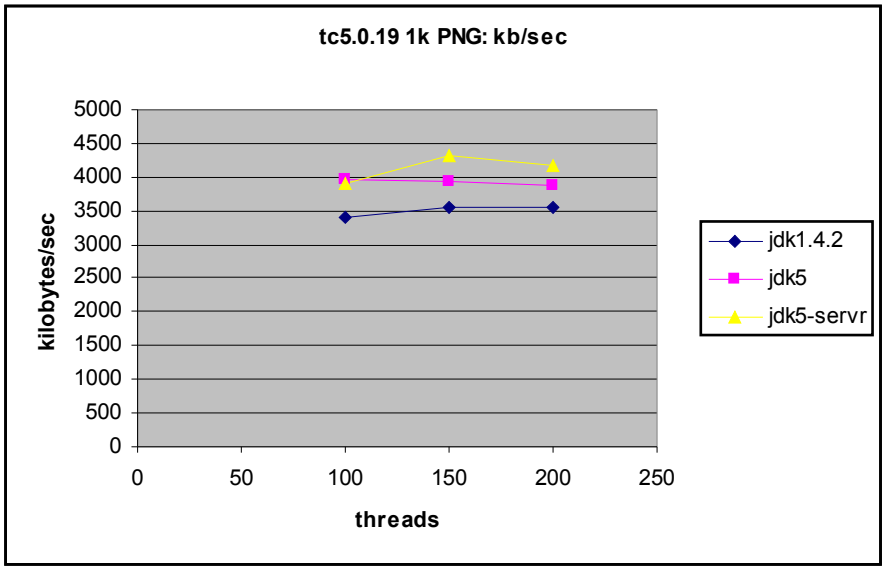

Chart B: ab test series 2

	100	150	200
jdk1.4.2	3410.29	3550.71	3552.43
jdk5	3956.41	3933.22	3866.16
jdk5-servr	3910.69	4305.74	4158.97

Table B: ab test series 2

Requests per second and kilobytes per second for TC 5.5.4. The results are show a similar pattern to TC 5.0.19.

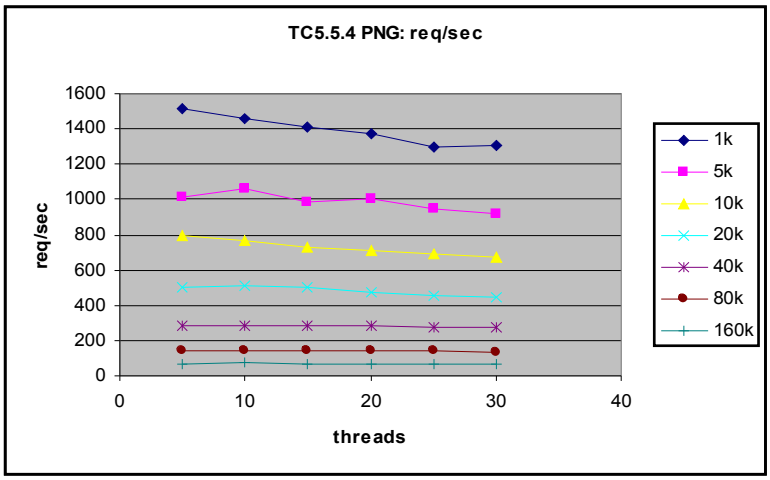

Chart C: TC5.5.4 Jmeter test series 1

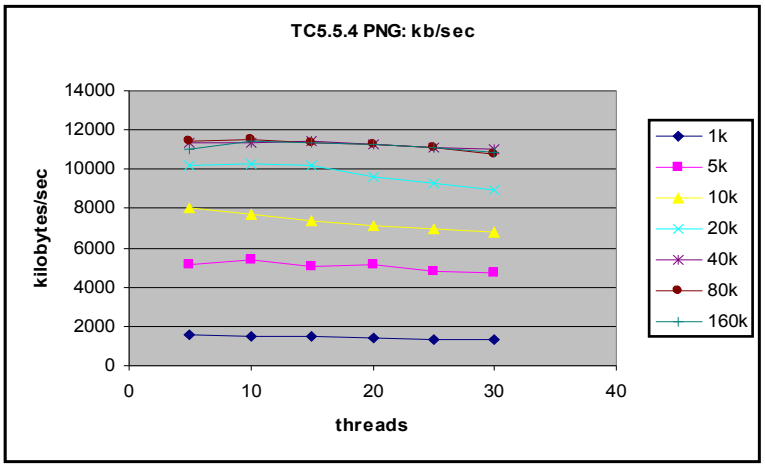

Chart D: Jmeter test series 1

Test results for Apache 1.3.3 and 2 show similar patterns. Nothing unusual to report here.

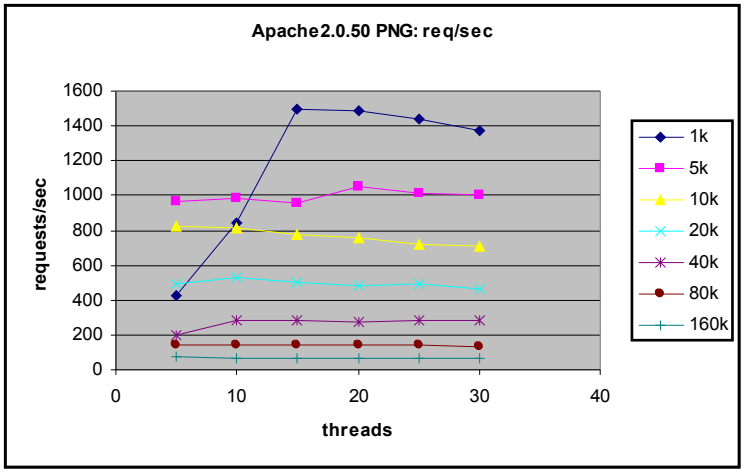

Chart E: Jmeter test series 1

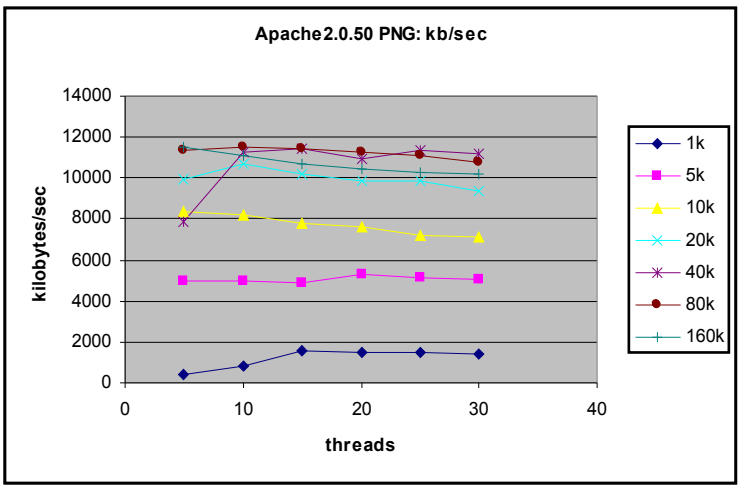

Chart F: Jmeter test series 1

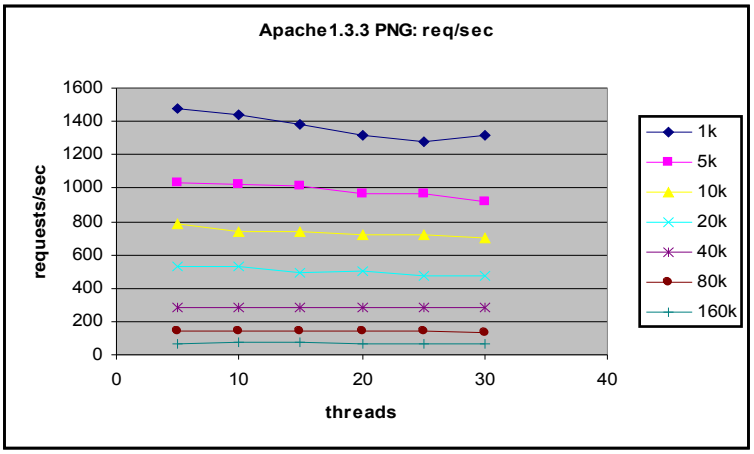

Chart G: Jmeter test series 1

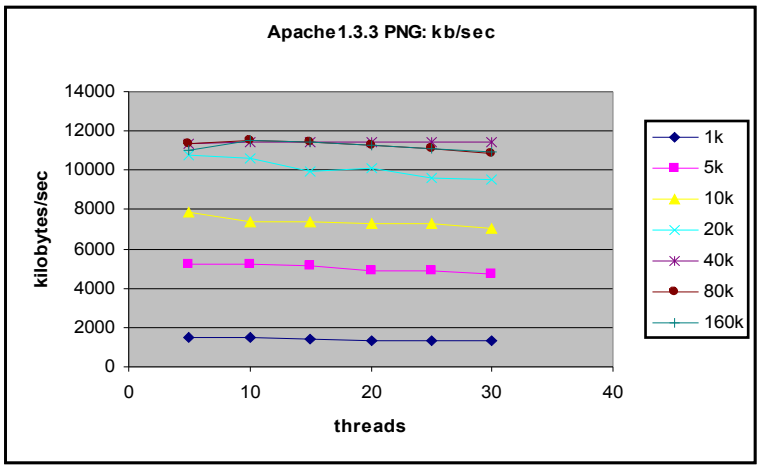

Chart H: Jmeter test series 1

I'm sure everyone wants to know how Tomcat compares to httpd, so here it is.

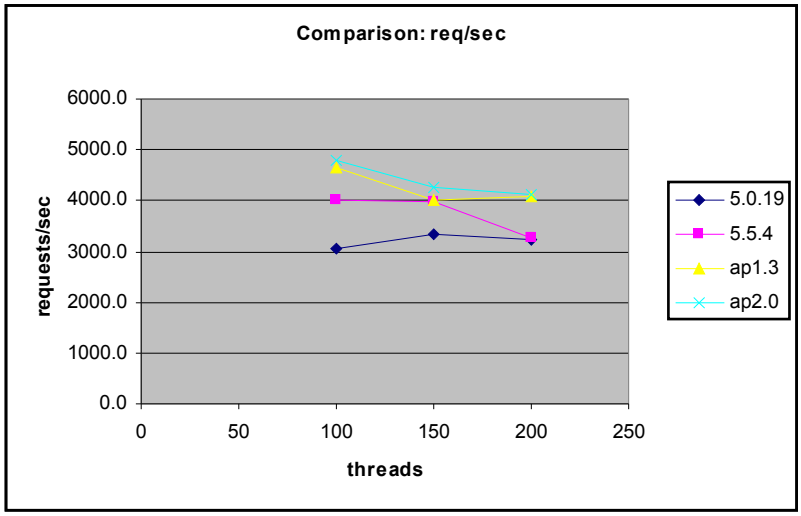

Chart I: ab test series 2

	100	150	200
5.0.19	3041.1	3348.3	3234.2
5.5.4	4004.49	3969.62	3266
ap1.3	4647.06	4027.49	4078.22
ap2.0	4798.92	4254.96	4110.83

Table I: Netra X1 Client

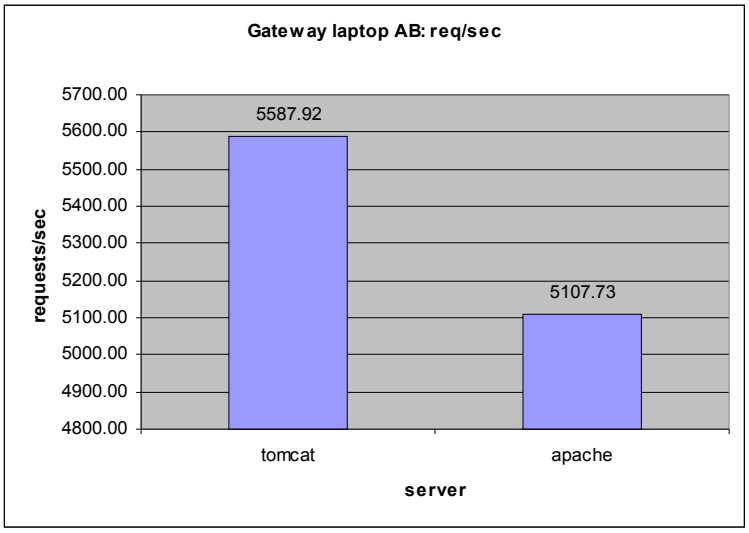

Chart J: ab test series 2

	tomcat	apache
gateway	5587.92	5107.73
HPUX	5699.14	
Netra X1	4004.49	4798.92

Table J

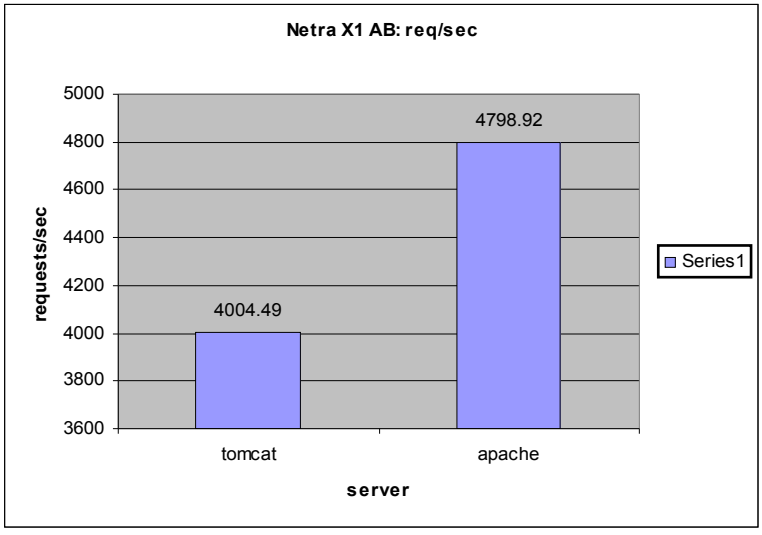

Chart K: ab test series 2

	%
gateway	16.4
hpux	18.8
netra X1	-16.6

Table K: Compared to Apache2 max throughput

For 1k PNG, TC 5.5.4 is roughly 16% slower than Apache2 using Netra X1 as the client, which manages to reach 4798 requests per second. In contrast, when the client is HPUX or the Gateway laptop, TC 5.5.4 is 16% faster than Apache2. The netra X1 is an old rackmount server and is about 4 years old, so it's unable to max out either Apache2 or TC 5.5.4. The results from HPUX and laptop should be more accurate. For some odd reason, the laptop couldn't run apache2 ab with more than 50 clients. With 100, 150 or 200 clients on the laptop, ab wouldn't run and gave me a cryptic error message. I didn't bother investigating the cause. If someone knows how to get around this problem, I will re-run the ab tests with 100, 150 and 200 clients. Using Jmeter on the laptop, I was able to simulate 100, 150 and 200 clients, but Jmeter is unable to max out the server. The charts and results for 100, 150 and 200 clients aren't included in the article, but they are in the excel file.

Although I could run more tests and figure out theoritical max throughput with multiple clients, very few websites keep their HTML to 1k. If we look at google's homepage, it is 1.11k of HTML and 8.36k for the logo.

	5	10	15	20	25	30
1k	1513.3	1457.9	1413.1	1375.4	1298.8	1309.3
5k	1010.7	1056.6	987.4	1006.1	947.4	921.2
10k	796.3	769.9	729.9	708.7	695.4	676.1
20k	503.8	506.9	504.3	473.9	459.1	440.9
40k	282.7	284.2	284.8	281.5	277.2	275.4
80k	142.7	143	141.3	140.3	137.8	133.6
160k	68.6	71.6	70.9	70.3	69.4	67.9

Table L req/sec: TC5.5.4, JDK5, Jmeter, Laptop client

If we look at Yahoo's homepage, it is 11.1K of HTML and includes 20 images. The icons on Yahoo's homepage are smaller than 1k, but largest is 11.3K. In the case of Yahoo, they get on average 1.9 billion page views (http://public.yahoo.com/~radwin/talks/one-year-of-php-oscon2003.htm), so every byte they shave is money saved on bandwidth. For a site like yahoo that gets millions of page views per day, it makes sense to host all the images on a dedicated image server with gigabit ethernet, or use a service like Akamai.

Chart D shows the bandwidth maxing out around 11Mb/sec with a 40K png. As the image size increases, the IO throughput remains the same, while the requests/second drops. The maximum throughput for 40K png is approximately 280-290 requests/sec regardless of the server. The only way to get higher throughput is to use gigabit ethernet or multiple ethernet ports. Realistically, most ISP only provide 10mbit of bandwidth unless you happen to be Yahoo, Google, or AOL. The bottom line is that Apache and Tomcat won't be your bottleneck for serving static files. In a “real-world” setup, the actual bandwidth to your servers will probably be closer to 5mbit/sec in bursts. Sustained throughput will most likely be closer to 2mbit/sec for a second/third tier ISP. Full 100mb bandwidth is available from backbone providers, but it won't be cheap. Keep in mind that real world conditions are very different from a LAN and packet loss can easily exceed 50%. Unfortunately, I don't have the resources to test tomcat with real modem connections to simulate “real-world” congestion. If someone has those resource available to them and would like to run the tests, I will gladly add those results to this report.

Before I conclude the report, lets look at what 5K requests/sec means.

5K req/sec x 60 seconds/min X 60 min/hr x 24hr/day = 432,000,000 req/day

That would translate to over 400Gb of data transfer per day. I might be going out on a limb here, but I'm guessing only large corporations consume that much bandwidth. In those cases, the production environment is a cluster of servers with redundancies and backup systems. In large deployments, the question isn't “should we use apache or tomcat for static files?” Websites like Yahoo, CNN, Espn and MSN use hosting services like Akamai.

Conclusion

For those who want to view the entire excel spreadsheet, it is available here http://cvs.apache.org/~woolfel/tc_results.html. The testplans used for the benchmark are also available http://cvs.apache.org/~woolfel/testplans.zip. After 400+ benchmarks, what does this say about Tomcat?

First off, I'm bias in favor of Tomcat. Based on these results, I can state with confidence Tomcat 5.5.4 has made great strides since Tomcat 3.3. Tomcat 5.5.4 is faster, more reliable and more efficient than previous releases. For those who wonder “can tomcat handle static files?” My opinion is yes. If you only have a single server co-located at an ISP and can't afford a dedicated image server, Tomcat will work just fine. For sites that need high performance/high availability, the best option is to setup dedicated Apache2 for the static files. This setup allows tomcat to focus on generating dynamic content, instead of clogging the network IO. Hopefully these results help dispell the myth that Tomcat 5.5.4 can't handle static files efficiently. If you find errors in the article, or would like to contribute additional results, please feel free to email woolfel AT gmail DOT com.