Extreme Availability: Determining the Limits of Availability in Commercial Systems

David W. Coleman, Todd W. Mummert

IBM T.J. Watson Research Center, Hawthorne, NY, 10532, USA
{xinu, mummert}@us.ibm.com

ABSTRACT

High availability has become increasingly important to businesses with the advent of the Web and various associated e-commerce services. This paper describes the key attributes of the very highest availability systems. Such systems and their associated support environments are qualitatively different than other systems, even other high availability systems; for the purposes of this paper these systems at the current limit of possible availability are termed *extreme availability* systems.

Keywords: High availability, fault tolerance, commercial systems

1. INTRODUCTION

The basic requirements that any trading floor application for the New York Stock Exchange (NYSE) needs to meet are fairly easy to define. The system needs to be extremely reliable, highly available, very fast, and highly scalable. While these requirements are not unusual *per se*, the levels needed by the NYSE are well beyond what most commercial off-the-shelf software provides. The NYSE is one example of an *extreme availability* environment, where no outage during trading hours is acceptable.

Historically, extreme availability has been a specialty area, of interest mostly for space-craft systems and those systems where failure would cause significant destruction and loss of life. The usual techniques for improving availability, such as redundancy and independent backup systems, have been used with great success in such systems.

Commercial systems are extremely cost sensitive, although the degree of sensitivity varies by industry. Most businesses think about availability in much the same way that they think of disaster recovery costs – they determine the cost per hour (or sometimes per minute) of an outage, and then take appropriate steps to mitigate outages using the outage cost per hour as a cap on potential mitigation expenses. One corollary of this approach is that when business critical systems are computerized the willingness to spend to increase availability goes up sharply, since the expense cap becomes tied to the entire business revenue stream.

Given this background, it should come as no surprise that e-commerce is causing many companies to become much more concerned about high availability, and in some cases businesses are starting to look seriously at what is required to achieve extreme availability. At the moment, financial companies are most interested in extreme availability, partly because they are often the most advanced in computerizing their business processes and partly because they tend to have the highest cost per hour for outages. It appears, however, that a secondary effect of the shift to e-commerce is that high availability is becoming a critical competitive attribute – and that extreme availability may in turn start becoming a competitive differentiator.

There are a few key ideas which appear over and over when considering the various aspects of availability. They are listed here both as an introduction to the concepts and as a convenient reference point.

MTBF stands for mean time before failure, or sometimes mean time between failures. These two phrases are equivalent only for those cases where the curve which describes the failure probability over time is a negative exponential curve, but as it happens such a curve is frequently a good fit for hardware failures, particularly for electronic or electrical equipment. In summary, MTBF is a measure of how long something is likely to stay up. Generally MTBF values are specified as some number of months or years.

MTTR stands for mean time to repair. It is a measure of how long it takes to fix something once it breaks. Note that the repair time includes whatever time it takes to diagnose the problem, acquire the repair components, pull the failing component, do whatever repair is needed, and reinstall the now working component.

SPOF stands for single point of failure. A single point of failure is a system or environmental component which, when it fails, takes down the entire system. The significance of a SPOF is that the reliability of that component will be an upper bound on the overall reliability of the system. In general, any SPOF is a weak link in a high availability chain.

This paper provides a background in the general approaches and concerns for very high availability systems. While these general approaches are well known in certain areas, it is useful to revisit them in the

context of distributed e-commerce systems to understand what aspects are most important in that environment. To do this, the paper follows the general overview with an overview of the techniques and approaches used by those organizations which are most aggressively pursuing extreme availability. Finally, some general information is given from case studies of such extreme availability systems, to indicate what the limits of the commercial state of the art are today.

2. FAULT TOLERANCE

Fault tolerance is the study of how to design systems or subsystems so that they continue to operate correctly in the presence of components which fail. Normally the measure for a successful design is an increase in the overall system MTBF. There are some curious tidbits from this area of study, such as the fact that redundancy can actually *decrease* system MTBF. To understand how and why such things can happen, it is necessary to understand how component MTBF can impact system MTBF. [1] [2]

2.1 MTBF – Underlying Assumptions

Large systems are composed of many parts, and to compute the MTBF for such large systems we need a way of composing the MTBFs for the parts. In general,

$$P(\text{exactly one of } E_1 \text{ and } E_2 \text{ occurs}) = P(E_1) + P(E_2) - (P(E_1) * P(E_2)) \quad (1)$$

Note that for sufficiently small probabilities, the product term will be quite small with respect to the overall value, and hence we can approximate the probability that exactly one of E_1 and E_2 occurs with the sum of the probabilities of E_1 and E_2. (Since for fault tolerance we are dealing with failure probabilities, and most failure probabilities for IT systems are less than 5%, this simplification is useful and fairly accurate. Note that 5% * 5% is 0.25%, while the sum is 10%, and that further the accuracy of the approximation improves as the probabilities get smaller.)

Because we are interested in the expected life span (Mean Time Before Failure), rather than the probability of failure in a given time span, we must find a new function of P(E) which yields this information. For random probability distributions this can be quite difficult, but it turns out that a useful approximation is to assume that the probability is memory-less, which means that the probability of failing in the next time interval is independent of how long it has already been in service. With that assumption, we get

$$MTBF(E) = 1/P(E) \quad (2)$$

It turns out that many IT devices actually exhibit a bathtub curve probability, with a high infant mortality rate and a high wear out rate joined by a long period of essentially constant and low mortality. Manufacturers can convert such devices into memory-less devices by using a burn-in period to eliminate infant mortality, and setting the nominal end of life (or end of warranty) prior to the rise due to end of life mortality.

For a device with a 5% probability of failure in any given month, we will therefore get a 20 month MTBF. Most systems are composed of many modules. If the modules fail independently, we can compute the probability of the system failing. In general, for any set of N identical parts the MTBF for the entire set is 1/N times the MTBF for one such part.

2.2 Failure Modes

Since we are interested in fault tolerance, it is natural to ask about the failure modes of the various devices we are considering. Unfortunately, this question is extremely hard to answer with complete accuracy for most devices. Further, the answer to such a question is tightly tied to the exact technologies used, so even having the answer for a given device doesn't necessarily help in determining the answer for another similar but not identical device.

There are a few general principles which can help increase fault tolerance. The first such principle deals with what are commonly called soft, or transitory errors. Soft errors have the property that the probability of occurrence is independent of the work being done, and further they do not always occur even when the same work is done. There can be various causes for soft errors, but in terms of fault tolerance the ultimate goal with soft errors is always the same – to mask the occurrence of the error in such a way that there is no immediate fault generated. The solution in these cases always comes down to redundancy, and typically it is implemented via error correcting codes. While ECC is a large and complex field of study, it is also sufficiently well understood that such codes are in widespread use by device manufacturers to control transitory errors. Such codes are used in all disk drives, and on almost all communication channels.

Workload independent hard errors (also known as non-transitory or permanent) can also be handled by manufacturers by using redundancy, but in this case the redundancy needed is additional hardware.

Unfortunately there are several additional classes of errors which do not easily lend themselves to general solutions. Workload dependent errors, such as the floating point problem in Intel processors, are one major example. Additionally, many devices have partial failure modes, where things are neither working completely nor failed completely. While most manufacturers attempt to identify such problems with testing, even the most rigorous testing will occasionally miss such problems.

There is one final general approach to dealing with failure which can be helpful, both in the real world and for theoretical analysis of fault tolerance. This approach is called failstop, and it works by forcing all faults into the same terminal state, namely the stopped state. This approach significantly aids in the design and operation of fault tolerant systems, and as will be seen below it also aids in the analysis of the behavior of such systems.

2.3 Fault Tolerance Approaches

In general, fault tolerance requires two things, namely a way of detecting failure and enough redundancy to allow the failure to be overcome.

Under certain conditions, it is possible to turn any device into a failstop device. The simplest way to do that with non-failstop components is to duplicate the device, and then use some kind of comparator to see if the outputs of the two copies of the device agree. If the outputs agree, the device continues to work. If the outputs disagree, the device stops. For this process to be feasible, of course, it must be possible to build the needed output comparator. The major drawback to this simple form of failstop creation is the impact on MTBF. Because there are two copies of the basic device, the failstop device has an MTBF which is no more than ½ the MTBF of the original device. This can of course be mitigated by using more copies of the device; with three copies (and a more sophisticated comparator) the failstop device has an MTBF of 5/6 of the MTBF of the original device. (An additional advantage of triplex or higher redundancy arises if soft errors predominate in the failure modes for the base components. In that case the comparison process will completely eliminate the soft errors, and significantly boost the overall MTBF.)

Although building failstop devices doesn't sound obviously attractive at this point, things become better once higher layers are built on top of such failstop devices. With non-failstop components, the device must necessarily fail as soon as there is no longer a majority of devices working. With failstop components, the device can continue to work as long as at least one of the components is working, since each component will either work correctly or stop. While this won't ease the redundancy cost of the additional hardware, it has a substantial impact on MTBF. With a duplex device, the overall MTBF moves from ½ the component MTBF to 1.5 times the component MTBF. With a triplex device, the overall MTBF moves from 5/6 the component MTBF to 11/6 times the component MTBF.

The most interesting advantage of building with failstop components only appears with one additional assumption. If it is possible to replace failed components without taking down the entire device, and synchronize the new component if necessary, then the MTBF calculations change significantly. The device will fail if all but one of the components are unavailable, and then the remaining component fails. This yields the following equation for a device made with n copies of a failstop component:

The MTBF for the n component device with repair is:

$$MTBF/n * (MTBF/MTTR)^{n-1} \quad (3)$$

or rewriting slightly

$$MTBF^{n} / (n * (MTTR)^{n-1}) \quad (4)$$

Using our usual 95% reliable components, and assuming that repair takes a full day, we find that the MTBF for a duplex device is made from 20 month MTBF components is approximately 1000 months, or roughly 80 years. Using the same components, a triplex device will provide an MTBF of approximately 80,000 months, or roughly 6500 years.

Note that using a comparator and duplex or triplex (or more) is not the only way to provide fault tolerance. The key to fault tolerance is spare hardware and some way to detect failure, with the spare hardware used to correct the detected fault. As an example of another alternative approach, consider RAID 5 disk drives. What is done for RAID 5 is to use a redundant coding on the, with the extra drives needed by RAID 5 used to increase storage to hold the extra bits required by the redundant encoding. [3] [4] [5]

2.4 The Bad News

From the preceding discussion, it is clear that whenever we have workload independent failure modes, either soft or hard, and devices which are memory-less in their failure mode, which are relatively reliable and easy to repair, it is possible to build fault tolerant devices with very high MTBF.

An obvious question is then what large system components do not exhibit the needed properties. And unfortunately there is one key area which does not fit the picture – software. Even worse, the characteristic which is least compatible with the assumptions is the primary assumption behind much of the fault tolerance work, namely the assumption that most failure modes are workload independent.

When most of the failures are a function of the workload done, no voting scheme which uses multiple copies of the same software increases reliability.

As a practical matter, most people designing systems for high availability deal with the issue of software faults by relying on one of the demonstrated advantages of software, namely that the defect rate decreases with use. Thus, it is common for systems with high sensitivity to failure to prefer software which has been in wide or long term use elsewhere to take advantage of the

improved defect rate. Additionally, there is likely to be a much greater emphasis on testing for such systems.

One additional area of workload dependent failure warrants special attention currently. This is the area of security issues and faults. While there have been security issues and bugs since the earliest days of computing, even in systems which were designed from the ground up with security in mind, the Internet has increased the significance of such issues. The problem is that the Internet makes networked computers more widely available. While this is a good thing in terms of increasing the usefulness of such computer systems, it also allows those with harmful intentions access to the systems. These people will typically search out security faults of various types, in order to exploit the weaknesses for their own purposes.

2.5 Fault Tolerance Summary

Standard failure analysis has yielded tremendous gains in hardware component reliability (disk drives have gone from MTBF of hundreds of hours to MTBF of tens of thousands of hours over the last few decades for instance). Further, the work on high availability systems has yielded a number of highly effective techniques for composing the improved components into larger systems which are even more reliable (one notable innovation being RAID 5).

The approaches used for hardware fault tolerance have not been noticeably successful in providing software fault tolerance. Curiously enough, until recently this has not been a major issue in the design of highly available systems, largely because the software failure rate was sufficiently low that other modes of failure dominated in the overall failure modes. Improvements in hardware reliability, coupled with a trend toward attacks on systems via software defects, are likely to increase the urgency for solutions in this area going forward.

3. HIGH AVAILABILITY

High availability is yet another way of looking at the issue of IT system faults and the key ideas of MTBF and MTTR. In the case of high availability, the focus is on the end user experience. Thus, high availability is normally expressed in terms of per cent availability of the process or function supplied by the associated IT system. Note that availability will be increased by either increasing MTBF or by reducing MTTR

.

3.1 Availability and Target Percentages

While availability is normally expressed in percentage terms, since those percentages are frequently over 90%, it can be useful to convert the percentages into minutes of downtime per year. Thus 99.9% availability is about an hour a year of downtime, while 99.999% availability is about 5 minutes a year of downtime.

While this is easy enough to understand, there is a complication in how availability can be calculated which can impact the downtime numbers above. In computing downtime, some people exclude planned downtime (for things like upgrades or hardware maintenance), and some don't.

The mathematical calculations above are interesting. More interesting is the question of what end users should really expect. In general this is a hard question, which depends on many details of the actual system under consideration, but the following quote seems pertinent.

The availability specification for the Windows NT server is about 99.5 percent. UNIX has something like 99.7 percent availability and OS/390 has 99.999. When you think about it, that means your unplanned downtime on an NT server will average about 50 hours per year. With UNIX, the number is around 26 hours per year. And, with OS/390, you are looking at 5 minutes of unplanned downtime per year. Considering the cost to companies for an hour of unplanned downtime -- $100,000 for a retail operation, $6 million or more for a brokerage – you can see that the drive to client/server is a costly one. [6]

3.2 Improving Availability

Improving availability is not an all or nothing process. It is possible to improve the availability of individual subsystems. This is fortunate, since the cost of applying high availability principles to entire IT systems will increase the cost of the system by a factor of 2 to 4, depending on system details.

The obvious place to start in improving availability is wherever outages are causing the most pain. For each such pain point, alternatives can be developed using the various techniques which have been outlined above. Then the cost of implementing the various alternatives can be determined, and weighed against the business benefit to be gained. As an example, let us assume disk drive failures are causing a significant productivity loss. A few alternatives to increase availability are backups (which can significantly reduce MTTR), raid arrays, various kinds of networked storage (SAN or NAS), and of course using a more reliable brand of disk drive. The disk drive business has grown sufficiently competitive that all the major vendors offer roughly equivalent MTBF ratings, so that alternative is unlikely to bring useful gains in reliability. Backups are likely the cheapest alternative, so if an hours long outage for restoration (and possible loss of any work done since the last backup) is acceptable, this might be a worthwhile alternative. Raid arrays will entail the cost of a new raid controller and additional disk drives for the particular raid mode chosen, but will provide

significantly increased MTBF, and in some cases lower MTTR also. Networked storage will entail acquiring the appropriate storage boxes, as well as some increased administrative overhead. Note further that both networked storage solutions and backup solutions can be shared among multiple systems, so even determining the cost of an alternative is specific to the business making the choice. Given the multitude of technically feasible solutions, business requirements and tradeoffs dominate in determining the exact nature and level of high availability appropriate for any given IT system.

Any business with a large number of similar systems can usually determine fairly easily which components most commonly cause outages. For smaller businesses, or for unusual configurations, it will be necessary to rely on outside data to determine the components most likely to cause problems.

One final point may be useful for any business working on improving availability. Initially it is easy to look at overall statistics on business wide outages and determine which areas need attention. However, once availability improves enough, even a large business is likely to find that the weekly variation in outage causes starts to disguise the underlying trends, and make it hard to gain further improvement. The exact point at which this occurs is largely a function of the size of the business, but in general this sort of problem will start to appear around 99.5% availability. When this starts to happen, it will likely be useful to shift to considering not overall outage statistics, but rather statistics on the longest outages. Normally at this point reducing the length of the longest outages will provide a significant boost to overall availability. [7]

3.3 An Overview of System Failures

As we have seen above, having information on what kinds of system or component failures to expect makes the process of improving availability much easier. This makes published data summarizing all failures, by source of failure, for a large number of systems of considerable value. There are, however, two problems with such data.

The first problem is that there are almost no such studies. The only published work we are aware of that cover all system failures for a large number of systems is the work covering an analysis done by the Japanese Information Processing Development Corporation (JIPDEC) in the mid 1980's. This study surveyed the outages reported by 1,383 businesses in 1985, and covers 7,517 outages with an average outage duration of about 90 minutes. This yields an overall MTBF for the systems of about ten weeks, and an availability of 99.91%. [8]

The second problem is that the data is very dependent on the technologies used in creating the system. While there are specialists who can determine the expected reliability of a system, using detailed calculations on all the various components of the system, this work is quite hard and expensive – normally it is easier to simply observe actual performance. This is particularly true since one of the key components of the system, the software (operating system, middleware, and application) typically does not come with a rigorous MTBF guarantee.

Despite these two caveats, it is interesting to see what has been found, and combine it with the technology trends in the intervening years to provide some guidance on what areas will most likely benefit from improved availability efforts.

In the mid-1980's conventional wisdom was that disks were the major concern for reliability, because they were mechanical. This was borne out by disk MTBF values of about 10,000 hours, although some vendors were producing disks which were both significantly more expensive and more reliable. Although disks have remained mechanical, they now exhibit advertised MTBF values of 50,000 hours, and actual experience indicates MTBF values of between 5 and 10 years. While not all the MTBF news is good, as will be seen in the next section, in general computer hardware has become significantly more reliable, and requires significantly less maintenance than was true during the JIPDEC study.

But is improved hardware and maintenance enough? Note that while the MTBF with 1980's hardware and software was 10 weeks, even with perfect vendor hardware and software the JIPDEC average MTBF would only rise to 4 months. With an average repair time of 90 minutes, this would correspond to an availability of only about 99.95%. Vendor product improvements are clearly important, but also clearly not enough to achieve the high availability goals.

3.4 External Factors

It is useful to consider the impact of WAN connectivity and the power network, which are typically not under the direct control of the business attempting to provide a high availability system. In North America, communication line suppliers will provide service guarantees of 95% error free seconds, bit error rates of no more than 1 error per million bits, and 99.7% availability. Redundant message encoding and duplexing transmissions via independent paths can mask most transient failures, but 99.7% availability is significantly lower than even a PC server. Further, a 1998 study by RTO West, a Canadian power supplier indicated that for their system power was available at the point of delivery only about 98% of the time. Once again, this sort of low external availability can be overcome with careful system design.

3.5 Fault Tolerance Summary

There are five normal sources of outage: 1) Hardware; 2) Software; 3) Operations; 4) Maintenance; and 5) Environment. (Environment includes all items which are not considered part of the system of interest, such as the building(s) the system sits in or the power or machine room cooling systems.)

The base reliability of computer and network hardware has been improving for some time. One recent study suggests that even with PCs, which are normally considered the least reliable server alternative, it is possible to achieve 99.5% availability and an MTBF of about 35,000 hours. While this may change with increasing heat stress in newer PCs, it is still a very promising starting point.

On the other hand, there is no evidence that software reliability is improving anywhere near as fast. Further, virus attacks, which rely on defective software for their effects, seem to be increasing at a rather startling rate. This suggests that for the foreseeable future the limiting factor for availability is likely to be the software.

Operations and Maintenance outages are typically the result of human error and/or vendor design problems in tools or procedures.

Environmental outages include all the sorts of issues that are considered as part of a disaster recovery plan. It is clear from looking at just a few of these items that if they are not taken into account then environmental issues will provide an upper bound on availability significantly below 99%. On the other hand, with careful planning most or all such outages can be controlled.

It is worth noting that high availability is not an all or nothing attribute. By addressing any of the normal sources of system failure, the overall availability will improve. Similarly, by improving the MTTR for any source of failure, availability will improve. For most businesses, the key to improving availability is gathering the data which will cost justify the needed system changes.

4. EXTREME AVAILABILITY

Extreme availability is the ultimate form of high availability, when the desired goal is no outages at all, because the perceived harm to the business is immeasurably large. How does this differ from high availability? To answer that, a comparison is useful. In the United States, both the nuclear power industry and nuclear powered naval vessels were expertly designed with careful attention paid to fault tolerance. Further, in both instances the systems are run by thoroughly trained experts. Nonetheless, the nuclear power industry has had several “incidents”, while no such incidents have been encountered in the nuclear navy.

4.1 Critical Components

In even the most carefully designed fault tolerant system, there will be unexpected events. While such events will be rare with careful planning, proper handling is vital if outages are completely unacceptable.

In addition to the importance of people, most extreme availability organizations pay close attention to a number of other areas in order to achieve their results. These areas include culture, fault tolerance, upgrade procedures, testing, security, using mature software, documentation, and simplicity – each of which is covered in more detail in the following paragraphs.

The first common feature of extreme availability businesses is that they have a strong culture of reliability, which emphasizes such things as always taking extra care and always being ready for a surprise. While culture is always somewhat specific to the organization if appears in, extreme availability cultures seem to share certain traits.

As we have seen in earlier sections, there are a great many possible sources of problems. Extreme availability requires a fanatical attention to possible single points of failure. In power, for example, the building should have two power feeds, coming in on opposite sides of the building, from separate power substations. Each power feed should feed an independent electrical distribution network inside the building. The outside power should be backed up by batteries, which in turn are backed up by generators, with an on site fuel storage sufficient to last many days. Good planning might even get multiple fuel delivery contracts (with independent providers) which would start a week or so before the fuel store would be emptied.

One area where extreme availability organizations pay particular attention to fault tolerance is in the upgrade process for hardware and/or software. For upgrades, extreme availability organizations use a gradual rollout process of some sort, which allows the organization to observe the upgrade in actual operation. Further, this gradual rollout is coupled with a way to back out the upgrade if problems occur. Note that being able to do gradual rollout and immediate back-out (if necessary) may well impose additional requirements on the design for system redundancy.

Redundancy and fault tolerance can protect against random errors, but they provide no protection against systemic errors, such as a programming mistake or misconfigured hardware. The only remedy for such systemic problems is testing. Extreme availability organizations take testing very seriously. Testing

periods are never shortened. Frequently there are multiple independent testing organizations, and each of them has the ability to stop deployment. Testing is normally not under development.

In addition to systemic threats, there are the active threats from people who for whatever reason want to impact the system. Security precautions will include both physical security (site, building and machine room) and network security. When possible, the system should be isolated from the Internet. If Internet access is a key feature of the system, then layered protection against penetration attacks are needed, as well as a mechanism for handling denial of service attacks.

Security precautions are important, of course, but beyond that extreme availability organizations typically implement a higher than normal degree of isolation. For example, development networks may be fully independent of production networks. And of course testing environments are fully independent of both development and production.

Software is markedly different from hardware in the shape of its failure curve over time. New software is markedly more likely to fail than software which has been used by many organizations over several years. Many organizations recognize this effect by avoiding all x.0 releases. Extreme availability organizations extend that rule to try and avoid any commercial software until it has a track record with other organizations.

As we have seen from earlier sections, high availability relies on fault tolerance plus repair. Good monitoring is key to understanding when there is an otherwise transparent failure, and hence when repair is needed. A correct view of what is happening in the system also lowers the likelihood of an erroneous action being taken by an operator.

Extreme availability requires an unusually complete set of operational documentation. Why? People can be single points of failure also. Worse, even if you have several people cross trained you may find that the backup is out sick while the primary is on vacation.

Simplicity is a watchword for most extreme availability organizations. Simplicity means using the fewest pieces possible for the function needed, because of the impact on MTBF of additional components. It also means using simple administrative interfaces, to lessen the chance of error in an emergency.

4.2 Extreme Availability Summary

Extreme availability takes a high availability system as a base, and adds operations personnel with good skills handling emergencies and a strong culture of failure avoidance.

While this may sound simple, it is not. Any culture change is difficult, and the culture of high availability is not common in the industry. This means that it is a difficult adjustment for many.

5. CONTINUOUS AVAILABILITY

As the Internet becomes more significant to business, increasing numbers of firms are looking at the issue of making some or all their systems always available. eBay, for example, attempts to make their systems always available. For convenience, this type of availability is called continuous availability. Given the success of the various approaches above in increasing availability, an obvious question is what impact the additional constraint of continuous availability might have.

At first blush, it would seem that there should be very little impact produced by the continuous constraint. After all, the most highly available systems are already up almost all the time. But in fact there is a substantial impact, and the reason is contained in one of the points touched on earlier, namely the variations in the definition of availability. Most very high availability systems have availability defined in terms of scheduled uptime. Or to put it another way, they do not include scheduled downtime in their availability calculations.

5.1 Implications of No Scheduled Downtime

As we saw in the section on fault tolerance, to get the highest levels of availability, it must be possible to repair a system while the system continues to operate. Given this, it seems it should be possible to achieve the same levels of availability for continuously available systems as for those that have some small degree of scheduled downtime.

The reason this is not in fact the case is that manual repair time is actually quite dangerous from a system availability point of view. The data in this area is quite hard to come by, and is also very fragmentary, but it appears that outages are approximately 30 times more likely during a repair session. This is of course a function of human fallibility. What appears to be happening is that, since repairs are a rare event, people are much more likely to make mistakes at that time than at other times. With scheduled downtime, such mistakes can be caught and fixed without impacting availability.

There is of course a considerable body of data on human errors during emergencies. (A continuous availability outage is naturally an emergency.) It is interesting to speculate on whether well designed repair procedures, which are protected against human error, could in fact allow continuous availability to be a trivial byproduct of high availability. Current hardware and software upgrade and repair procedures do not allow this. As a

result, companies such as eBay, which aim for continuous availability, have an availability target of 99.9%, while firms which aim for extreme availability are likely to have a target of 99.999%.

5.2 Continuous Versus Extreme Availability

Despite the fact that continuous extreme availability appears to be beyond the current state of the art, continuous availability and extreme availability still share a number of similarities.

Both continuous availability and extreme availability take as their base careful attention to high availability principles. Further, both rely on skilled operations personnel working in a culture which emphasizes availability.

The same cultural focus items which are important to extreme availability tend to be important for continuous availability, although there are a few differences which will be detailed below.

While continuously available environments need the same degree of fault tolerance as extreme availability environments, there is an additional requirement. The additional requirement is that repairs must be possible to a running system. This is a design constraint on the system, and it applies to all parts of the system. Notice for example that this can imply a need to be able to replace a power outlet without impacting any other outlet on the same circuit.

Gradual rollouts are important for both extreme availability and continuous availability, but for continuous availability the rollout process (or rollback process) must occur while the system is live. Once again this is a design constraint, but in this case the design constraint may impact such things as the software communication protocols, since a gradual rollout will create situations that could otherwise be avoided.

Security needs are similar in extreme availability and continuous availability, but for continuous availability it will be necessary to have security upgrades rolled out on a live system.

Monitoring is somewhat more complex for continuously available systems, since monitoring will be needed for the repair process also.

5.3 Continuous Availability Summary

At the moment, the very highest levels of availability appear to be unachievable in a continuous availability setting. Careful attention to the high availability precepts, combined with an attentive workforce, can achieve about 99.9% availability, or roughly 9 hours of outage per year.

6. CONCLUSION

While great strides have been made in improving IT hardware availability and in understanding the design principles and operating procedures which will aid in increasing availability, it remains true that high availability is neither easy nor free.

Well designed redundant hardware operated by skilled practitioners can achieve availability rates of 99.999% for key periods each day, or 99.9% on a continuous basis. Organizations which achieve such values spend significantly more on their IT infrastructure than the average commercial enterprise. Unfortunately as we move towards greater degrees of electronic integration between businesses, there will be a need to both close the gap in continuous availability and a need to make extreme availability feasible for businesses that do not have a large team of highly skilled operators.

REFERENCES

[1] Avizienis, A., H. Kopetz, J. Laprie. *Dependable Computing and Fault-Tolerant Systems.* New York: Springer-Verlag, 1987.

[2] Bowen, J. P., M. Hinchey. *High-Integrity System Specification and Design.* London: Springer-Verlag, 1999.

[3] Patterson, D., J. Hennessy. *Computer Architecture: A Quantitative Approach.* San Mateo, California: Morgan Kaufmann, 1990.

[4] Pradhan, D. *Fault Tolerant Computing: Theory and Techniques.* 2 volumes. Englewood Cliffs, New Jersey: Prentice Hall, 1986.

[5] Siewiorek, D., R. Swarz. *Reliable Computer Systems, Design and Evaluation.* 2nd edition. Bedford, Massachusetts: Digital Press, 1992.

[6] Toigo, J. W. *Disaster Recovery Planning:Preparing for the Unthinkable.* 3rd edition. Upper Saddle River, New Jersey: Prentice Hall, 2003.

[7] Marcus, E.; H. Stern. *Blueprints for High Availability.* New York: John Wiley and Sons, 2000.

[8] Gray, J., A. Reuter. *Transaction Processing: Concepts and Techniques.* San Francisco, California: Morgan Kaufmann Publishers, 1993.