Simulating WWW Workloads:
Throughput and System Resource Utilization

Diane T. Lark
Martha Evens
William Lidinsky
Illinois Institute of Technology

ABSTRACT

World Wide Web traffic exhibits self-similarity over a wide range of traffic intensities. Therefore, traffic generators capable of generating such traffic are required to simulate such real-world conditions. One such traffic generator is SURGE (Scalable URL Reference Generator). The developers produced evidence that SURGE generated network traffic with higher degrees of self-similarity than that of another tool, SPECWeb96.

Workload generators are of particular interest to those who wish to implement systems that are able to handle particular loads. A process used to determine hardware system sizing is to conduct benchmarks of the anticipated workloads followed by an analysis of actual resource utilization under those loads. This document reports on the Apache Web server throughput and resource utilization for workloads generated utilizing SURGE. We also assess the resources needed by the client to generate the workload.

INTRODUCTION

Given the difficulties of predicting Web site popularity prior to installing hardware, it is often desirable to benchmark the expected workload in order to estimate proper system sizing. Several workload generators are available for these purposes. Notable work by several research teams [2,3,4] has shown that those tools prove to be inadequate. Leland, et al. [3] produced evidence that Ethernet traffic exhibited self-similar statistical properties. This is often referred to as long-range dependence, or the Joseph effect. Paxson and Floyd [4] established that the Poisson distribution was inappropriate for network modeling due to the long-range dependence observed in network traffic. Bestavros and Crovella [2] discovered that HTTP traffic exhibited self-similar properties. Barford and Crovella [1] introduced SURGE in 1998 and substantiated that their workload generator was capable of simulating workloads that generated network traffic with statistical self-similarity properties. Our goal was to evaluate system resource utilization of the Web server given certain SURGE workloads and to assess the client system requirements for generating these loads.

To this end, we defined and generated SURGE workloads and examined server throughput, client response times, and both client and server system resource utilization for processor, memory, and network.

Our efforts have resulted in verifying what one might expect in the area of throughput as well as illustrating a relationship between Web server configuration and throughput. Total data transferred increased linearly as a function of load and time. The mean number of HTTP Gets processed per second proved to increase linearly as a function of load until the point at which the load exceeded 15% of the Maxclients Web server parameter where it peaked out. Maxclients is an Apache configurable directive. Mean HTTP Gets per second remained constant for a consistent load over increasing time frames. We have also shown that workload definition had little impact on the mean number of HTTP Gets processed per second but could significantly impact the total amount of data transferred.

With respect to system resource utilization for the Web server and workload generators, in both cases memory was consumed immediately, yet swap utilization never exceeded 10% for any load. Both server and client used processing power sparingly. Processor utilization rose linearly as the load and the mean number of HTTP Gets processed, or from the perspective of the client, requests issued was increased. Web server disk utilization was intensive with a maximum number of 65 disk transactions per second (TPS) for the default workload and 143 TPS for a modified workload definition. Disk activity on the workload generator was nominal for the 1GHz clients.

This paper describes the environment used to execute the simulations and the results of the system resource utilization analysis.

DESCRIPTION

The next sections describe the experimental apparatus, review the SURGE workload generator, and describe the throughput of the Web server, client response times, and utilization for processor, memory, and network bandwidth.

Experimental Apparatus

The experimental hardware apparatus consisted of five systems. There were three 1 GHz Pentium III HP Pavilion 7940 platforms and two 66MHz Pentium II ACP platforms. One Pentium III had 384MB RAM and the other two had 128MB RAM. Each of the two Pentium II systems had 64MB RAM. All systems ran Red Hat Linux. One Pentium III system functioned as the Apache Web server and Network Time Protocol server. The other four systems served as SURGE workload generators. These five systems were connected to an 8-port Linksys switch to form an isolated LAN.

SURGE Workload Generator

SURGE (Scalable URL Reference Generator) was developed by Barford and Crovella to overcome a deficiency noted in other tools like SPECWeb96 and Webstone. SURGE was shown to produce high degrees of self-similarity in network traffic under heavy loads. This achievement was attributable to a number of workload generator parameter distributions and a non-fixed request generation process that created new threads for every request. Unlike earlier workload generators, the intensity of the workload and the number of threads grew proportionally resulting in a corresponding increase in the number of simultaneous connections. This meant that as the number of requests per second increased the number of threads used increased as well. With an independent parameter controlling the idle periods between active request periods, the time between the requests would not decrease.

The developers of SURGE introduced the concept of a user-equivalent (UE) to capture the behavior of a real Web user. A user-equivalent was an ON/OFF process that exhibits long periods of activity followed by long periods of inactivity. Request rates and idle periods were based according to a distributional model. A variable request rate source was used. The thread idle period follows the associated distributional model and the numbers of threads increases as the number of requests are increased. In contrast, forerunners of SURGE used a constant rate source. The number of threads was held constant and the request rate was held constant. This caused the idle period to be greatly reduced under higher loads. Idle, or OFF, times were not preserved. As the workload intensified, the idle times of threads exhibited lower variability in demand. This failed to generate traffic with high degrees of self-similarity.

The SURGE workload can be characterized by six variables. They include: 1) server file size, 2) relative file popularity, 3) temporal locality of reference, 4) request size, 5) idle periods, or OFF times, of individual users, and 6) embedded file references. Each followed its own distributional model. The default values for SURGE are described in Table 1.

Table 1. SURGE Default Distribution Parameters.

Distr	Model	Probability Density Function	Parm
File Sizes – Body	Lognormal	$p(x) = \frac{1}{x\sigma\sqrt{2x}} e^{-(\ln x-\mu)^2/2\sigma^2}$	μ=9.36; σ=1.32
File Sizes – Tail	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	κ=9300 α=1.0
Popularity	Zipf	$P = kr^{-1}$	
Temporal Locality	Lognormal	$p(x) = \frac{1}{x\sigma\sqrt{2x}} e^{-(\ln x-\mu)^2/2\sigma^2}$	μ=1.5; σ=0.80
Request Sizes	Lognormal	$p(x) = \frac{1}{x\sigma\sqrt{2x}} e^{-(\ln x-\mu)^2/2\sigma^2}$	μ=7.88; σ=1.34
Request Sizes	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	κ=3558; α=1.177
Active Off Times	Weibull	$p(x) = \frac{bx^{b-1}}{a^b} e^{-(x/a)^b}$	a=1.46; b=0.382
Inactive Off Times	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	κ=1; α=1.5
Embedded References	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	κ=1; α=2.43

Combining the above distributional models into a single representative workload exhibiting those characteristics introduced two problems [1]. The first was how to generate the appropriate number of requests for the desired file sizes. The second was how to generate a request stream that exhibited temporal locality.

The first problem was handled by Barford and Crovella [1] through a method designed to match requests to files and to generate a representative output sequence. The matching problem involved the popularity, file size, and request size distributions. No matter how many clients were involved with issuing requests, the distribution models for popularity and file size needed to be generated according to the defined distribution model parameter values. Success was determined by the difference between the desired number of references to a file size and the actual number of references to that file size. Two methods were assessed by Barford and Crovella [1]. Splitting the head of the distribution from the tail of the distribution produced optimal results when matching. Matching the distribution tail resulted in close fits. Large errors were not introduced in the body of the distribution

The second problem was addressed by placing file names in a stack and defining a small window in which each file was assigned a weight with a value proportional to the outstanding requests for that file [1]. The file at the top of the stack was based upon the probabilistic weight of the files within the window. The file sequence generation produced the appropriate file distribution, and the stack distance values followed the lognormal distribution. However, distributions would be less accurate when numerous clients were configured since clients could not share a common list without associated synchronization overhead. SURGE managed this situation by generating independent file name sequences and their associated stack distance properties. The lognormal distribution output values were scaled according to the number of clients. This produced the same stack distance distribution as the single client.

RESULTS

The following sections report on the server throughput represented by two metrics, total bytes transferred and HTTP Gets per second, and system resource utilization for both the server and the client. Throughput analysis was performed to verify linearity between the amount of data transferred and an increase in time or user load.

Server Throughput

Our first step was to verify that the total bytes transferred increased linearly as a function of load and time. The default SURGE workload was used for two different series in order to make this assessment. The first series included eight tests where the user load was increased from 300 UEs to 1350 UEs with each test being executed for 60 minutes. The second series consisted of five tests with the user load held constant at 600 UEs and the time was increased from 60 minutes to 300 minutes.

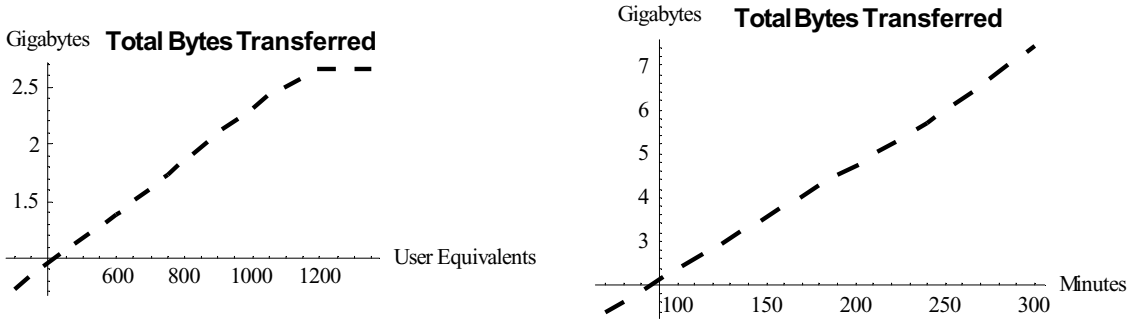


Figure 1. Total bytes transferred as a function of load and time. The graph on the left reflects an increased load and the graph on the right reflects increased time.

The graph on the left in Figure 1 shows that as the number of users increased to 1200, the number of bytes transferred increased linearly until it limited at 1200 UEs. This limit was attributed to the Apache Web server Maxclients configuration of 1024. As the number of user's exceeded approximately 15% of that parameter, the request rate exceeded the server's ability to respond thus reducing throughput. This limitation was observed with other workload definitions with identical results. The graph on the right in Figure 2 illustrates the linear relationship between bytes transferred and time.

This was further verified by comparing default workload tests with two workload variations. One workload definition was similar to that of a real-world example as described by Crovella and Bestavros [2]. SURGE workload shape parameter values for File Size Tail and Request Size Tail distributions were set to 1.06 and 1.16, respectively. This was a 6% increase in the file size distribution shape parameter and a 1% decrease in the request size distribution shape parameter. For the purposes of comparing a second variation, SURGE workload parameters were modified to 1.06 and 1.21, respectively. Here, the file size distribution was increased by 6% and the request size distribution was increased by 3%.

A series of eight tests with each test running for 60 minutes was executed for each of the two-workload variations. The vertical axis is gigabytes transferred per second. The user loads ranged from 300 to 1350 UE's. The graphic on the left in Figure 2 illustrates this comparison. The default workload is the dashed line, the first workload variation is represented by the multi-sized dashed line, and the second variation is the solid line. To assess the impact over time, the first workload definition was used to execute three additional series. Time was increased from 60 to 240 minutes for 600, 900 and 1200 user loads. These results are depicted on the right of Figure 2. The bottom dashed line is the 600 user series, the middle multi-sized dashed line is for the 900 user load, and the upper solid line is the 1200 user load.

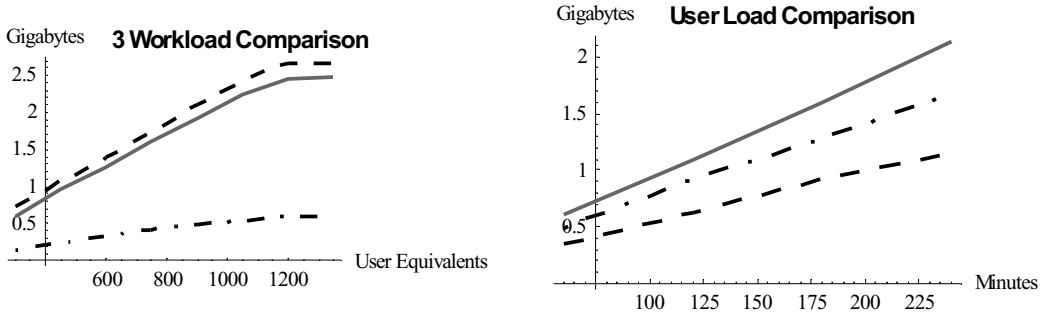


Figure 2. Comparison of 3 simulated workloads and comparison of 3 simulations with varying user loads.

Both graphs in Figure 2 further support the claim that data transferred are a function of user load and time. The graph on the left shows that the data transfer rate peaks at 2.5 gigabytes transferred per second with a user load of 1200. Configuring the Maxclients parameter for the Apache Web server influences the overall throughput given peak loads. The peak load was reached at 1200 UE's, or approximately 115% of the Maxclients value of 1024. The left graph also exhibits the impact of varying the workload parameters. There was a significant decrease in bytes transferred with the first workload definition when compared with the default definition workload. An increase of 6% in the file size distribution shape and a 1% decrease in the request size distribution shape produced a considerable decrease of between 65% and 77% in bytes transferred, depending upon the load. The second workload variation where the file size was increased by 6% and the request size was increased by 3% resulted in an approximate decrease of 17% in bytes transferred. Understanding the implications related to changing parameters requires further investigation.

The graph on the right of Figure 2 illustrates in impact on data transferred given increased load and time. As you would expect, increasing the load 1-1/2 times or two times emitted comparable increases in transferred data.

Throughput was also measured by the mean number of HTTP Gets serviced per second (gps). The series based on the default workload definition with a test period of 60 minutes each for 300 to 1350 users illustrates the linear relationship between bytes transferred and user load. The results are shown on the left graph in Figure 3. The default workload definition also was used for two additional series where the user load was held constant at 600 for one series and 900 for the other series with the job duration ranging from one to five hours. This was done to understand better the impact of time on throughput.

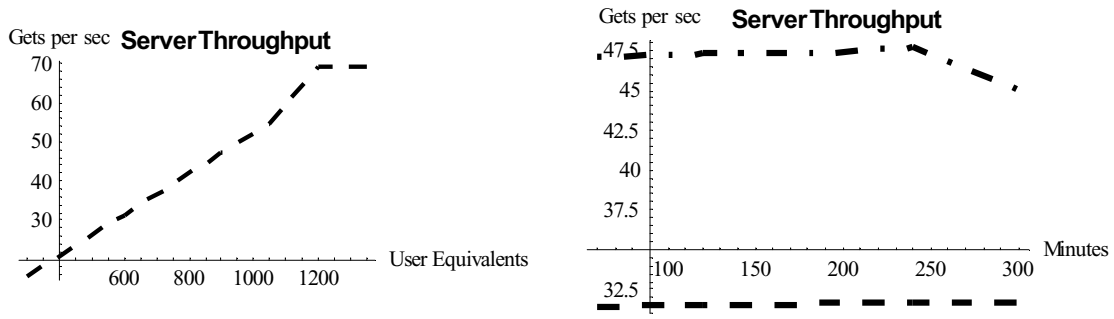


Figure 3. Gets per second (Server Throughput). Throughput of default workload with varying user load and job duration.

The left graphic in Figure 3 shows the linear relationship between the user load and the mean number of HTTP Gets that the server is able to service within a 60 minute time frame. Similar to the amount of data transferred, a limit was reached at approximately 70 gps and throughput peaked at 1200 users. The graphic on the right of Figure 3 illustrates that for a given workload, throughput (gps) was time invariant.

A comparison of gps was made of two other workload definitions. The default workload is represented by dashes, the real-world workload is represented by the multi-size dashed line, and the alternate workload variation is the solid line. Figure 4 shows the results.

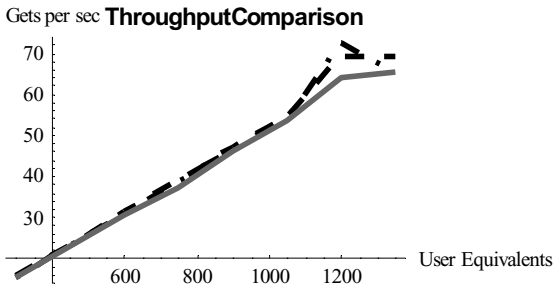


Figure 4. Server Throughput Gets executed per second.

The maximum gps achievable was between 65 and 70 for all three series. The peak was reached at the 1200 user load. Figure 4 indicates that the modifications to the workload definition had little impact on gps throughput. The first workload definition variation emitted a 5% increase in gps and the second variation tracked the default identically.

Server System Resource Utilization

The left graphic of Figure 5 illustrates the server processor mean utilization for the series based on the default workload consisting of eight tests ranging from 300 UEs to 1350 UEs. Each test was executed for a duration of 60 minutes. It is placed side-by-side with the graphic from Figure 3 to establish the relationship between throughput and processor utilization.

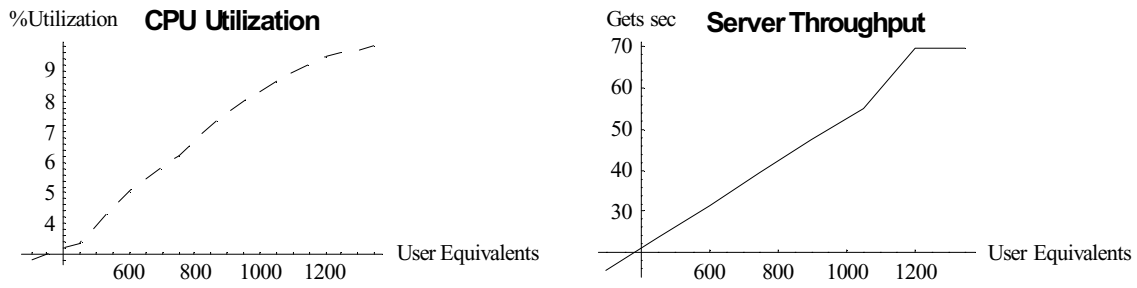


Figure 5. Mean processor utilization of the server with increasing user loads compared with the HTTP Get's processed per second over increasing user loads.

As the user load was increased, processor utilization increased linearly. Applying linear regression elicits the equation of y=0.63155 + 1.09x. Server processor utilization did not exceed 10% using the default SURGE workload parameter settings with 1350 UEs. As you can see from comparing the vertical axis of both graphs, a ratio of 7 to 1 was observed between server throughput and processor utilization. Table 2 lists the minimum and maximum processor utilization values for each of the eight tests.

Table 2. %Processor Utilization

User Load	Minimum	Maximum
300	0.2	12.2
450	0	14.4
600	1	19.2
750	1.4	21
900	2	49
1050	0.2	33
1200	4.6	28
1350	1.8	27.4

Following are the graphs that illustrate memory and swap utilization for the tests beginning with a 300 UE load to a 1350 UE load.

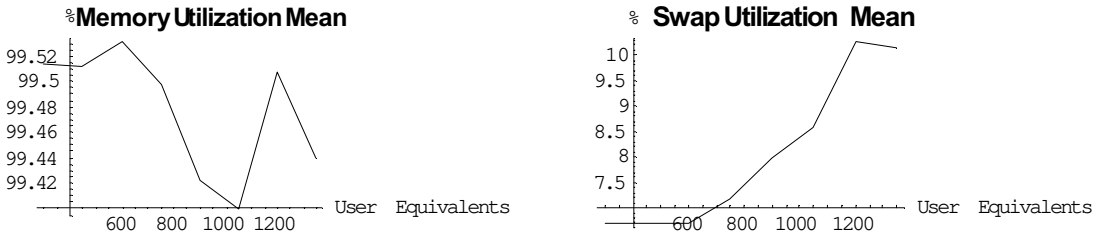


Figure 6. Mean server memory and swap utilization with increasing user loads.

Memory utilization was relatively constant at 99.5% for all tests between 300 UE's and 1350 UE's. The mean Swap space used was between 7-10%. Based on these results, four clients simulating 300 UEs or more nearly consumes all of the available virtual memory. Assuming of course that one would not exceed 20-30% of swap area and all of physical memory.

Figure 7 shows the result of our analysis of the number of disk transactions per second. It displays the mean number of server disk transactions per second.

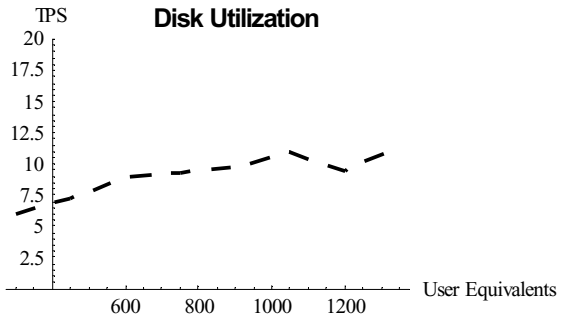


Figure 7. Mean Server Disk Utilization.

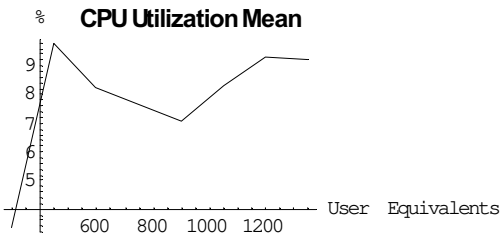
Table 3 lists the minimum and maximum number of transactions per second for each of the eight tests.

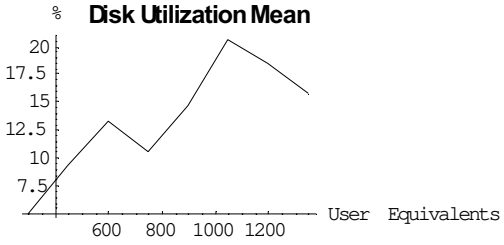
Table 3. Disk Transactions per Second

User Load	Minimum	Maximum
300	0.4	26.6
450	0.4	46.2
600	0.4	50.8
750	1.2	46.6
900	0.4	54
1050	0.6	65.2
1200	0.4	49.8
1350	0.6	42.6

Following are the server utilization results for the real-world workload. Eight tests were executed for one hour each with a user load ranging from 300-1350 UE's.

Starting clockwise from the upper left in Figure 8 is the CPU, Disk, Swap and Memory Utilization means.





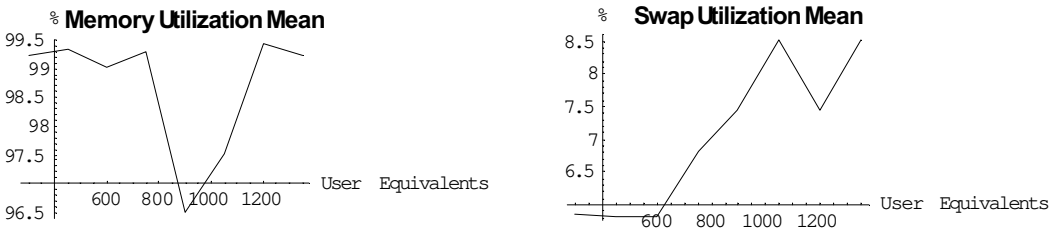


Figure 8. Real-world workload example.

Table 4 lists the minimum and maximum values for server processor and disk utilization.

Table 4. Real-World example

User Load	CPU % Util		Disk TPS	
	Min	Max	Min	Max
300	0.4	17	0.4	38.2
450	0	15.2	0.4	110.6
600	0	18.4	0.4	116.2
750	0	23	0.4	121.2
900	0	32.8	0.4	98.4
1050	0	28.8	0.4	134
1200	0.2	20.6	0.4	132
1350	0	29.8	0.4	142.8

Following are the server utilization results for the real-world workload. Seven tests were executed for one hour each with a user load ranging from 300-1350 UE's. Starting clockwise from the upper left in Figure 9 is the CPU, Disk, Swap and Memory Utilization mean.

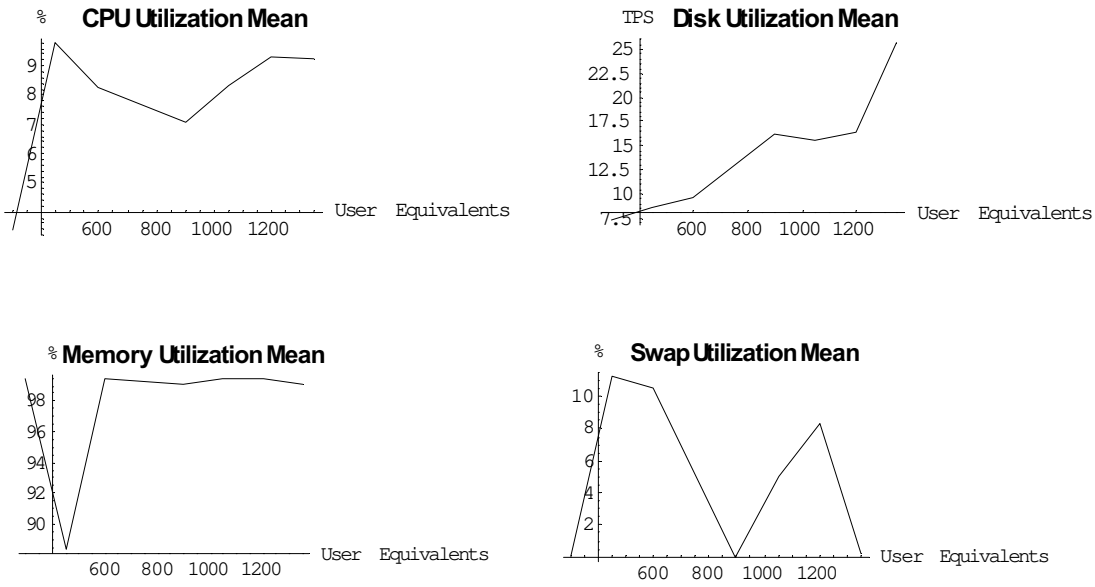


Figure 9. Real-world variation workload example.

Table 5 lists the minimum and maximum values for the percentage utilization of CPU and the disk TPS for all seven tests.

Table 5. Real-World variation example

User Load	CPU % Util		Disk TPS	
	Min	Max	Min	Max
300	0.2	15.2	0.4	31.2
450	0	24.2	0.4	25.2
600	0	21.4	0.4	64.6
900	0.4	23.8	0.4	51
1050	0	29	0.4	67.8
1200	2	25.8	0.14	75.8
1350	0	23.62	0.4	84.4

The results depicted in Figure 8, 9 and Table 4, 5 illustrates variability in simulated loads using SURGE. This variability can be attributed to the use of independent parameter distributions. Increasing the user load for the same workload definition causes the resultant distribution to vary from the expected workload. In general, as demand increase, disk utilization trends upward while other system resources remain relatively constant once it peaks.

Client System Utilization

This section reports the system resource utilization from the client perspective using SURGE with varying user loads. Four client systems were involved in the tests. Two systems had a single 66MHz processor and 64MB RAM. Two systems had a single 1GHz processor and 128MB RAM. Executing the tests using the default SURGE parameters required that each client generate a fraction of the overall UEs. Where the server received requests from UEs, each client generated 75 UEs of the 300 UE total. Memory, processor, and disk usage were examined for the default SURGE workload.

Processor utilization for the clients is depicted in Figure 10.

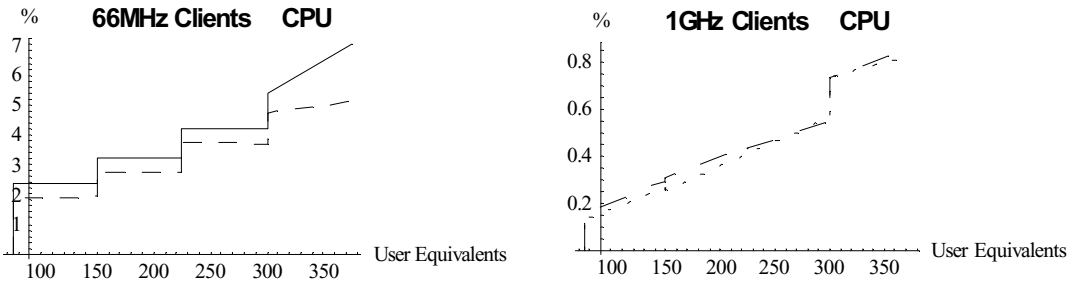


Figure 10. Mean processor utilization for SURGE clients.

Processor utilization for the 66MHz systems was comparable with a peak of approximately 7%. The 1GHz systems peaked at nearly 1%. Comparing the average processor utilization over the increasing application of User Equivalents shows that the ratio of the slow system to the fast system was approximately constant at seven to one. The ration of the processor speeds was fourteen to one.

Figure 11 shows the mean memory utilization for the clients.

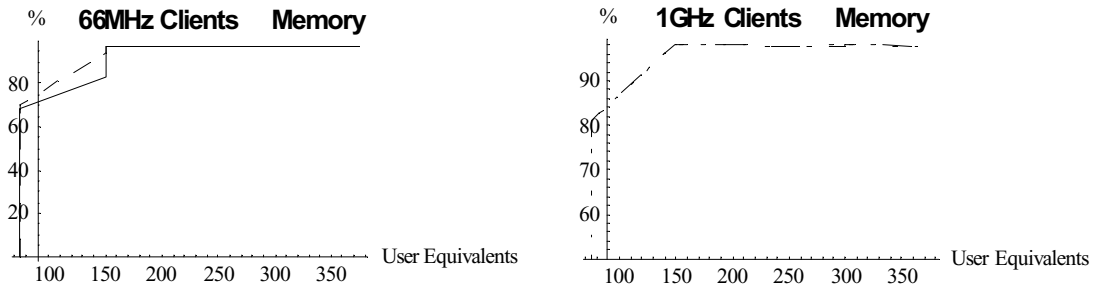


Figure 11. Mean memory utilization for SURGE clients.

The memory utilization percentage for the 66MHz clients is shown on the left and the mean percentage utilization for the 1GHz systems is shown on the right of Figure 11. Both like-configured systems utilized memory identically. Memory was consumed at 99.5% on all clients when a 150 UE load was present.

Mean client disk activity was minimal during SURGE simulations. Figure 12 shows the mean disk transactions per second.

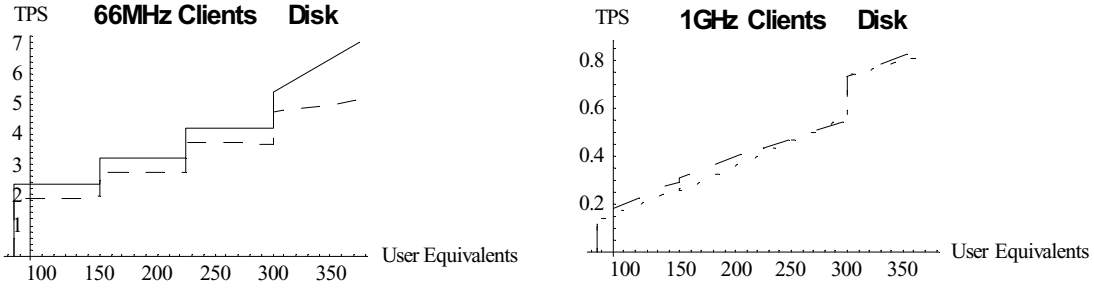


Figure 12. Client disk activity.

The 66MHz systems have an approximate mean of 7 disk transactions per second and the 1GHz systems mean approaches 1 disk transaction per second. Like processor utilization, there is a seven to one ratio between the slow and fast clients mean disk transactions per second.

CONCLUSION

As you might expect, the amount of data transferred increased linearly as a function of load and time. Throughput measured as Gets per second, however, increased linearly as a function of load while remaining constant over varying time frames. The value for the Apache Web server Maxclients directive influenced the mean number of HTTP Gets processed per second. The peak mean HTTP Gets per second was reached with a 1200 user load, or 115% of the Maxclients value. Less intuitive was the impact on Gets processed per second when simulating different workloads. The number processed was consistent for three different workloads.

Web servers are known to be memory intensive. We have found this to be true in our tests as well. Memory was nearly consumed on the server beginning with a load of 300 UEs. The SURGE clients proved to be memory intensive and they reached near full utilization at only 150 User Equivalents. Both server and clients used the processor frugally where utilization never exceeded 10% for any load. Based on this, it would seem that ten clients would be needed to stress the server processor. In other words, the server processor capacity could support approximately 3000 UEs. Further study would be required to verify this theory.

We did observe a 7 to 1 relationship between the mean number of Gets processed per second by the Web server and the percentage of processor utilization on the server. With respect to the clients, processor utilization was low. But interestingly, with regard to processor utilization, we observed a 7 to 1 relationship between the slow clients and the fast clients. The 66MHz systems have an approximate mean of 7 transactions per second and the 1GHz have an approximate

mean that approaches 1 disk transaction per second. As with processor utilization, there is a seven to one ratio between the mean disk transactions per second produced by the slow and fast clients.

Throughout this effort as well as other related areas of research, we have conducted hundreds of simulations. SURGE has proven to be a nicely behaved tool. The user, however, must understand it's proper use. This effort has uncovered other areas where further investigation is warranted. For example, a thorough analysis of the network adapter on both server and client would be interesting. Due to time limitations, it was not performed here. Another area would utilize SURGE to stress more fully the Web server. Hardware resource constraints were a limitation during this study.

Reference List

[1] Barford, P. and Crovella, M. E., "Generating Representative Web Workloads for Network and Server Performance Evaluation", *1998 ACM Sigmetrics Conference Performance Review Proceedings*, pp. 151-160, Madison, WI, June, 1998.

[2] Crovella, M.E. and Bestavros, A. "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes", *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1996.

[3] Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V., "On the Self-Similar Nature of Ethernet Traffic" (extended version), *IEEE/ACM Transactions on Networking,* Vol. 2, No. 1, pp. 1-15, February 1995.

[4] Paxson, V., and Floyd, S., "Wide Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, Vol. 3, No. 3, pp. 226-244, June 1995.

[5] Veitch, D., and Abry, P., "A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence", *IEEE Trans. Info. Th., special issue on Multiscale Statistical Signal Analysis and its Applications,* Vol. 45, No. 3, April 1999.

Biography

Diane Lark is a senior member of Hewlett-Packard IT where she is the HP Consulting knowledge architect. She has also provided Internet, system and network consulting. Prior to her employment with HP, she performed system administration and UX education roles at Motorola and Digital Equipment Corporation. She is a Ph.D. candidate at Illinois Institute of Technology (IIT). In 1991 she received her MSCS from IIT and in 1986 received her BSCS from the University of St. Francis.

Martha Evens is a Professor of Computer Science at Illinois Institute of Technology. She received an A.B. in Mathematics from Bryn Mawr, spent a year in Paris on a Fulbright Fellowship, received an A.M. in Mathematics from Radcliffe, and a Ph.D. in Computer Science from Northwestern University. She has published two books and more than two hundred and fifty papers and received over forty research grants.

William Lidinsky is a member of the faculty of the Computer Science Department at Illinois Institute of Technology. Prior to joining IIT he was the Head of the High Energy Physics Network Resource Center at Fermi National Accelerator Laboratory, supervised research groups at AT&T Bell Laboratories, International Harvester's Science and Technology Laboratory and Argonne National Laboratory. He chaired the IEEE802.1 Local Area Network Standards Working Group and was a member of the IEEE802 Executive Committee. He has chaired several professional conferences, published a number of papers, holds a number of patents and has consulted in his fields of interest. He holds an MSEE degree from IIT and an MBA from the University of Chicago. He is a senior member of the IEEE and the ACM.