Measuring End-User Availability on the Web: Practical Experience

Matthew Merzbacher
University of California, Berkeley
merz@eecs.berkeley.edu

Dan Patterson
University of California, Berkeley
patter@uclink.berkeley.edu

Abstract

For service applications on a network, measuring availability, performance, and quality of service is critical. Yet traditional software and hardware measures are both inadequate and misleading. Better measures of availability that incorporate end-user experience will lead to meaningful benchmarks and progress in providing high-availability services. In this paper, we present the results of a series of long-term experiments that measured availability of select web sites and services with the goal of duplicating the end-user experience. Using our measurements, we propose a new metric for availability that goes beyond the traditional sole measure of uptime.

Keywords: availability, retry, metrics, dependability

1. Introduction

The World Wide Web has revolutionized commerce. Goods, services and information are meant to be available nonstop and the vitality of online businesses depends on ensuring this high availability. But how do we measure availability? The traditional measure is uptime, calculated by taking the percentage of time servicing requests relative to total time (excluding scheduled downtime). This availability is typically reported as a number of "nines", a measure within an order of magnitude. Four nines of availability equates to 99.99% uptime (just under an hour of unscheduled downtime per year). Currently, server manufacturers advertise that their products offer six or more nines (less than thirty seconds of unscheduled downtime per year).

Experience shows these numbers to be misleading – no web sites are anywhere near as robust as their hardware and server software alone. The advertised numbers reflect performance under optimal operating conditions, rather than real-world assumptions. Any meaningful availability measure must capture the end-user experience, which includes several independent components: the network, multiple server software layers, and separate client software and system. More meaningful measures of availability will help direct development of future systems, forcing focus on the real sources of downtime instead of a single number and an unrealistic operating environment [2] [3].

In order to measure existing systems, undergraduate students in a joint class between Mills College and UC Berkeley devised and ran an experiment over several months to on several prominent and not-so-prominent web sites. Our experiment made hourly contact with a list of sites and, in some cases, performed a small transaction. We measured and analyzed the responses in terms of success, speed and size. Our analysis shows that local and network conditions are far more likely to impede service than server failure.

1.1. Related work

There are several systems and products available that attempt to monitor and measure the end-user experience, such as Topaz [5], SiteAngel [1], and Porvio [7]. These products identify weak points in the end-to-end experience by emulating transactions, but are geared toward measuring performance in terms of transaction speed (and causes of slowdowns) rather than availability. In general, they are trying to monitor and measure end-user service-level agreements. Other services, such as Netcraft [6], monitor and report server availability and reliability, but do not measure end-user experience.

In contrast, our objectives were to measure the end-user experience in terms of availability (including, but not limited to response time) and locate common sources of trouble. Based on our results, we also evaluated the efficacy of retry, a step absent in the other techniques.

2. Experiment

Our experiment, coded in Java, accessed a variety of different sites, ranging from local machines to servers around the world. By including local URLs, we could determine a baseline response time and network delay and evaluate local problems. The experiment ran hourly for six months on machines on two campuses, UC Berkeley and Mills College. Both colleges are located in the San Francisco Bay Area (California, USA) and share some network connections to the outside world.

To avoid problems with regular access, the experiment was delayed by a random number of minutes at the start of each hour, so it might run at 3:11 and then 4:35.

We present results from three prominent sites that characterize the behavior that we observed:

a) An online retailer, with international country-specific versions of its site
b) A search engine, where we executed several searches
c) A directory service, again with country-specific versions of its site

We also measured results from sites providing news, auctions, and other services, but we do not include the specifics for those sites, as they are qualitatively similar.

3. Results

Quantifying availability with a single number is impossible. In addition to the basic question of whether or not we received a response, we must also consider whether the response was partial or complete and how long it took to arrive. Further, our model must also allow for retry when failure does occur.

We start by presenting the raw "success or failure" numbers for each service. We factor out different kinds of errors from our data to try to determine where errors occur. Our results are summarized in Table 1.

Table 1: Availability under various assumptions

	All	Retail	Search	Directory
Raw	.9305	.9311	.9355	.9267
Ignoring local problems	.9888	.9887	.9935	.9857
Ignoring local and network problems	.9991	.9976	1.00	.9997
Ignoring local, network and transient problems	.9994	.9984	1.00	.9999

We describe the detailed meaning of each row in subsections 3.1 – 3.4. Briefly, the first row is the overall frequency that each site returned its data both perfectly and quickly, the second row excludes local problems on the end-user machine, the third row further excludes problems with the network, and the fourth row also excludes any non-repeatable problem. What remains are persistent problems that are likely happening at the company's website or throughout the corporation. So, for example, the search engine failed about 6.5% of the time, but that was mostly due to local problems on our experimental platform. After local and network problems were eliminated, the search site never failed.

The pie chart in Figure 1 shows the frequency of different kinds of errors, each described in detail below.

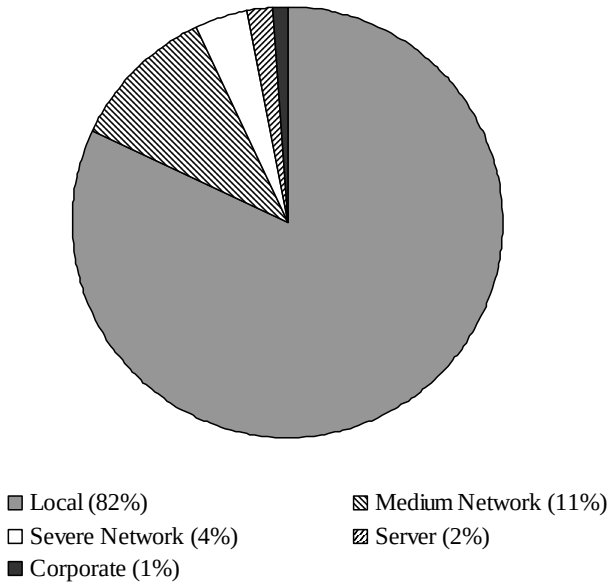

Figure 1: Types of errors

3.1. Local problems

By far the most common causes of unavailability were local problems on our test machines, including system-wide crashes, system administrator configuration errors, client power outages, attacks by outside hackers, and a host of other troubles. No particular problem was especially more common than another – there were no few characteristic local errors where one could focus preventative energies. However, almost all had a component of human error that either caused or exacerbated the problem.

Sometimes, for example with power outages, the trouble would be resolved by the next hour, while other troubles required considerably more intervention. Part of this was because we ran our experiments on multi-user server machines, but individual workstations should fare no better. Our machines, while in an academic setting, were relatively carefully administered by professional systems administrators and were more stable than typical campus machines. Other students and faculty also used them only sparingly – almost an ideal set-up.

In summary, our key observation:

Local availability dominates the end-user experience

Factoring out local outages, as done in the second row of Table 1, provides a better estimate of availability before the so-called "last mile" without considering the end-user.

3.2 Losing data

A second kind of unavailability occurred when some, but not all, of the requested data was retrieved. We measured the size (in bytes) of data returned. This value fluctuated slightly from hour to hour as sites updated or as advertisements changed. However, when size dipped below a fixed threshold, it indicated a problem. A size of zero clearly shows unavailability, but even if the site returned with a radically small size, we believe that this indicated incomplete data. More likely, anything less than 4KB returned indicated some error, such as 404 (page not found). We compared data between our two experimental platforms, and in cases where neither location returned close to the full size, ascribed it to server error.

3.3 Corporate failure

Where they existed, we also accessed international versions of the server sites. Naturally, such accesses took longer, although they frequently had less variance than the US versions. Some failures spanned all sites for a particular enterprise – both US and international. We call such failures "corporate" failures and indicate them on the failure chart.

3.4 Timely results

Another type of unavailability is when a site returned successfully, but slowly. The important consideration of what is "too slow" is hard to define in a strict sense, so we went about this a couple of ways. The first was to chart

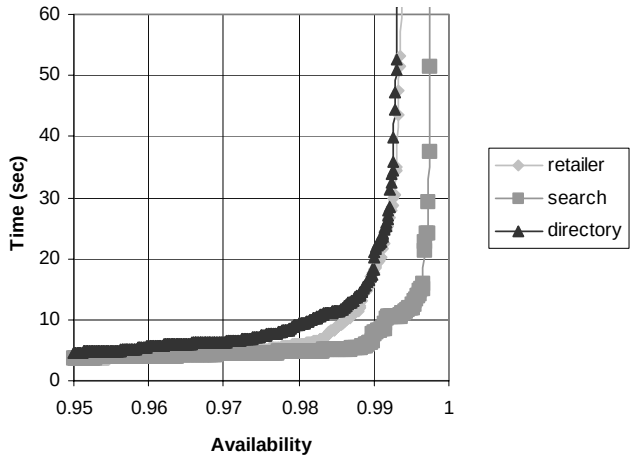

Figure 2: Time grows quickly at high availability

"availability versus time", shown in Figure 2. This graph plots expected availability against the length of the wait. From it, a user can determine how long to wait for a certain degree of availability (e.g. 99.9%) or the availability found when waiting for a fixed time (e.g., 10 seconds). As expected, the graph is fairly flat until it gets above 99%, at which point the time needed to wait approaches grows exponentially.

As with size, we chose a threshold for time, marking sites above the threshold as unavailable. This kind of error can indicate network delays, although it may also indicate an overloaded server. Ignoring such slow-but-successful errors yielded the results in the third row of Table 1. In the pie chart, we divide timing problems into "medium" (ten seconds) and "severe" (thirty seconds). Ten seconds is well above the normal return time for all the sites studied. Above thirty seconds, even large changes in the threshold don't have much impact on the final value.

After client errors, one-time network errors (medium and severe) were the most frequent kind of error.

3.5 The effect of retry

When presented with an unavailable web site, users often retry at least once. Whenever sites were unavailable for one of the aforementioned reasons, we checked the failure for persistence. That is, for each error, the data was inspected to see if the error repeated when the site was accessed again in the next hour. This allowed us to categorize the errors, differentiating one-time failures from problems that persisted for many hours. Such analysis is particularly useful for sites with a long delay, because it is difficult to pinpoint the source of the delay in the chain between user and server. When a single site is consistently slow relative to other sites, it suggests trouble at (or near) the server[1]. In any case, by measuring the availability of the site after retry, we were able to more accurately measure the total user experience. The final row of Table 1 includes only sites that experienced persistent inexplicable downtime.

Table 2: The value of retry

Error	All	Retailer	Search	Directory
Client	0.2667	0.2706	0.2647	0.2648
Medium Network	0.8621	0.8704	0.9286	0.8318
Severe Network	0.7895	0.9231	1.00	0.6889
Server	0.9111	0.7857	1.00	0.9600
Corporate	0.4210	0.3125	1.00	n/a

[1] From the end-user perspective it is impossible to distinguish a server failure from a network failure near the server. Further, in cases where there is no response, it is impossible to distinguish server software failure from hardware failure.

Table 2 shows how well retry overcomes each class of failure. At the client, most errors were persistent, and so retry had a limited effect, while retry was much more effective for most other kinds of problems. Corporate errors were different; the vast majority of corporate errors occurred at the retailer site, and when they did they were largely persistent.

3.6 Persistent non-local failures

Once local and one-time errors are removed, only a few errors remain. The distribution of these more severe errors (shown in Table 3) is domain-dependent. For instance, the directory site was much more prone to persistent network problems, suggesting a problem with the connection to their international sites. By contrast, the retailer site was more prone to corporate errors, due to the added complexity of having to deal with storing user information. This site also suffered persistent problems with its servers, and a handful of times when all editions of the site worldwide had to be taken offline – a corporate disaster.

Table 3: Persistent errors by type

Error	Retailer	Search	Directory
Medium persistent network	7	3	**18**
Severe persistent network	2	0	**14**
Server persistent errors	**3**	0	1
Corporate persistent errors	**11**	0	0

4. Analysis

Our retry period of an hour is unrealistically long. In practice, the likelihood of retry, number of retries, and the time before the user stops retrying are based on several domain- and user-dependent factors. Users may retry immediately to try to overcome a short-term problem, but, if presented with persistent unavailability, may continue to retry or return after some time. Economists know that shoppers in conventional stores, when thwarted in making a purchase, often will find another source for the desired item or skip the purchase entirely [4]. There is no reason to expect that e-commerce sites, especially where competition is easily found, would be any different. However, there are certain sites where users, even after an extended outage, will retry. The likelihood of retry is determined by:

a) Uniqueness – if a site offers unique service, goods, or information, then there is less competition. This increases the chance of retry, even after a long wait.
b) Import – the value and cost of the transaction to the user is a key factor in retry. The more costly and important the transaction, the more likely to have retry.
c) Loyalty – users exhibit faithfulness when browsing the web. The higher the degree of loyalty, the longer the retry period.
d) Transience – the length of delay before retry is inversely proportional to the speed at which the content of the site changes. That is, sites where the information is highly transient are unlikely to have long-term retries, while sites that don't change (and whose content may well be unique) are more likely to have a long retry period.

5. Conclusion

This experiment was designed to model the user experience, and was successful. We found about 93% raw availability, but the great majority (81%) of errors that occurred were due to errors on the local side that prevented the experiment from being run. Removing local and short-term network errors, availability increased to 99.9%. Lastly, we saw the effect retry had on the availability, and found that, while local errors were reduced by only 27%, non-local errors fell by 83%. It's clear that local availability has the largest impact on the user experience. Also, when local problems are factored out, retry becomes a strategy that cuts error substantially. When removing local and network problems and retrying, we generally get at least three nines of availability, which, while lower than what is found in an ideal environment, is still respectable.

There are several areas in which this work can be expanded. The first is to continue the experiment and refine our availability numbers in the final row. In several cases, the number of errors is just above the granularity of the experiment. Also, the experiment should be distributed to distant sites so that we can better assess the source of errors. Lastly, we need better experiments to measure the efficacy of retry, both in the short-term and in the longer-term.

With better measures of availability, we can devise test suites and benchmarks that will lead to more reliable systems and the pinpointing of sources of failure.

6. Acknowledgements

We thank the students from Mills College and UC Berkeley who participated in a joint class and helped design and implement this experiment. We also thank the members of the ROC research group, who provided valuable feedback on our work. Thanks to the referees for their valuable commentary on this work. Lastly, we thank the National Science Foundation for generous support of this research.

7. References

[1] "Building an Internet-centric Monitoring Infrastructure", ASP News Review, September 2001. **http://www.aspnews.com/strategies/technologies/article/0,2350,10584_924571,00.html**

[2] A. Brown and D. A. Patterson, "To Err is Human", *Proceedings of the First Workshop on Evaluating and Architecting System dependabilitY (EASY '01)*, Göteborg, Sweden, July 2001.

[3] A. Brown and D. A. Patterson, "Embracing Failure: A Case for Recovery-Oriented Computing (ROC)", *2001 High Performance Transaction Processing Symposium*, Asilomar, CA, October 2001.

[4] R. H. Frank and B. S. Bernanke, *Principles of Economics*, McGraw-Hill/Irwin, New York, 2001, pp. 92-3.

[5] "Topaz: The Complete Solution for Application Management", Mercury-Interactive, December 2001. **http://www-svca.mercuryinteractive.com/products/topaz/index.html**

[6] "Netcraft: What's that Site Running?", Netcraft, December 2001. http://www.netcraft.com

[7] "Quality of Experience: measuring true end-to-end Web performance", Porvio, December 2001. http://www.porivo.com/peerReview/Porivo_paper.pdf