Describing and Visualizing the Capacity of a System with Behaviour Uncertainties

Peter Maly[1], C. M. Woodside[1], G. K. Karam[2], Andrew Forrest[2]

1. Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, K1S 5B6
{maly, cmw}@sce.carleton.ca

2. AT&T Shannon Labs, 180 Park Avenue, rm E267, Florham Park, NJ 07932
karam@research.att.com
ajforrest@att.com

Abstract

User and system behaviour is difficult to predict for novel systems, and this affects the capacity of the system (the number of active users that can be supported with acceptable response delay). This leads to a range of values, in the form of a feasible or acceptable region for the potential capacity, conditional on the uncertain parameters. This work considers uncertainties in the delay between requests (user think time), network latency, and cache behaviour due to users' locality of reference. The acceptable region is shown to be bounded approximately by linear constraints which are easy to derive. This simple result is useful for sensitivity and scalability analysis, and appears to have been overlooked. It is applied to a web-based system for telephony, using VoiceXML for service ranging from interactive voice response, to voice-based e-mail.

1.0 Introduction

The capacity of an interactive system can be defined as the number of active users that can be supported with adequate delay performance. This number depends on the system, but also on *external factors* related to the network and the users' behaviour. This paper is motivated by a desire to convey the dependency on the external factors, in a simple and usable way. In a space of parameter values, including the number of users, a *feasible region* defines those values that give systems that meet the delay specifications. This work shows that for some parameters the boundaries of the region have simple properties that make them easy to establish, and uses them with a system that integrates telephony and web services.

Capacity planning methods such as described by Menasce et al [8], [9] or Jain [6] seeks to find a configuration with acceptable performance, subject to cost constraints. Acceptable performance may be defined by a set of service level agreements, which may also impose requirements on capacity. The steps for capacity analysis of client/server systems described in Chapter 5 of [8] are to characterize the workload and the environment, to use this information to build a performance model and a cost model, and to compare alternative solutions, based on cost and performance. A similar systematic guide is given by Hahn et al [5] specifically for business intelligence systems.

Variations in parameter values change the performance results from the model, and its outcome. They can be investigated by a sensitivity analysis of the performance results. For example in [4], Borowsky et al described capacity planning for storage subsystems with complex streams of operations. They studied the sensitivity of a performance bound to factors such as the heterogeneity of the separate streams of requests, and their degree of correlation.

The next step is to use the sensitivity analysis to describe a set of acceptable parameter values and their interactions in some way, but that has not been described to date. It is the goal of this paper.

2.0 Capacity and External Factors in Interactive Systems

Figure 1 describes the interactive systems considered here, with a set of N users sending requests and receiving responses across a medium which may introduce a round-trip latency L_2. Within the system is the application under study, which requests data from aux-

iliary components (e.g. a web server) across a Network, introducing another latency. We consider a single class of requests first.

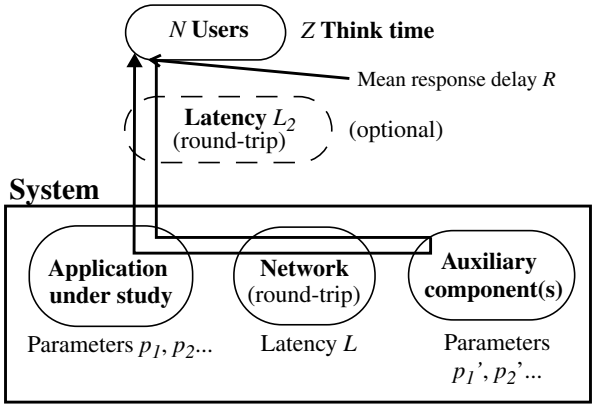


Figure 1. Class of system under consideration

Each request from the user to the system follows a *think time* with a mean of Z seconds, and the application has parameters p_1, p_2... which may affect delay. Parameters which we will consider later include

- a cache hit rate which is governed by the popularity of selections made by users, and their locality of reference in sequences of operations.
- the mean file size to be fetched across a network in order to respond to a users' request.

The application makes requests across a network which introduces a total round trip latency of mean L, to fetch data from the auxiliary components.

The capacity is defined as the largest N such that the mean response delay R satisfies a specified value R^* seconds:

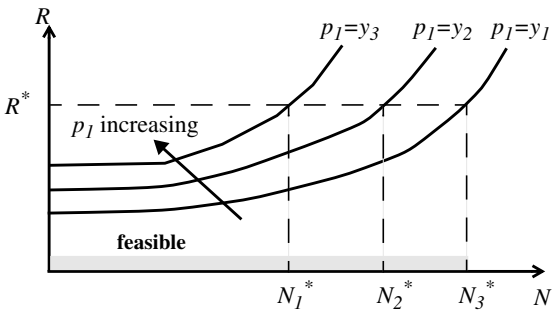
QOS constraint on delay: $R < R^*$ *sec*

The discussion can be generalized to multiple classes of requests, different specifications of response delay (such as 95% of response delays are less than R^*), and specifications on multiple QoS measures. These are however not considered in the present paper.

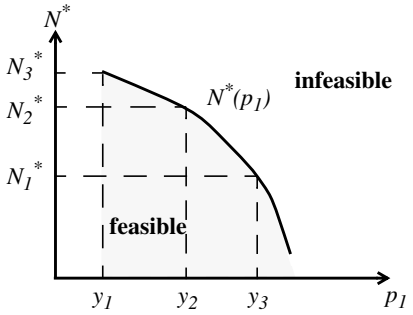
2.1 Capacity and the Effect of Parameters

Figure 2(a) shows how R might vary with N for certain values of some parameter p_1, defining a set of *feasible* values of N that satisfy the constraint $R < R^*$. The capacity N^* is defined as the largest feasible value of N.

The shift in perspective we wish to adopt in this work is to map the curves in Figure 2(a) into the boundary of an *acceptable or feasible region* in a



(a) Effect of parameter p_1 on R

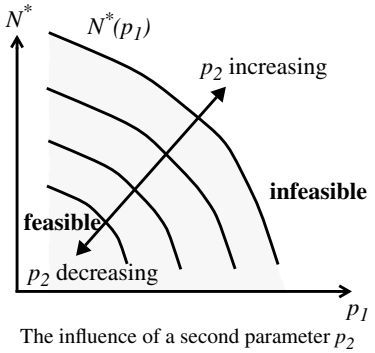


(b) Dependence of $N^*(p_1)$ expressed directly

Figure 2. The effect of a typical parameter p on the maximum feasible user population N^*

parameter space, as indicated for the parameter vector (N, p_1) in Figure 2(b). Then if another parameter p_2 varies, the boundary moves as shown in Figure 3.

The value of a plot like Figure 3 is that a great volume of exploration results can be summarized compactly. The direction of influence of a parameter is shown by how the feasible space expands or contracts. Sensitive parameters are obvious: contour lines close together show insensitivity, while far apart they show high sensitivity. In a common case described next, the boundaries also have a simple form which is easy to estimate.



The influence of a second parameter p_2

Figure 3. Sensitivity of $N^*(p1)$ to other parameters

2.2 Capacity boundaries for User-related Delays (Think times)

Study of a web-based telephony system showed that user think times significantly affected its capacity. If, in Figure 2(a) the varying parameter p is taken to be the think time Z, and the performance curves are taken from the well-known asymptotic bounds for delay [7], we get Figure 4(a). The straight line asymptotes are the bounds; the curves sketched in above them indicate the actual relationship.

The asymptotes are shown for three values of Z. As Z decreases the capacity decreases, and it turns out that if the upper asymptotic line is used to approximate the response delay, the relationship is linear. The upper asymptote $R = ND_{max} - Z$ applies for N sufficiently large, above the junction of the two asymptotes. D_{max} is a constant signifying the largest single execution demand per response, of any device in the system. The horizontal lower asymptote $R = D$ is the no-waiting total demand of the response.

The assumption we will make is that the system is operating above the junction of the asymptotes, which we will denote by

$$N > N' = \frac{(Z + D)}{D_{max}}$$

The assumption is reasonable if the system is intended to be quite busy at capacity, and not grossly underloaded. For a constant value $R^* > D$, the capacity value N^* given by the upper asymptote is

$$N^* = (R^* + Z)/D_{max}$$

and the value of Z at the capacity N^* is given by

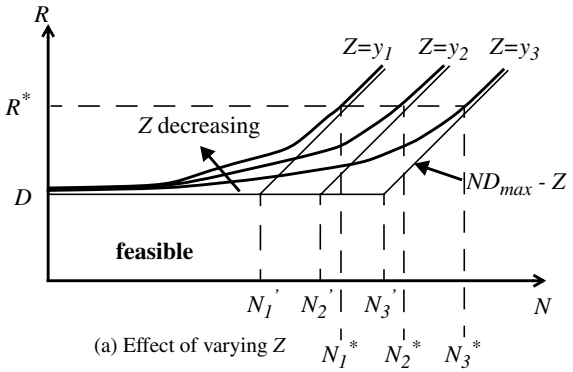
$$Z = N^* D_{max} - R^*$$

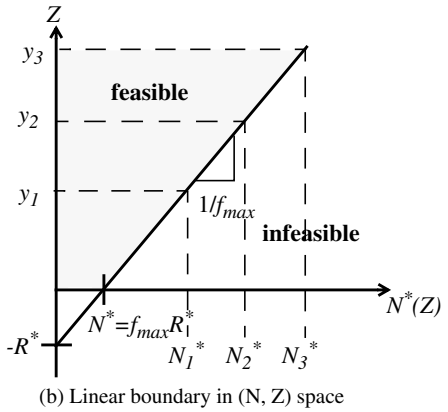
This is our capacity boundary. The constant D_{max} is (in the asymptotic analysis) also given by the inverse of the saturation throughput f_{max}, so we can write this alternatively as:

$$Z = \frac{N}{f_{max}} - R^* \quad (1)$$

This relationship represents a linear bound of feasibility as shown in Figure 4(b) for a given R^*.

The linear relationship also simplifies parameter exploration because only one point along the boundary has to be obtained, compared to obtaining results for the all the permutations of the Z and N parameter space to determine the boundary. In reality for the class of systems we are dealing with (closed queueing networks), as N increases and Z is varied to achieve a constant R^*, the system throughput actually decreases a little, which we shall ignore if N is large.





(b) Linear boundary in (N, Z) space

Figure 4. Feasible boundary as Z varies.

The linear relationship follows when the delay requirement R^* corresponds to a nearly constant system throughput, as in nearly saturated systems. Also, for large populations (but below the asymptotic region) Pittel [10] showed a constant limit in throughput. The delay solution of a closed product-form queueing system with independent think times and large N approaches the open queueing model solution:

$$R = \sum_i \frac{D_i}{1 - f D_i}$$

where D_i is the total demand (in sec.) on device i per response and f is the throughput of the system. A conservative bound can be placed on the decreasing throughput as N increases by the limiting solution $f = f^{\infty}$ given $R = R^*$. This justifies a constant in the denominator of our relationship (1), with value $f_{max} = f^{\infty}$.

2.3 Capacity boundaries for network latencies and other delays

Another notoriously unpredictable and uncontrollable parameter is the latency of network connections. First consider a network with a mean round-trip latency L seconds between the users and the application. Now the system must respond more quickly, in a mean of R' seconds:

$$R' = R^* - L$$

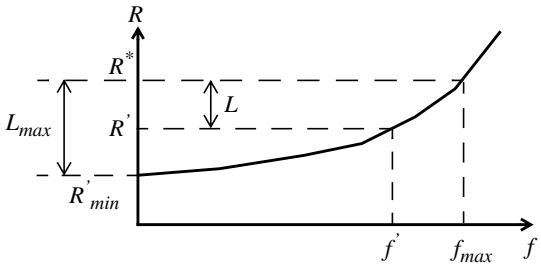
Figure 5(a) shows how R' increases with f and has a minimum possible value R'_{min} for an idle system. The largest network latency for feasible responses is:

$$L_{max} = R^* - R'_{min}$$

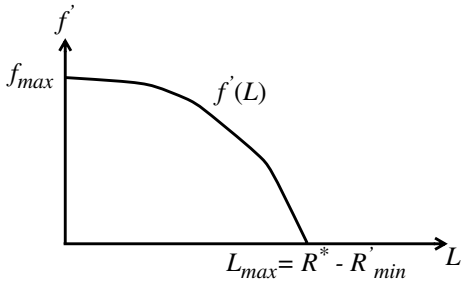
Further, the input rate f' that gives a mean user response delay R^* is a function of L, with a shape similar to Figure 5(b). Little's result can be applied with

$$N = f'(L)\,(Z + L + R') = f'(L)\,(Z + R^*),$$
$$Z = (N / f'(L)) - R \qquad (2)$$

Equation (2) defines a linear capacity boundary with a slope of *1/f* in this case also, as shown in Figure 6. Now f' decreases as L increases, and goes to zero at L_{max}. The internal latency has a pivoting effect on the boundary line around the point $(0, -R^*)$ in the (N, Z) plane.



(a) Response Time and Latency for a fixed set of parameters



(b) The Dependence of f' on L

Figure 5. The effect of network latency on the maximum feasible throughput f'

For a network delay *inside the system*, say between a server and a database that it must access, an amount L is added to R. The effect is essentially the same as described in the previous paragraph. If it is a pure latency with total mean value L during a response, and does not hold any system resources idle, it simply adds a mean amount L to each response.

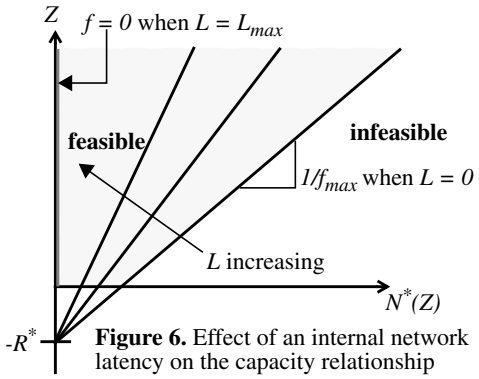


Figure 6. Effect of an internal network latency on the capacity relationship

This argument about the impact of variable internal latency does not apply to other measures, other than delay. The capacity boundaries for non-latency parameters are not straight lines.

3.0 The PhoneWeb system for Interactive Voice Response

The capacity visualization technique described above was discovered while analyzing performance results that were obtained from measurements and models of a new software system called PhoneWeb, providing web services to control IVR (Interactive Voice Response system) sessions over the telephone. In particular the straight-line boundaries helped us to interpret the results of a great number of model solutions which we used to extrapolate the system behaviour beyond the measured situations.

The system stores the speech segments and scripts which control the behaviour of a session on a web server located on the premises of the customer. This simplifies management of the system by the customer.

A sample PhoneWeb system is shown in Figure 7. Users call a phone number which is connected to the IVR subsystem, and are prompted by voice to make selections ("Press 1 for Sales, 2 for Service"). Customers are companies that provide a phone number connected to the IVR which will be called by users. A customer's web server provides VoiceXML files to govern session behaviour and sound files to be played

back to the users (for instance, files in WAV format for direct sound encoding, or text files for a text-to-speech system). These files are provided on demand.

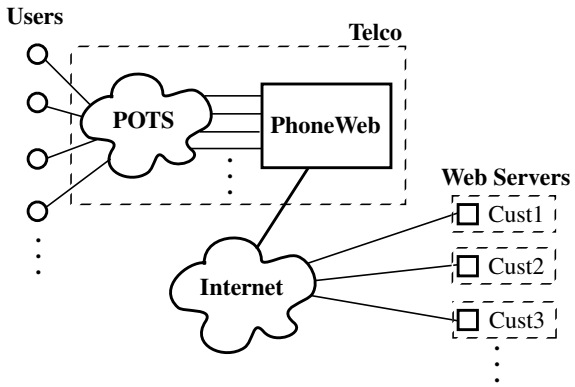


Figure 7. High level picture of the PhoneWeb IVR system

3.1 Software architecture

The software architecture of PhoneWeb shown in Figure 8 has three processes:

Call Manager sets up and tears down calls, and does the initial processing for each DTMF event (when a touch tone button is pressed).

PMLI (the PhoneWeb Markup Language Interpreter) controls the conversation for each user. It fetches the script, interprets it to control the sequence of decisions, and maintains state for each caller. Its operations include fetching data (like VoiceXML scripts, or WAV files), and initiating feedback to the user via WAV file playout. The PMLI uses virtual threads in which the context for each user is maintained in tables.

Iproxy is a web proxy server that caches data on a local disk after obtaining it from remote sites. Its purpose is to reduce the system response time by avoiding the latency of Internet accesses. Every request from the PMLI goes through the Iproxy cache and is served locally if possible. Only after a proxy cache miss does the Iproxy process need to fetch data from the internet.

Also, the *WAV playout* entity is hardware on the system bus that can be commanded to play WAV files to a user, controlled by DMA access to main memory.

3.2 Model of the PhoneWeb System

A Layered Queueing Network (LQN) performance model [2], [11] was constructed for the PhoneWeb system, based on two system request types which together create 98% of the workload:

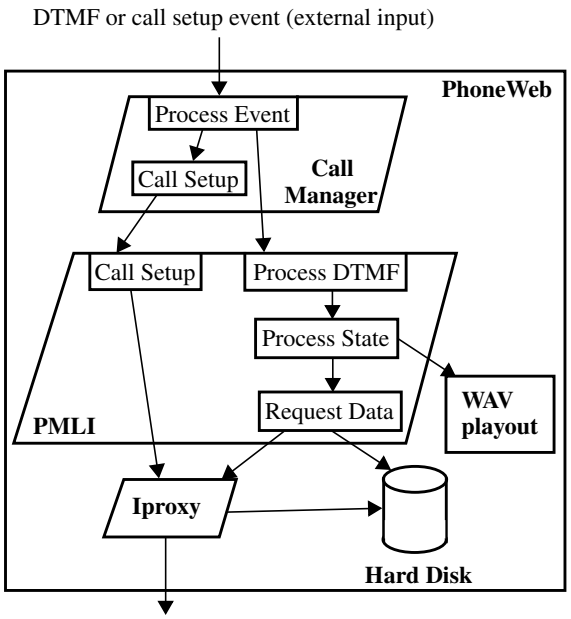


Figure 8. Software architecture of PhoneWeb, showing important software modules and call structure.

- a call setup request from a telephone switch, when a user connects to the number and triggers a Call Setup operation in the Call Manager,
- a DTMF event from a user button-press, which triggers a Process Event operation.

The response to either request is a voice playout. The response time is defined until the *start* of the playout, and the user's think time runs until the next DTMF event.

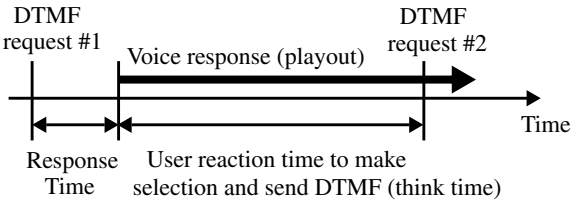


Figure 9. Assumed user behavior

The user sees a simple repetition of listening to a message, making a request by sending a DTMF signal, and waiting for the next message, shown in Figure 9. The user may send a DTMF signal before or after the message ends.

The LQN model is shown in Figure 10. Each software process in the real system corresponds to a "task" in the layered model, with a set of "entries" which provide the operations. Entries make calls to other entries, shown as synchronous (the calling thread blocks to

wait for a reply, indicated by a solid arrowhead) or asynchronous (there is no reply; an open arrowhead). Notice how Process Event in Call Manager uses three synchronous calls to the entries in PMLI to execute the three activities shown as a sequence in Figure 8. The analysis models the queues at tasks and at processors.

Because the tasks use a virtual thread for each caller, they are modeled as having infinite threads. The configuration has a single CPU for all three tasks. Each user in the system is represented by a "DTMF Events" source task (with its own "processor") which generates DTMF requests for the system to process.

As soon as the playout begins, the think time of the DTMF Events task begins for the next cycle. This is modeled using a layered queueing feature called a "second phase" of service [3], in which the reply from "Process Event" is sent to DTMF Events before the call to "Play WAV File". The effects of DMA access from both the hard disk and WAV playout devices is also accounted for in the model by a higher priority task DMA_Activity which occupies the bus.

The Hard Disk "task" models I/O operations, with a processor to model the disk device. Net Delay is a delay center to model the network latency of the internet access, and assumes that network bandwidth is not a bottleneck. A remote Web server called "Page Server" represents any customer's web server.

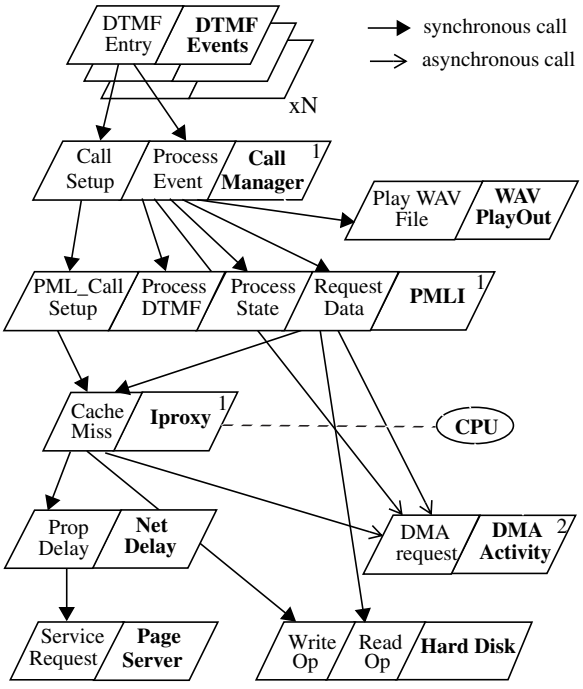


Figure 10. Detailed model of the Phone Web system

3.3 Obtaining model parameters from the system

Each software task has entries to describe the operations. CPU demands were obtained from log traces from a test bed with many callers entering and leaving the system, and making DTMF requests. The Intel hardware clock was accessed to obtain approximately 1 microsecond granularity on times. Since this was a wall-clock time, the values for each function were filtered to remove outlier intervals that contained processing due to pre-emptions. There were virtually no such outliers. Because the logs did not record the use of processes other than the three in Figure 8, such as internet daemons, or miscellaneous kernel processing, an allowance of 30% was made for limitations on workload capture.

3.4 Assumed parameter values

Some parameters values were assumed without measurement, including:

- The remote web server was modeled with 50 threads and an average access time of 61 ms to fetch a document of average size 62.5 Kbytes [1]. This time was scaled to the average file size used.
- DMA activity was accounted for in the model, based on the I/O bus speed. It assumed that the entire WAV file is played out every time, and thus may overestimate the demand.
- the average size of a web file was taken as $F = 80$ Kbytes, corresponding to a WAV file playout time of about 10 seconds.
- Operations on the local disk (for PLMI to check the cache) were estimated at 15 ms for an 80 Kbyte file, and file caching by the kernel was ignored.

Some other parameters were assumed to take a range of values, as discussed next.

4.0 Determining PhoneWeb capacity

PhoneWeb is complex, and does not fit into the category of simple product form queueing networks. Even though the model is simpler than the system, it is difficult to understand intuitively. This section presents the application of the ideas presented earlier to a real system.

The capacity of the PhoneWeb system was explored with more than 3,800 model simulation runs of many parameter variations, including the think time and the user population. The simulations were stopped

when waiting times, processor and task utilizations had a 95% confidence interval of less than +/- 0.5% of the stated result. The linear approximation for the feasible boundary was developed to abstract the results.

The user response time was sensitive to many model parameters, and was the result of primary interest. Parameters with substantial uncertainty were varied, including:

- proxy cache hit ratios H, from 0.5 to 1.0 (we considered steps of 0.1).
- an average round-trip network delay L, incurred after a Proxy cache miss. A value of 100ms was used to represent a transfer across a LAN (80 Kbytes at 800 Kbytes/sec). A value of 1 second was used to model a transfer across the Internet at 80Kbytes/sec.

Also, the user population and user think time were varied:

- the average user think times Z ranged from 2 to 14 seconds in steps of 4 seconds,
- the user population N ranged from 10 to 290 active users on one PhoneWeb system in steps of 40 users. As soon as one user session ends, the model begins another for that user.

The stated requirement was that 95% of waiting times should be less than 1 second, however results confirmed that waiting times were close to exponentially distributed. The 95% value is thus ensured by a target mean response time R^* of 0.33 sec.

Data was gathered for all combinations of H, L, N and Z described above, contour plots in the *(N, Z)* plane were created for each pair (H, L), and straight lines were fitted by eye. Figure 11 shows results for a range of values of H, and L = 0.1; Figure 12 has results for L = 1.0. The term "waiting time" in these figures is the quantity R that has been defined here.

The contours for different mean waiting-time values (meaning our R) are in line with the linear approximation described in Section 2.2. The plots are roughly linear and take on the form of:

$$N = f_{max}(R^* + Z)$$

where

- R^* is the negative Z-intercept of the line, in this case 0.33 seconds
- f_{max} is the inverse slope of the line

and each contour can be summarized by its values for (R^*, f_{max}).

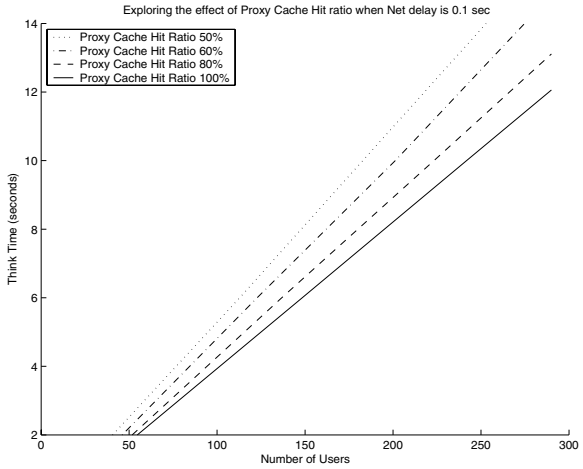


Figure 11. Summary of capacity results for a Net Delay of 0.1 sec

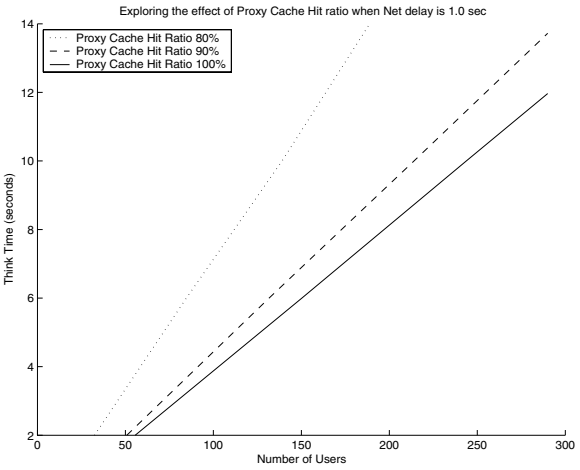


Figure 12. Summary of capacity results for a Net Delay of 1 sec

4.1 Parameter set for sensitivity analysis

An initial set of parameters were considered:

- Proxy Cache hit ratios of H = 0 to 1.
- WAV file size from F = 80 KB to F = 120 KB.
- the length d of a user session in DTMF events varied from 5 to 20 in steps of 5 (nominal 20). This affects the ratio of connection processing operations to DTMF handling.
- Hit ratio for PMLI peeking in the cache, with values h = 0 to 1.

As in Section 4.0, R^* = 0.33 seconds was used for the mean waiting time. The results to these experiments can be found in Tables 1, 2, 3, and 4.

These tables show that hard drive caching, and increasing WAV file sizes from 80 Kbytes to 120 Kbytes have a negligible effect on capacity due to the insensitivity of f_{max}. Shorter phone calls have a larger effect in decreasing capacity, but have less impact than varying the proxy cache hit ratio for higher network latencies. The other interesting observation is that the system cannot achieve $R < R^* = 0.33$ seconds with proxy cache hit rates of less then 0.8 when the Net Delay is 1.0 second.

TABLE 1. Results exploring Proxy cache hit rate *H*

	***L* = 0.1 sec**		***L* = 1.0 sec**	
H	f_{max}	$-R^*$	f_{max}	$-R^*$
0	7.82	-0.1030	infeasible	
0.1	9.51	-0.1397	--	
0.3	13.45	-0.3850	--	
0.5	17.78	-0.2840	--	
0.6	19.41	-0.3727	--	
0.8	21.44	-0.4144	13.32	-0.434
1.0	23.40	-0.3347	23.53	-0.362

TABLE 2. Results exploring effect of shorter phone calls.

	***H* = 0.5**		***H* = 1.0**	
d, L	f_{max}	$-R^*$	f_{max}	$-R^*$
5, 0.1	16.18	-0.324	22.18	-0.416
10, 0.1	17.06	-0.316	22.83	-0.383
15, 0.1	17.84	-0.248	23.21	-0.356
20, 0.1	17.78	-0.284	23.40	-0.335
5, 1.0	infeasible		22.06	-0.583
10, 1.0	--		22.77	-0.449
15, 1.0	--		23.40	-0.370
20, 1.0	--		23.53	-0.362

TABLE 3. Results exploring effect of increasing WAV files from 80Kbytes to 120 Kbytes.

***W* (KB), *L* (sec)**	***H* = 0.5**		***H* = 1.0**	
	f_{max}	$-R^*$	f_{max}	$-R^*$
80, 0.1	17.82	-0.297	23.40	-0.338
120, 0.1	15.82	-0.329	22.54	-0.379
80, 1.0	13.75	-0.293	23.49	-0.365
80, 1.0	11.54	-0.141	22.84	-0.382

TABLE 4. Results exploring effect of hard drive caching, using a net delay of 0.1 sec.

***h*, *W* (KB)**	***H* = 0.5**		***H* = 1.0**	
	f_{max}	$-R^*$	f_{max}	$-R^*$
1, 80	18.64	-0.271	23.46	-0.343
0.7, 80	18.26	-0.332	23.70	-0.319
0.4, 80	17.91	-0.363	23.46	-0.348
0, 80	17.88	-0.306	23.69	-0.360
1, 120	18.03	-0.300	23.1758	-0.356
0.7, 120	17.75	-0.313	23.3419	-0.347
0.4, 120	17.13	-0.316	22.9727	-0.364
0, 120	16.36	-0.241	22.4768	-0.387

The Z-intercept of $-R^*$ is quite close to -0.33 in every case, validating the approximation of Section 2. The two most sensitive parameters are the Think Time *Z* and the Proxy cache hit ratio *H*.

4.2 More parameters considered for sensitivity analysis

Supplemental analysis was done on a larger range of parameters described in Section 4.1:

- proxy cache hit ratios of $H = 0, 0.3, 0.5, 1.0$
- WAV file sizes of $F = 80$ K, 120 K, 512 K, 1024 K bytes
- longer net delays of $L = 1, 5, 10$ seconds.

It is impractical to demand a 0.33 second mean delay in most of these cases, so the users were assumed to be more patient, and to take the network delay into account. An allowance for network latency was added to the target delay, by adding an "effective network latency" (ENL) value to the required delay. This made:

$$R^* = \text{ENL} = L\,(1 - H) + \textit{target delay},$$

with the target delay chosen to be 1.0 second.

Tables 5, 6, and 7 are the same experiments as shown in Section 4.1, except using $R^* = \text{ENL}$. Figures 13, 14, and 15 also examine more extreme variations of Net Delay and WAV file sizes.

For the results in Tables 5, 6, and 7, capacity increased slightly by the shift of the value of $-R^*$ from -0.33 to -1.0 to -1.5 depending on the value of ENL. The same parameters that are sensitive to capacity in Section 4.1 are still sensitive (think time, proxy cache hit, and shorter phone calls), and the same goes for the insensitive parameters (hard disk cache, file sizes from 80 Kbytes to 120 Kbytes). When $R^* = \text{ENL}$, proxy

cache hit rates greater than 0.5 when Net Delay are 1.0 seconds is now feasible.

TABLE 5. Results exploring effect of shorter phone calls using ENL

***d*, *L* (sec)**	***H* = 0.5**		***H* = 1.0**	
	f_{max}	$-R^*$	f_{max}	$-R^*$
5, 0.1	21.52	-1.337	24.76	-1.168
10, 0.1	23.09	-1.157	25.94	-1.042
15, 0.1	23.43	-1.136	25.93	-1.063
20, 0.1	23.58	-1.121	26.87	-0.939
5, 1.0	21.66	-1.963	24.86	-1.332
10, 1.0	23.06	-1.693	26.20	-1.098
15, 1.0	23.53	-1.605	26.35	-1.067
20, 1.0	23.38	-1.537	26.55	-1.024

TABLE 6. Results exploring effect of hard drive caching, using a net delay of 0.1 sec. using ENL

***h*, *W* (KB)**	***H* = 0.5**		***H* = 1.0**	
	f_{max}	$-R^*$	f_{max}	$-R^*$
1, 80	23.87	-1.082	26.47	-0.982
0.7, 80	23.7	-1.105	26.65	-0.967
0.4, 80	23.77	-1.083	26.50	-0.987
0, 80	23.60	-1.105	27.05	-0.946
1, 120	23.54	-1.109	26.43	-0.981
0.7, 120	23.53	-1.095	26.32	-0.996
0.4, 120	23.31	-1.134	26.70	-0.952
0, 120	23.00	-1.140	26.66	-0.955

TABLE 7. Results exploring effect of increasing WAV files from 80Kbytes to 120 Kbytes using ENL

***W* (KB), *L* (sec)**	***H* = 0.5**		***H* = 1.0**	
	f_{max}	$-R^*$	f_{max}	$-R^*$
80, 0.1	23.88	-1.073	26.45	-1.001
120, 0.1	23.16	-1.1078	26.36	-0.989
80, 1.0	23.79	-1.566	26.46	-1.036
80, 1.0	23.63	-1.515	26.34	-1.045

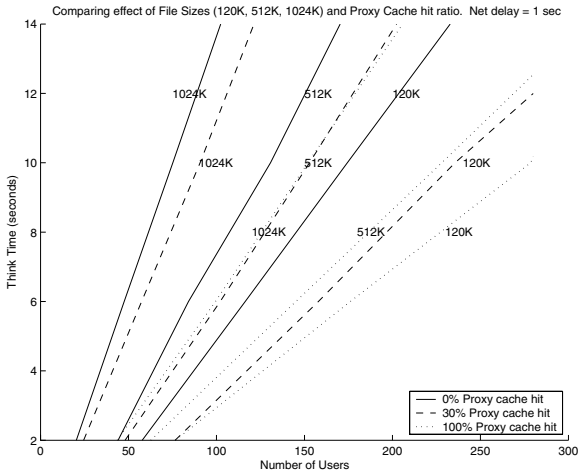


Figure 13. Exploring file sizes of 120K, 512K, 1024K bytes, for network delay of 1 sec. using ENL

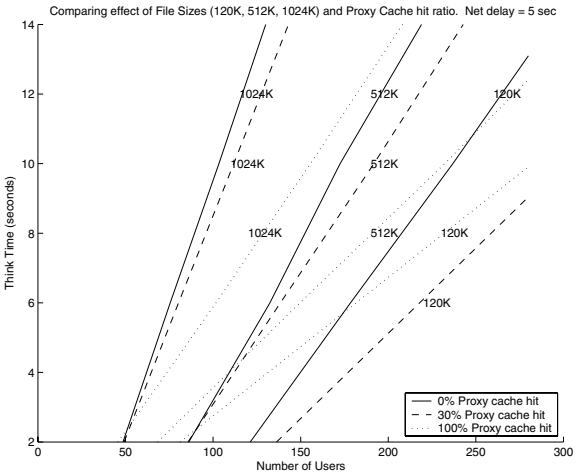


Figure 14. Exploring file sizes of 120K, 512K, 1024K bytes, for network delay of 5 sec. using ENL

In the exploration of larger file sizes and network latencies (Figures 13, 14, and 15), the curves show more sensitivity. One interesting observation is for lower think times, and higher network latencies, the capacity of the system is greater for lower proxy hit rates. This phenomena is more pronounced as the WAV file size decreases. The explanation for this is, user cycle times increase proportionally as proxy cache hit rates decrease. Therefore more users can exist in the system, "trapped" in a high network latency and not incur work on the system itself. Which means more users can enter the system, and since R^* is adjusted by the ENL formula, the PhoneWeb itself is not penalized for the long network delays. As think times and populations increase, there is a cross over point when it is

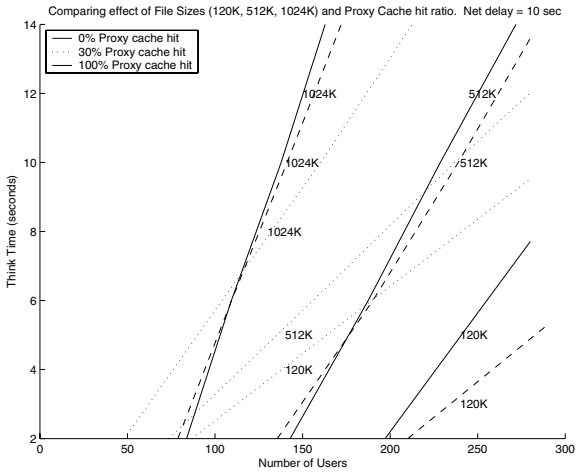


Figure 15. Exploring file sizes of 120K, 512K, 1024K bytes, for network delay of 10 sec. using ENL

beneficial for the capacity of the system to have higher proxy cache hit rates. These cross over points can be seen in Figures 14, and 15 for a given file size.

5.0 Conclusions

This paper has described a visualization of system capacity as it is affected by various kinds of parameters, and a simple linear capacity boundary that applies to some parameters. The linear boundary is easy to calibrate for varying think times and latencies. This can lead to a very substantial savings in effort to explore capacity limits. For instance in the PhoneWeb system studied here, 32 simulations were required to derive one contour plot for one varying parameter and N, whereas only two simulations are needed to establish the constant parameters of the straight line boundary. Also the compact linear representation makes it easier to understand how changes will affect capacity, and to communicate the results to others.

Overall this visualization can speed up the exploration of capacity, even though other parameters do not provide linear boundaries. It showed which parameters could be safely ignored in trying to optimize the design before deployment.

A successor to this system has been deployed since the study was carried out.

6.0 References

[1] Jussara M. Almeida, Virgilio Almeida, and David J. Yates, “WebMonitor: a tool for Measuring World-Wide Web Server Performance”, *First Monday*, Vol. 2 No. 7 - July 7th. 1997.

[2] R.G. Franks, S. Majumdar, J.E. Neilson, D.C. Petriu, J.A. Rolia and C.M. Woodside, “Performance Analysis of Distributed Server Systems”, *Proc. Sixth International Conference on Software Quality*, Ottawa, Canada, October 28-30, 1996, pp. 15-26.

[3] R. G. Franks, C. M. Woodside, “Effectiveness of Early Replies in Client-Server Systems”, *Performance Evaluation*, v 36-37, 1999.

[4] E. Borowsky, R. Golding, P. Jacobson, A. Merchant, L. Schreier, M. Spasojevic, and J. Wilkes, “Capacity planning with phased workloads”, *Proc. First Int. Workshop on Software and Performance (WOSP98)*, Santa Fe, Oct. 98, pp 199 - 207.

[5] Seungrahn Hahn, M.H. Ann Jackson, Bruce Kabath, Ashraf Kamel, Caryn Meyers, Ana Rivera Matias, Merrilee Osterhoudt, Gary Robinson, “Capacity Planning for Business Intelligence Applications: Approaches and Methodologies”, *IBM Redbooks*, 2000.

[6] Raj Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[7] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik, *Quantitative Performance Modeling*, Prentice-Hall, 1984.

[8] Daniel A. Menasce, Virgilio A. F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, Prentice-Hall, 1998.

[9] Daniel Menasce, Virgilio Almeida, Lawrence Dowdy, *Performance by Design: Computer Capacity Planning by Example*, Prentice-Hall, 2002.

[10] B. Pittel, “Closed exponential networks of queues with saturation: The Jackson-type stationary distribution and its asymptotic analysis”, *Mathematics of Operations Research,* Vol. 4, No. 4, November 1979, pp. 357 – 378.

[11] C. M. Woodside, J.E. Neilson, D. C. Petriu, and S. Majumdar. “The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software”, *IEEE Trans. Computers*, Vol 44, No. 1, January 1995, pp. 20 - 34.