When Does Fast Recovery Trump High Reliability?

Armando Fox, Stanford University
David Patterson, University of California, Berkeley
Stanford/Berkeley Recovery-Oriented Computing Project (ROC)

A widely accepted equation for availability is Availability=MTTF/(MTTF+MTTR), where MTTF is the mean time to failure of a system or subsystem (i.e. the reciprocal of reliability), MTTR is its mean time to recovery, and A is a number between 0 and 1. The equation suggests that to improve availability, a 10x decrease in MTTR (for example) is just as valuable as a 10x increase in MTTF, a founding observation of the Recovery-Oriented Computing (ROC) project [9]. In this paper, we argue that for interactive Internet applications, a decrease in MTTR is sometimes *more valuable* than the corresponding increase in MTTF to improve Availability by the same amount, and we make a case for adopting MTTR as the primary metric for reasoning about system availability and focusing designs on fast recovery.

The availability equation characterizes the "percent availability" of the system, i.e. for a "five nines" system we expect $A \geq 0.99999$, meaning that the system is up (available) 99.999% of the time. A common interpretation of availability is that it represents the probability that any given request made of the system will be successfully serviced. Today, data centers and Internet sites achieve availabilities between 0.99 and 0.999, meaning that MTTF is 100x to 1000x MTTR. The following observations summarize the claims supporting the argument that lowering MTTR is sometimes more valuable than increasing MTTF:

1. In the case of hardware, today's component MTTF's are so high that directly measuring them requires many system-years of operation; most customers cannot afford this and must largely rely on vendor claims to assess the impact of MTTF on availability. On the other hand, MTTR *can* be directly measured (for both hardware and software), making MTTR claims independently verifiable.

2. For end-user interactive services such as Web sites, lowering MTTR can directly affect the user experience of an outage. In particular, reducing MTTR can be shown to reduce the impact (and therefore the cost) of a specific outage, especially when redundancy and failover are used to mask the failure. In contrast, increasing MTTF may reduce the frequency of failures (and therefore the probability that a given user will experience a failure during her session), but does not capture the impact of a particular outage on the user experience or cost to the service provider.

3. When low MTTR cannot completely mask a transient failure, it may be possible to structure the service so that the level of service gracefully degrades under partial failure (and hence under recovery). We give a simple, performability-inspired [23] argument that if the degree of degradation can be made small enough, the duration of degraded service (i.e. recovery time) short enough, and the frequency of degraded service intervals low enough, experience suggests that users are willing to tolerate such graceful degradation. This observation can be translated directly to techniques that fall back to degradation while recovering from a failure or avoid unscheduled downtime by "inducing" a failure prophylactically and recovering from it (rejuvenation [11]).

1 Direct Measurability and Relevance of MTTR

Today's hardware MTTF's are too large for all but the largest customers to measure. For example, many disks have quoted MTTF of 120 years; verifying such claims requires many system-years of operation. Verification is further complicated because hardware manufacturers exclude operator error and environmental failures from their calculations, even though they account for between 7% and 28% of all unplanned downtime in a some H/A and F/T cluster and mainframe installations [27] and more than half of unplanned downtime in a selection of contemporary Internet services [26].

In the case of software, MTTF's vary from days to months, whereas MTTR varies from minutes to hours. A 1995 study of 1200 randomly-selected Internet servers [18] observed a skewed MTTF distribution with a median of 5.5 days and a mean of about 16 days. Since these studies took steps to isolate "near-destination" failures (as opposed to, e.g., "near-middle" failures of the network infrastructure), and since server updates requiring manual operator intervention are relatively rare, these are defensible first-order estimates of Internet host software MTTF. In contrast, [17] finds typical Internet host MTTR's under 15 minutes, and even the longest MTTR's for complex commercial database products (such as those typically used for mid-size Internet service installations) are on the order of hours, making software MTTR claims verifiable. So for both hardware and software, MTTF>>MTTR, and support for recovery benchmarking is therefore starting to appear [32].

A related complication resulting from high MTTF is the duration of the observation window. For example, last year Microsoft blamed operator error for a 24-hour outage affecting most of its major Web sites. If the company wanted to achieve "five nines" availability despite that outage, those Web sites would need to operate outage-free for the next 250 years. Besides the fact such a claim would stretch credibility, it is not meaningful to begin with because most businesses cannot wait 250 years to assess customer satisfaction—they need finer-grained, ongoing metrics of such things as system availability as perceived by their customers.

We can summarize the above arguments as follows: high MTTF can never guarantee a failure-free interval (unless MTTF is infinite), so MTTF at best measures the frequency with which something will happen or the chance it will happen during a given observation window, e.g. there is a 5% chance an outage will occur this month. However, the measurable cost to the service provider is not in an increased likelihood of an outage, but in the impact of a specific outage, as we will show next. MTTF does not capture outage cost, nor does increasing MTTF directly lead to lowering outage costs, but lowering MTTR can mitigate the impact of a failure, and can be measured for a specific incident. We now support this claim by identifying some technology-independent thresholds for MTTR.

2 End-User Benefit of Lower MTTR

In April 2002, Ebay had a 4.5-hour outage affecting most of its services. If Ebay suffered only two such unplanned outages per year[1], their MTTF would be half a year and their availability using the above formula (considering only unplanned outages) would be (182*24 hours)/(182*24 + 4 hours) = 99.9%. But long outages are newsworthy [4][5][13][21][28] because of their impact

[1] In fact, Ebay also suffered a 12-hour outage on 11 April 2002 that affected certain auction categories , a 22-hour sitewide outage in June 1999 [13], a 5-hour outage on 3 May 1999 [21], a 7-hour outage on 21 May 1999 [4], and a 9-hour outage in December 1998 [5]. All except the 3 May 1999 outage were caused by software-related failures.

on user loyalty and investor confidence: after their June 1999 outage, investors dropped Ebay's stock price by 26% and its market capitalization fell US$4 billion, and the company incurred direct costs of between US$3M and US$5M in credits to affected Ebay users[2], whileYahoo Auctions reported a substantial (though short-term) rise in number of users [6]. In contrast, if the site were to suffer one unplanned 1-minute outage per day, the availability would be (1440/1441) = 99.9%. Even with one 10-minute outage per week, the availability would still be (7*24 hours)/(7*24 + 1/6 hours) = 99.9%. The same availability is achieved, but shorter outages are likely to affect a smaller percentage of the site's subscribers simultaneously, since not all subscribers of a large service are using the site at any given time. (For example, although America Online has 100M registered email users, a typical 1-2 hour outage affects about 1.5% of all users [16].) Outages that affect only a small percentage of users are less newsworthy than those affecting many users, and hence much more valuable to a company's reputation. This is especially true when the cost to a customer of switching to a competitor is relatively low, as it would be for (e.g.) online book retailers; the switching cost is slightly higher for Ebay users who have accommodated "credibility points" in their Ebay profiles, and even higher for email services.

Claim: For end-user interactive services, there exist specific MTTR thresholds below which the user experience is not appreciably disrupted by a transient failure (so that further improvement beyond that threshold yields marginal benefit) and above which it is intolerable (so that users give up or click over to a competitor).

In the human-computer interaction literature, we find thresholds regarding user perception of latency when interacting with a computing system. Miller [25] found that when a user must wait more than 1sec before the system reacts to her command, she perceives the system as "sluggish"; and if she must wait 10sec, she gets distracted and is likely to move on to another task. A recent small study of end-user perception of WWW site latency [1] as well as recent industry-analyst research [33] confirmed that 8-10sec for a page view is the approximate threshold at which users complain of the site being excessively slow, and presumably are then likely to give up. According to [1], most users considered site performance "acceptable" when the page view latency was around 5sec.[3] Hence we might write: T_{stop}=10sec and T_{ok}=5sec, as an initial approximation.

These are relevant because many sites contain machinery for automatic fast failover in the event of a front-end node failure [26], and existing commercial load-balancing products do some of this automatically. If S is the steady-state response latency of the site (i.e. when not operating under partial-failure conditions), then:

- As long as $S+MTTR \leq T_{ok}$ (or $MTTR \leq T_{ok}-S$), the user will not perceive the site as excessively slow, i.e. the failure is completely masked; there is no reason to invest significant effort in lowering MTTR further, and the impact of the failure is negligible.
- If $T_{ok} \leq S+MTTR \leq T_{stop}$, the user will perceive sluggishness but she is unlikely to click "Stop" and give up or move to another site. There may be incremental benefit to improving MTTR; the impact of the failure depends on individual user perceptions.

---

[2] Ebay's outage policy is to grant a free 1-day extension to all affected auction listings if the outage exceeds 2 hours; the site typically receives about 600 bids a minute on 1.9 million items for auction [21].

[3] The variance associated with the 10sec number was high enough that the authors were unable to conclude that there exists a uniform specific latency that users will tolerate before complaining. Also, the threshold for "acceptable" was higher (i.e. users were more patient or forgiving) when incremental page loads rather than all-at-once page views were used.

- If S+MTTR ≥ T_{stop}, the user is likely to give up. The impact of the failure is likely to be the temporary or permanent loss of a customer, perhaps to a competitor.

Because T_{ok} and T_{stop} are technology-independent, they can be used to assess the impact of achieving particular *absolute* levels of MTTR given S (or of MTTR+S together).

Note that this can only work for transient failures that can be handled automatically, as any outage that requires human intervention effectively sets a lower bound on resolution time that is measured in tens of minutes [20]. Exploiting this observation should therefore be a goal of the *design* of the service rather than a goal of managed operations, as we illustrate by example in subsequent sections.

3 Low MTTR and Degradation vs. Outages

Some transient failures can be resolved using simple techniques but in a way that takes time longer than T_{stop}—for example, rebooting a typical cluster node takes under a minute [2]. Although these failures cannot be masked simply as added latency, we can take advantage of another key property of many large-scale Internet services: users may be willing to tolerate other forms of service degradation (besides latency) as long as the amount of degradation is bounded, the duration of degraded service is bounded, and the incidence of degraded service is rare. In particular:

Claim: Users are willing to tolerate temporary service degradation as long as the degraded quality, $0 \leq Quality \leq 1$, is greater than some threshold *Quality_min;* the duration of degradation does not exceed some threshold T_{outage}; and the probability that a given user will experience degradation during a given visit to the site does not exceed $p_{degraded}$ (i.e. the frequency of degraded-service intervals is low from the point of view of any given user).

Performability analysis [23] provides a formal way of constructing a probability measure that captures the varying quality level at which a system performs over a given observation period. Though originally designed to capture such behaviors for long-running systems, it has recently been proposed [24] that performability be applied to Internet services by capturing, for example, the incremental latency experienced by user requests under high server load, or the probability that a user request is simply rejected (admission control) under high server load. We would like to generalize this notion of degradation to cases where a response is delivered successfully and with acceptable latency, but the quality of the answer is less than perfect – e.g., the answer is imprecise, stale, etc. This requires that the service software be structured in such a way that partial failures map directly to specific user-visible deviations in service behavior. Fortunately we have several existence proofs that this design is possible, though we do not have a systematic set of guidelines for how to achieve the design for a particular application; we give an example below.

If duration of degradation is greater than T_{outage}, we declare that an outage has occurred; what we mean is that the transient failure is now much more likely to be perceived by users as a real outage, i.e. a user's tolerance for "temporary" degradation runs out after T_{outage}, and a user is more likely at that point to abandon the site either temporarily or permanently. T_{outage} may vary from service to service, from user to user, and may trend over time, i.e. as more users move to broadband connections the baseline tolerance for long latency may decrease.

To make the above claim useful as a design guideline, we must answer the following questions:

1. How can we determine the user-centric thresholds *Quality_min*, T_{outage}, and $p_{degraded}$ for a given service?

2. For a given service, how are quality and probability of degradation for a given user actually measured? And specifically, what is the quantitative effect on these metrics when partial failures occur in the service?

3.1 Estimation of Quality_min, T_{outage}, and $p_{degraded}$

By definition, estimating these thresholds requires observing user end-user behaviors. Since direct observation and interrogation may be costly and impractical, large sites attempt to indirectly infer user intentions by monitoring behaviors across large user populations. For example, one way to estimate a value for T_{outage} is *abandonment detection* [19]: if a user abandons an incomplete task while visiting a site, one possible interpretation is that the user's patience has run out or that she unwilling to use the service until the temporary degradation is repaired. Large sites are not generally forthcoming about the way they arrive at specific T_{outage} numbers, but (for example) America Online's email server reportedly operates with a T_{outage} of about one minute [16].

Similarly, thresholds for *Quality_min* are service-specific, but one (admittedly extreme) anecdotal example involves the physical move of the Inktomi search engine from one datacenter to another [2]. During that move, which took many hours, the service remained online but delivered search results from only half its database, i.e. *Quality*=0.5. (Actually, this is a *lower* bound on the quality; since not all searches touch every part of the database and frequently-hit pages are replicated on mutliple database partitions, the quality of the *typical* search was higher than 0.5.) Presumably, therefore, $Quality\_min \leq 0.5$ for this example, in which we have used *harvest* [10] as the measure of search results quality. Similarly, a recent survey by Jupiter Media Metrix [19] reports that for certain classes of users, a 10% decrease in "site performance" leads to a 5% increase in site abandonment, and also that users' method of accessing a site (i.e. dialup vs. broadband) result in different expectations of site performance.

Some sites, such as Anonymizer.com, Hotmail and Yahoo!, offer both a free service and a premium for-pay service. In the case of Hotmail and Yahoo!, the main benefit of the pay service is more per-user storage space for email, but in the case of Anonymizer, it is actually faster performance. Whether Anonymizer prioritizes their premium users or deliberately degrades service for their free users is unclear, but it introduces the possibility that users may be willing to tolerate higher values of $p_{degraded}$ in exchange for the service waiving a usage fee.

3.2 Example: *Quality* and $p_{degraded}$ under Rolling Reboot

As a specific example of mapping partial failures to effects on *Quality* and *p*, consider the Inktomi search engine as described in [2]. Every user query hits every node in a farm of N workstations. On average, each node contributes 1/N of the result of every search, though not every search will yield hits on every node and some frequently-requested pages are replicated on multiple nodes. To avoid reducing throughput during transient node failures, the search engine returns results only from the *k* nodes that responds to the search query within a prespecified timeout; we refer to the fraction *k/N* for a given search as the *harvest* of that search, between 0 and 1 [10].

Like many Internet sites, the search engine nodes are rebooted on a rolling basis to rejuvenate them. If we assume that in the worst case every search hits all the nodes, the effect of a single node being rebooted is a reduction in harvest of 1/N, implying a degraded quality of 1-(1/N). (Typically $N \approx 100$ in large clusters such as these.) In fact 1-1/N is a lower bound on quality, since some searches may not be affected at all. But in general the degradation is experienced by all users using the service during the rolling-reboot period.

If each node takes time *r* to reboot, *N* nodes will take time *Nr* to be rebooted sequentially. If the decrease in quality is perceptible in practice while *any* node is rebooting, then it had better be the case that $T_{outage}>Nr$, otherwise *any* user using the service during the rolling reboot is likely to abandon (since the outage is taking too long to resolve). One possible remedy would be to selectively do rolling reboots in "chunks", so that a given "chunk" of rolling reboot does not degrade quality for longer than T_{outage}. Another remedy, if it is determined that *Quality_min* is less than 1-1/N, would be to reboot *k* nodes at once, resulting in a quality (harvest) reduction to 1-(k/N). As long as $1-(k/N)\geq$ *Quality_min* and $Nr/k < T_{outage}$, we will stay within the user's tolerance thresholds for both amount of degradation and duration of the incident.

If rolling reboots are "preventive maintenance" induced by sysadmins, we can control *p* by setting the frequency of rolling reboots. In particular, if the rolling reboot is initiated with period *T*, then under the simplifying assumptions that load is uniform at all times and a given user's visits to the site are uniformly randomly distributed, a given user has probability *Nr/T* of hitting the site during a rolling reboot and thereby experiencing degraded service. To make it concrete, if *T* is once a day, *N* is 100 nodes and *r* is about 30 seconds, then this probability is .035; it could be made smaller by scheduling reboots during predictable "troughs" in the workload if such troughs exist. In this example, then, it had better be the case that $p_{degraded}\geq 0.035$.

This method can be extended to model transient node failures by using a node's MTTF to approximate how frequently *some* node will be out of service when a user request arrives. If rolling reboots and node failures are independent (in practice they are not, since reboots often have the effect of preventing failures before they happen [11]), the probabilities of hitting the site during a rolling reboot and hitting the site while a given node has suffered a transient failure can simply be added to get an upper bound on the probability of user-perceived degradation in quality.

Rolling reboot is admittedly a simple example of controlled partial failure. Nonetheless, the example serves to show that in some cases the specific user-visible effects of partial failures can be quantified and that measurement tools are available in the Internet community whose goal is to capture changes in user satisfaction as a result of such effects.

4 Related Work

We have begun to explore fast-recovery techniques such as recursive restartability [3] for systems designed to operate in a degraded fashion under independent partial failures. Continuing work focuses on mapping abstractions needed by applications directly to design techniques that allow partial failures to map to specific types of service degradation.

The nature of Internet service workloads presents other reasons for optimizing MTTR as well. Because most requests are independent, short-lived, and idempotent, retrying a failed request is a common end-user strategy. Merzbacher and Patterson [22] find that for some common services such as search, e-tailer and directory, if a failed request was retried within about an hour, the retry succeeded between 78% and 100% of the time. In addition, Xie et al. [30] find that when end-user behavior (tendency to retry and approximate retry interval) are considered, sites with lower MTTR and lower MTTF are perceived by users as more available than sites having the same Availability (MTTF/MTBF) but higher MTTR. This conclusion is currently based on a sophisticated model of end-user retry behavior rather than on direct observation, but it is tantalizing that these two quite different approaches point to conclusions similar to our own. We anticipate synergy between the two approaches in the future.

TACT [31] presents a general framework for measuring degradation of service, for the particular case in which degradation is measured as user-perceived inconsistency. By defining application-

specific units of consistency in a system that uses replication to improve availability, the tradeoff between consistency and availability becomes quantifiable. Similarly, online aggregation [15] treats the special case in which compute time is explicitly traded for precision of a numerical query, such as the computation of a statistic over an extremely large dataset. Both approaches assume that the *threshold* for "acceptable" consistency or precision is externally determined, as we have assumed for $p_{degraded}$, Quality_min and T_{outage}.

5 Summary

Raising MTTF can never guarantee a failure-free interval (unless MTTF is infinite), but lowering MTTR can mitigate the impact of a failure during that interval. In many cases, we can build services that fail-and-retry at multiple levels, as long as recovery is fast enough and perceptible degradation occurs infrequently enough. This can be done at two levels: "front end" failures may be recoverable within T_{stop}, the time at which a user becomes frustrated enough to abandon the site; "back end" failures may be recoverable within T_{outage}, the time at which a temporary degradation in service quality appears to the user to be long-lived and causes the user to abandon the site.

In both cases, MTTR's are directly measurable and can be used to estimate the impact of a specific outage, provided good estimates for Quality_min, $p_{degraded}$ and T_{outage} can be arrived at. We have not suggested any specific methods for estimating these thresholds, but one possibility is the inference of these thresholds from observations of large-group behaviors by tracking events such as abandonment; such tracking is widely offered by measurement services such as Keynote, Porvio, and others. In contrast, manufacturer MTTF claims may be difficult to verify directly and do not capture the end-user impact of outages. (Indeed, systems with low MTTR may appear to behave acceptably even when the component MTTF's are also low, as long as the MTTF is sufficiently long that a user will not perceive service degradation with probability greater than $p_{degraded}$.) A focus of the ongoing research agenda is clearly to derive guidelines for architecting applications such that partial failures do in fact map directly to measurable effects on service quality and probability of receiving degraded service, and to capture this degradation in simple and tractable analytical models similar to those used for performability.

By focusing this discussion on lowering MTTR, we are not advocating that less effort be spent on debugging or operations, nor do we that all "hard" failures can be masked through redundancy to achieve lower MTTR. Nonetheless, the above observations—measurability and relevance of MTTR as an availability metric, and the ability to exploit low MTTR to mitigate the effects of partial failures—suggest that as a community we should more aggressively pursue opportunities for improvement based on design for fast recovery: it has direct correlations to user satisfaction, and is benchmarkable to boot.

Acknowledgments. Thanks to the Recovery-Oriented Computing research group, its advisors, the anonymous reviewers, Amr Awadallah (Yahoo! Inc.), Peter Chen (Univ. of Michigan), Steve Gribble (Univ. of Washington), Sam Pullara (BEA Systems), and Lisa Spainhower (IBM) for contributions to and discussion of this draft.

References

[1] Nina Bhatti, Anna Bouch, Alan Kuchinsky. *Integrating user-perceived quality into Web server design*. Proc. WWW-9.

[2] Eric Brewer. *Lessons from giant scale services.* IEEE Internet Computing 5(4), July-Aug. 2001.

[3] George Candea, James Cutler, Armando Fox et al. *Minimizing time to recover in a recursively restartable system.* Proc. Intl. Conf. on Dependable Systems and Networks (DSN) 2002, Bethesda, MD, June 2002.

[4] Tim Clark. *Ebay recovers after outage.* CNET news.com, May 21, 1999. http://news.com.com/2100-1017-226166.html

[5] Tim Clark and Scott Ard. *Ebay blacks out yet again.* CNET News.com, June 13, 1999. http://www.idg.net/crd_sun_92801.html

[6] Chet Dembeck. *Yahoo cashes in on Ebay's outage.* E-commerce Times, June 18, 1999. http://www.ecommercetimes.com/perl/story/545.html

[7] P. Enriquez, A. Brown, and D.A. Patterson. *Lessons from the PSTN for dependable computing*. In Workshop on Self-Healing, Adaptive, and Self-Managed Systems (SHAMAN), New York, June 2002.

[8] Melanie Austria Farmer and Robert Lemos. *Microsoft blames technicians for massive outage.* CNET News.com, January 24, 2001. http://news.com.com/2100-1001-251427.html?legacy=cnet

[9] Armando Fox. *Toward recovery-oriented computing.* Proc. 28th Intl. Conf. on Very Large Data Bases (VLDB 2002), Hong Kong, China, August 2002.

[10] Armando Fox and Eric A. Brewer. *Harvest, yield, and scalable tolerant systems.* In Proc. HotOS-VII, Tucson, AZ, 1999.

[11] S. Garg, A. Puliafito, M. Telek, K.S. Trivedi. *On the analysis of software rejuvenation policies*. Proc. of the 12th Annual Conf. on Computer Assurance, June 1997, pp. 88–96.

[12] Auction Guild news posting. *Ebay Outage 29 April 02.* http://www.auctionguild.com/generic92.html

[13] Liane Guthrow. *Ebay outage leaves users out of luck.* CNN.com Technology, July 13, 2000. http://www.cnn.com/2000/TECH/computing/07/13/ebay.outage.idg/

[14] Clare Haney. *AP IT Summit: Sun altered policy after Ebay outage*. IDG News Service, November 18, 1999. http://www.idg.net/crd_sun_92801.html

[15] Joseph M. Hellerstein, Peter J. Haas, Helen J. Wang. *Online aggregation.* Proc. Intl. Conf. on Management of Data (ACM SIGMOD), Phoenix, AZ,1997.

[16] Carl Hutzler, *Workshop on dependability of Internet business systems: AOL Email*, presentation at E-Commerce Workshop of Intl. Conf. on Dependable Systems and Networks (DSN) 2002, Bethesda, MD, June 2002.

[17] M. Kalyanakrishnan, R. K. Iyer, and J. U. Patel, *Reliability of Internet hosts: A case study from the end user's perspective*, Computer Networks, vol. 31, pp. 45–57, 1999.

[18] Darrell Long, Andrew Muir, Richard Golding. *A longitudinal survey of Internet host reliability.* In Proc. of the Symposium on Reliable Distributed Systems, 1995.

[19] Jupiter Research. *Tying performance to profit: track abandonment to improve loyalty*. Jupiter Media Metrix research report STP01-C04, June 2001.

[20] Steve Levin. *Exploring the realities of Internet uptime.* Codesta Strategic Technology Consulting report, July 2002. http://www.codesta.com/knowledge/management/uptime_realities/

[21] Kora McNaughton. *Ebay suffers prolonged outage.* CNET News.com, May 3, 1999. http://news.com.com/2100-1017-225285.html

[22] Matthew Merzbacher and Dan Patterson. *Measuring end-user availability on the Web: practical experience.* Proc. Intl. Performance and Dependability Symposium (IPDS) 2002, held in conjunction with Intl. Conf. on Dependable Systems and Networks (DSN) 2002, Washington DC, June 2002.

[23] J.F. Meyer. *Performability: a retrospective and some pointers to the future.* Perf. Eval. 14, pp.139-156.

[24] J.F. Meyer and Lisa Spainhower. *Performability: an e-utility imperative.* In Proc. ICSSEA 2001.

[25] R.B. Miller. *Response time in man-computer conversational transactions.* Proc. AFIPS Fall Joint Computer Conference, 1968.

[26] David Oppenheimer and David A. Patterson. *Studying and using failure data from large-scale Internet services.* In Proc. SIGOPS European Workshop, Sept. 2002

[27] Lisa Spainhower. *Why do systems fail? Review studies 1993-1998,* Presentation at IFIP WG 10.4 (Dependable Computing and Fault Tolerance) 41st meeting, St. John, US Virgin Islands, Jan 4-8, 2002

[28] Brian Sullivan. *Ebay suffers three outages in three days.* CNN.com Sci-Tech, April 11, 2002. http://www.cnn.com/2002/TECH/internet/04/11/ebay.outages.idg/index.html

[29] Troy Wolverton. *Ebay hit with site slowdown.* CNET News.com, April 29, 2002. http://news.com.com/2100-1017-894196.html?tag=cd_mh

[30] W. Xie, H. Sun, Y. Cao and K.S. Trivedi. *Modeling of online service availability perceived by Web users.* Technical Report 2002, Center for Advanced Computing and Communication (CACC), Duke University.

[31] Haifeng Yu and Amin Vahdat. *Design and evaluation of a continuous consistency model for replicated services.* In Proc. OSDI 2000.

[32] Ji Zhu, James Mauro, Ira Pramanick. *System recovery benchmarking*. Proc. DSN Workshop on Dependability Benchmarking, Bethesda, MD, June 2002.

[33] Zona Research (now the Sageza Group). *The Need for Speed*. Research report, July 1999.