Obtaining Five "Nines" of Availability for Internet Services

Tzvi Chumash
tzvika@cs.rutgers.edu

Abstract

Ensuring high availability of a service is a costly task. This paper discusses arguments for and against obtaining five "nines" of availability (99.999% availability) with the assumption that the clients operate at only three nines of availability (99.9% availability). Based on a simulation created for this purpose and real market data, the paper takes the position against obtaining five "nines" of availability on account of user impact. However, for highly profitable Internet services, obtaining higher levels of availability will increase their profit regardless of the user experience.

1 Introduction

Service availability measured in "nines" relates to the amount of time (per year) that a service is up and running. The converse is even more useful and shows the amount of unscheduled downtime per year that a given service level provides (See table 1).

Each level of service is associated with a cost, which may grow by orders of magnitude when the level is increased. The basic assumption in this paper is that the average machine has three "nines" of availability and therefore has only about 8.76 hours of downtime per year.

The paper addresses two issues: first, the need of a general Internet service to upgrade its availability on account of its users. The second issue is the need of a high-profile Internet service to obtain higher availability regardless of the user experience.

Service Level	Uptime per Year	Downtime per Year
90%	328.50 days	36.50 days
99%	361.35 days	3.65 days
99.9%	364.64 days	8.76 hours
99.99%	364.96 days	52.56 minutes
99.999%	~365 days	5.26 minutes
99.9999%	~365 days	31.54 seconds

Table 1: Availability in terms of "Nines"

Section 2 of this paper discusses some relevant availability issues. Section 3 deals with the costs involved with both service downtime and achieving high availability. Section 4 discusses optimization of the service level. Section 5 discusses the availability simulation, and section 6 concludes the paper.

2 User/Service Availability

Availability of a system is defined as the probability that a system is available [1]. There are many gaps in this definition, as it does not include any context. A hard drive can be available to an operating system, yet if the network cable is unplugged, can we really say the system is available? What if the local router is down? Obviously, system availability needs to be looked upon in a broader end-to-end definition such as in [2]: One cannot ignore the user's experience when service availability is discussed.

For the purpose of this paper, a reverse question is posed – if a user of a service has low availability, is there even a need for the service to be highly available? One way to answer this question is to attempt to see the service from the users' point of view. A simulation was built for this purpose, and the details and results are described in section 5 of this paper.

3 Costs

Costs are associated with both ensuring a service level and as a result of service downtime. As the level of service or the length of the downtime period increases, the cost increases, and vice versa. Downtime costs can be enormous, downtime for a NYC retail stockbroker for example could be as high as $6.5 million per hour [3]. The following subsections refer to the two cost models.

3.1 Downtime Costs

Costs associated with service downtime can be categorized by causation: direct loss of income, or damages due to a breach of contract. As can be seen in table 2, an hour of downtime for eBay INC in 2005 would be equivalent to a loss of $520,000 in income (due to lost transactions).

	Approx. $/Yr	Approx. $/Hr
eBay	4.55 Billion	520,000
Amazon.com	6.92 Billion	790,000
Google	6.14 Billion	701,000

Table 2: Major Internet service providers reported yearly earnings [4,5,6] and the equivalent per hour earnings (assuming random distribution).

Costs due to breach of contract are the compensation (damages) paid to a customer because the service level agreement was breached by the service provider. These damages should offset the customer's loss of income due to an outage (to which the service provider is liable).

3.2 Service Level Costs

In order to obtain service levels higher than three "nines" (especially five or six "nines"), investment has to be made not only for better hardware and software, but also for duplicating all production related resources. On one end of the scale is a standalone low-end server that is placed in an office and is handled by a part time administrator. On the other end of the scale, the same service is provided by a cluster of high-end machines, placed in well-maintained server rooms, supervised by numerous full-time administrators, in possibly two or more sites, each with redundant ISPs. Naturally, the costs of additional locations, salaries and additional infrastructure may be extremely high [7].

Even though one can find advertisements for dedicated service hosting with five "nines" of availability for under $100 per server a month (~$1000/yr), this paper assumes that the service may be more complex than the standard website-oriented services those hosting companies provide, and therefore may be a lot more expensive.

In order to discuss an optimal service level (in the next section), there's a need to estimate the costs involved with increasing levels of availability. As the main contributor to cost, we can focus our cost model around the manpower,

and estimate that we can obtain service at $50/hr. Furthermore, our basic level of availability is three “nines”. Assuming there is relatively little to no maintenance involved with a “99.9%” service, 10 hours of service per year should be enough (~$500). As higher levels of availability are required, it is possible that each higher level of availability will require ten times the hours invested in the previous level. This is the basis for the cost model suggested in the following section.

4 Optimal Service Level

Following the ideas and data from the previous section about costs, it is possible to develop an economic model that will help in choosing the optimal service level. Assuming that corporations are rational they would try to maximize their profits:

Profits = Total Revenues – Total Costs

As the service level improves, total revenue increases, but so does the cost of obtaining the improved service level. Because these variables are not independent, cost minimization can be used instead of profit maximization:

Total Cost = Cost of Downtime + Cost of Service

Since the cost of downtime decreases as availability increases, and the cost of service increases as availability increases, the total cost curve (which is the vertical summation of those two functions) will have a minimum.

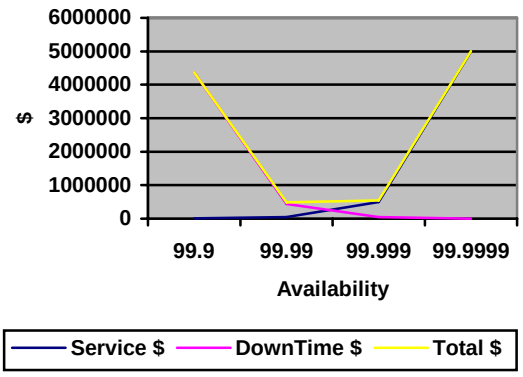

Figure 1: Choosing an optimal service level by cost minimization

Suppose the average monetary measure of downtime cost is $500,000/hr (a possible value based on the data in Table 2), and suppose it is possible to estimate the cost of the level of service by using a model such as:

Cost of Service = 500 / (1-Availability)

The result would be a graph similar to figure 1. As is evident in figure 1, for this cost model, the optimal choice of availability would be four “nines”, as that is the point with the minimal cost.

Tweaking the model further, if we do not change the cost of service, shows that five “nines” would be the optimal choice if the downtime cost goes above $570,000/hr. Six “nines” would be the optimal choice only when the downtime cost exceeds $58 million/hr.

5 Experiment

A simulation was created in order to see the difference in the average impact of a service provider with three “nines” and one with five “nines” on the availability of the clients (all at three “nines”). This was done in order to form a more ‘informed’ position – do the service providers need to invest in high availability if their customers have low availability?

5.1 Simulation Expectations

The expectations from the experiment were to see the impact unscheduled downtime of a server would have on its users: I wanted to see the difference from the end-user's point of view between a server with five "nines" of availability and one with "three" nines of availability. Specifically, I wanted to see whether there is any effect at all on the client's average availability, and attempt to quantify the effect, if it exists.

5.2 Implementation

In this simulation implementation, there is no actual interaction between clients and servers, but just interaction between states. The server state consists of a binary status (either up or down) and downtime left for the year. Each client state consists of:

- Own downtime (in seconds)
- Downtime left for the year (in seconds)
- Number of server connection attempts
- Server downtime (in seconds)

The granularity is in seconds, and in each second, the server randomly decides if it is up or down (based on the allowed downtime that is left for the year), and each of the clients randomly decides whether it is up or down, and in case it is up – whether it should attempt to connect to the server.

Operation from the client's perspective:

- If a client is down in a cycle, then its downtime counter is incremented by one, and the downtime remaining counter would be decremented by one.
- If a client is attempting to connect to the server in a cycle, then the server-connection-attempts counter will be incremented by one.
- In case of an attempted connection to the server, if the server is down in that cycle, the server downtime counter would be incremented by one.

All the decisions about server and client downtime, as well as all attempts by a client to connect to the server are based on random number generations.

Attempts by clients to connect to the server are limited to 10% of the time by each client. This translates to an average of just under two and a half hours per client per day.

5.3 Results

At each run of the simulation, the server availability was changed, while the client availability remained constant at '99.9%'.

Client availability is calculated the following way:

$$Availability \approx \frac{Client\ Downtime + Server\ Downtime}{Total\ Time}$$

The server downtime refers to the viewpoint of the client (i.e. the client must have tried to connect to the server first).

The server availability from the client point of view is calculated the following way:

$$Availability \approx \frac{Server\ Downtime}{Total\ time\ client\ \ tried\ to\ access\ \ the\ server}$$

These results are summarized in table 3.

Server Availability	Client Availability	Server from Client Side
99.9%	99.89%	99.99%
99.99%	99.899%	99.999%
99.999%	99.8999%	99.9999%
99.9999%	99.9%	100%
100%	99.9%	100%

Table 3: Experiment results

5.4 Analysis

As can be seen from table 3, there is a difference (degradation) in the availability of the client due to a service with lower availability. However, from the client's perspective, the differences between the server being 99.99% available and 99.9999% available are marginal (less than 5 minutes per year). Even though the difference from a %99.9 service is much higher, it boils down to about 50 minutes per year, which is not substantial.

5.5 Improving the Simulation

I believe there are at least three flaws in the implementation:

- The simulation is completely random both on the client side and on the server side and therefore, the created scenarios do not truly mimic real-world behavior (distribution-wise).
- There is no scalability: doubling the number of clients causes the runtime to increase drastically
- There are no requests, and no levels of availability. The supported status is binary (server up, server down), no partial failures.

Regarding the randomness problem, even though the starting point of the downtime itself is random, the length of the distributed downtime is always set to one second. My implementation distributes the downtime randomly across a year and therefore if the total downtime is 32 seconds per year (six "nines"), then in real life that time may be clustered to one block of 32 seconds, while in my implementation those 32 seconds would be randomly distributed across the year (average of about one second per eleven days). A second problem with the distribution is that the clients are connected to the server some set percentage of time per year, but that time is distributed randomly over the year, and not in a daily pattern (e.g. 8am-5pm) there is also no support for time zones.

Regarding scalability, the random number generator is relatively slow, and since in each simulator second there is a need for two random numbers per client, and one per server, and there are 31.5 million seconds in the simulated year, there are 63,000,000n+31,500,000 (where n is the number of clients) random number generations per simulation and thus using a high number of clients is prohibitive.

Regarding granularity, there are no actual clients connecting to a server, and no requests or replies. The server can only be up or down. Clients can't measure partial responses, or performance.

In future work it might be possible to address these three issues in order to obtain results that are closer to the real world runtime environments.

6 Conclusions

Based on my experimental results, it would seem that the users of a three "nines" service would encounter only slightly lower availability than if the same service had five "nines" of avail-

ability (on average 0.01% lower availability than if the service was 100% available). This translates to less than one hour a year of unavailability for a heavily used service. From the user's point of view, this might be acceptable (especially since the user might not know that the fault is with the service, see [2]). Therefore, it is my belief that when service providers consider upgrading their availability, they should not heavily consider the impact of a lower availability on their users.

However, as some services (such as monetary operations) exchange money in extremely high volumes (i.e. above a threshold as defined in section 4), then that service should get the appropriate amount of availability (even if it is 99.9999%) as long as the cost is minimized.

References

[1] Jim Gray. Why Do Computers Stop and What Can Be Done About It?, Technical report 85.7, Tandem Computers, Cupertino, CA. June 1985.

[2] Matthew Merzbacher, Dan Patterson. Measuring End-User Availability on the Web: Practical Experience

[3] T. Sweeney. No Time for DOWNTIME — IT managers feel the heat to prevent outages that can cost millions of dollars. InternetWeek n.807, 3 April 2000.

[4] eBAY INC. Q4-05 Earnings Release

[5] Amazon.com. 2004 Annual Report

[6] Google INC. 2005 Form 10-K

[7] W. Sawyer. Case Studies from HP's 5nines Program: The Cost of Moving from 99% to 99.999% Availability. HDCC, May 8th, 2001.