SUN BLUEPRINTS
[illegible]
APPROVED

High Availability Fundamentals

By Enrique Vargas - Enterprise Engineering

Sun BluePrints™ OnLine - November 2000

Sun microsystems
ENTERPRISE ENGINEERING

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-6857-10
Revision 01, November 2000

High Availability Fundamentals

We have become highly dependent on computer systems—a good example of this dependence was illustrated by the frenzied preparation for the year 2000 computer bug, Y2K. Apart from the monetary cost, the potential loss of our computer systems caused widespread fears that the world would face various disasters—ranging in severity from a failure of the ATM network to planes falling out of the sky like ripe plums, or even the risk of a nuclear Armageddon.

Availability has long been a critical component for online systems because business processes can quickly come to a halt when a computer system is down. For example, in the world of E-commerce, availability is critical due to a client's demand for instant access to a site—if a site is unavailable, for whatever reason, the competitor's site is only a mouse click away. Any disruption to service is measured, not only in dollars and cents, but perhaps more importantly, in reputation.

This article introduces different system categories to help find a server type best suited to specific business availability needs. This article also covers basic fundamental concepts—basic mathematics is emphasized for the sake of simplicity—of *Reliability*, *Availability*, and *Serviceability* (RAS) relating to the hardware, operating systems, and application elements. This information will assist administrators in making decisions when attempting to balance cost against availability.

Basic System Outage Principles

A system fault can be caused by internal or external factors. Examples of internal factors could include specification and design errors, manufacturing defects, component defects, and component wear out. Examples of external factors could include radiation, electromagnetic interference, operator error, and natural disasters.

Regardless of how well a system is designed, or how reliable the components are, failures cannot be eliminated completely. However, it is possible to manage failures to minimize the impact on a system.

An *error* is the occurrence of a system fault in the form of an incorrect binary output. If an error prevents a system (or subsystem) from performing its intended function, a *failure* has taken place. For example, a defective DRAM device may cause a data bit to deviate from its correct state and produce an error (parity or error correction code - (ECC). If the incorrect data causes a system to crash or reboot then the error becomes a failure.

All system outages fall into two major categories:

- **Unplanned System Outages (failures)**

  Unplanned outages are the result of uncontrollable random system failures associated with *faults* occurring within hardware or software components. Unplanned outages are the most costly, with the highest gains being achieved when steps are taken to avoid this type of outage. Unplanned outages can be minimized through component redundancy, enhanced RAS features, and a Sun™ Cluster environment.

- **Planned System Outages (maintenance)**

  A planned outage should be scheduled to have a minimum availability impact on a system. Planned outages are the result of maintenance events revolving around repair, backup, or upgrade operations. Repairs are intended to remove faulty components and restore a system to a functional state. Backups are intended to preserve critical data on a magnetic storage medium (disk or tape) to avoid the loss of data when a production system experiences a main disk storage failure. Upgrades are implemented to replace the current hardware or software with newer (or enhanced) versions.

System Types—Availability Requirements

High availability is often associated with fault-tolerant systems. The term *fault-tolerant* means a system can operate in the presence of hardware component failures. A single component failure in a fault-tolerant system will not cause a system interruption because the alternate component will take over the task transparently.

As the cost of components continues to drop, and the demand for system availability increases, many *non-fault-tolerant* systems have redundancy built-in at the subsystem level. As a result, many non-fault-tolerant systems can tolerate hardware faults—consequently, the line between a fault-tolerant system and a non-fault-tolerant system becomes increasingly blurred.

It is important to understand the RAS requirements that are spelled out in your application service level agreement (SLA), because different RAS levels will require a different design and configuration approach that can impact cost. For example, in a fault-tolerant highly available environment, any component failure that interrupts system operation may not be acceptable. However, in the general purpose computing environment, it may be desirable to improve overall system availability by choosing a configuration that will have multiple short outages over a configuration that has just one extended outage.

From a traditional availability perspective, computer systems can be grouped into four categories based on their intended applications:

- **General Purpose Computing**

  Applications that fall under general purpose computing cover a wide variety of disciplines—anywhere from engineering analysis to business purposes, or for computational science to data warehousing. Failures in general purpose computers are commonplace and users tend to be more forgiving. Occasional, short disruptions to system operation may be tolerable provided the system restarts automatically.

- **Highly Available Systems**

  Availability is the key metric for applications in these systems. Users demand a high probability of receiving service when requested. Although an occasional loss of a localized service may be tolerable, a system-wide outage is unacceptable. Banking and telecommunications systems are typical examples of high availability environments.

- **Critical Computational Systems**

  These applications have the most stringent fault tolerance requirements. For some real-time control systems, faulty computations can jeopardize human lives or have a high economic impact. The computations must be highly accurate with minimal fault recovery time. Aircraft flight control systems, military systems, and certain types of industrial controllers are some examples of systems that perform critical computations.

- **Long-life Systems**

  In this type of system, reliability is a priority. This includes mobile systems that depart from a central facility for extended periods of time, or systems under little supervision that are placed in remote areas. For these type of applications, unscheduled maintenance can be prohibitively expensive. Space flight and satellites are examples of long-life systems.

All single-node Sun™ servers (except the Netra ft™ series) are typically categorized as general purpose computers. The Netra ft series is considered a highly available system because of its fault-tolerant characteristics.

Note – Sun cluster systems assist in the recovery from both hardware and software failures, whereas a fault-tolerant system has the operating system and application as a single point of failure. Although Sun servers are not specifically designed for critical computation or long-life applications, there are cases where Sun servers have been placed in a remote, *lights-out* datacenter and the availability resembled that of a long-life system.

Reliability Fundamentals

The academic definition of reliability, *R(t)*, is the probability of a system performing its intended function over a given time period, *t*. As an example, a system with a 0.9999 reliability over one year has a 99.99% probability of functioning without failures throughout an entire year.

Reliability is only one of many factors that influences availability, for example, 99.99% reliability does not mean 99.99% availability. Reliability measures the ability of a system to function without interruptions, while availability measures the ability of a system to provide a specified application service level to clients. Therefore, reliability provides a metric of how often a component fails, while availability includes the effects of downtime any failures produce.

Mean Time Between Failures (MTBF)

A commonly used metric for measuring reliability is referred to as *Mean Time Between Failure* (MTBF), which is the average time interval (normally in hours) between two consecutive component failures. Reliability is improved when the time interval spanning separate failures is extended. The following are typical variations of MTBF:

- **Hardware MTBF**

  Refers to the mean time between hardware component failures. Component redundancy and error correction elements introduced by the Solaris™ Operating Environment (Solaris OE) *attempts* to prevent hardware component failures from becoming system interruptions.

- **System MTBF**

Refers to the mean time between system failures. A system failure reflects an application service level outage to users that can only be restored through repairs. Hardware component redundancy increases the overall system MTBF even though the MTBF of individual components remains the same. Because a failure of a redundant part does not bring a system down, the overall system MTBF is increased. However, since individual components fail at the same rate, the higher component count increases the probability of a component failure. Hardware designers have to rely on statistical analysis to quantify the true reliability gain when introducing component redundancy into a system.

- **Mean Time Between Interruptions (MTBI)**

  Refers to a temporary system outage where repairs are not required. The MTBI concept is important in the Sun server world because the Solaris OE includes the *Automatic System Recovery* (ASR) feature—this feature *attempts* to automatically fence off faulty hardware components to render a system functional after a component failure reboot.

Unlike hardware MTBF, system MTBF, and MTBI are measurements of events visible to end users. System failures can result in extended periods of application service level outage, while most system interruptions result in short outages that in many cases are restored without human intervention.

The MTBF value does not indicate exactly when a component failure will occur. The MTBF value is the result of statistical analysis—a failure can occur randomly at any point during a component's life cycle (Murphy's law dictates this will usually be at the worst possible time).

Failure Rate

Another metric for measuring reliability is the *Failure Rate*—defined as the inverse of either the hardware MTBF or system MTBF as displayed in the formula below. If the Failure Rate is high, the MTBF is small.

$$FailureRate = \frac{1}{MTBF}$$

Where
MTBF = Mean Time Between Failures

Each MTBF definition has its own corresponding failure rate. For example, the *Hardware Failure Rate* is the inverse of the hardware MTBF, while the *System Failure Rate* is the inverse of the system MTBF.

Hardware MTBF is probably the most frequently cited MTBF metric even though it does not reflect the typical end-user view of a reliable service level because hardware component failures do not necessarily result in a system interruption.

For most electronic components, the MTBF and failure rate change during the life cycle of a component, however, since the variance is small, it is safe to assume the value is a constant. The component failure rate curve of a component's life cycle is known as a *Bathtub Curve*, see FIGURE 1.

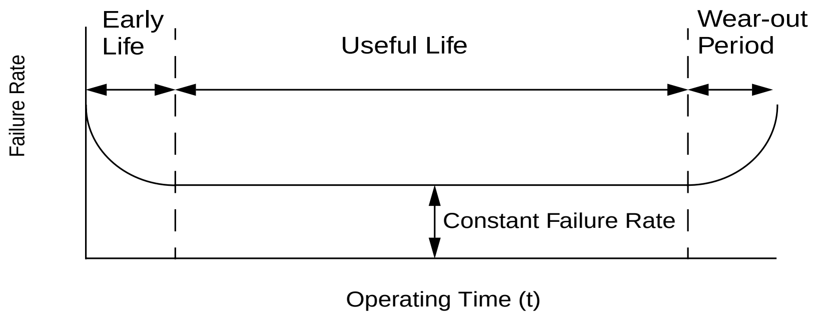

FIGURE 1 Component Failure Rate Bathtub Curve

FIGURE 1 is a plot of hardware component failures over time and it demonstrates a phenomenon known as *infant mortality*—where a component exhibits high failure rate during the early stages of its life. To help ensure component failures are detected early, manufacturers often subject components to a burn-in process (literally, exposure to high temperatures).

Over time the failure rate gradually reduces until it approaches the constant rate—which is maintained during its useful life. Eventually, the component enters the wear-out stage of its life—in this stage, failures increase exponentially. It is important to note that new technologies can exhibit a similar failure rate during the early life and useful life time periods.

Common MTBF Misconceptions

MTBF is often confused with a component's useful life, even though the two concepts are not related in any way. For example, a battery may have a useful life of four hours and have an MTBF of 100,000 hours. These figures indicate that in a population of 100,000 batteries there will be approximately one battery failure every hour during its four-hour life span. On the other hand, a 64-processor server platform may last 50,000 hours before it enters its wear-out period while its MTBF value may only be 2000 hours.

Another common misconception is to assume the MTBF value is higher in a system with strong component redundancy. While component redundancy can increase a system MTBF, it does quite the opposite for the hardware MTBF—generally, the greater the number of components in a system, the lower the system's hardware MTBF will be.

Availability Fundamentals

Steady State Availability is the most commonly quoted availability metric, for most computer systems. Availability is often measured by the *uptime ratio*—which is a close approximation of the steady state availability value and represents the percentage of time a computer system is available throughout its useful life. Another frequently used availability metric is *downtime per year* which is calculated using the following formula:

$$\text{Downtime Per Year (minutes)} = (1 - \text{Uptime Ratio}) \times 365 \times 24 \times 60$$

A system is often classified by the number of 9s displayed in its availability figure. TABLE 1 illustrates the different availability classes—based on the number of 9s, and their corresponding annual downtime calculated using the formula above. As an example, a system that has a five-nine availability rating means that the system is 99.999% available with a downtime of 5.26 minutes a year.

TABLE 1 Availability Classes and Downtime per Year

Availability%	Downtime per Year
99	3.65 days
99.9	8.76 hours
99.99	52.6 minutes
99.999	5.26 minutes
99.9999	30.00 seconds

Availability is commonly defined through the following formula:

$$Availability = \frac{MTBF}{MTBF + MTTR}$$

MTBF = Mean Time Between Failures
MTTR = Mean Time to Repair

The availability formula does not include enhanced RAS features typical of the Solaris OE. This is demonstrated in FIGURE 2 and assumes a system is either available, or experiencing a repair cycle.

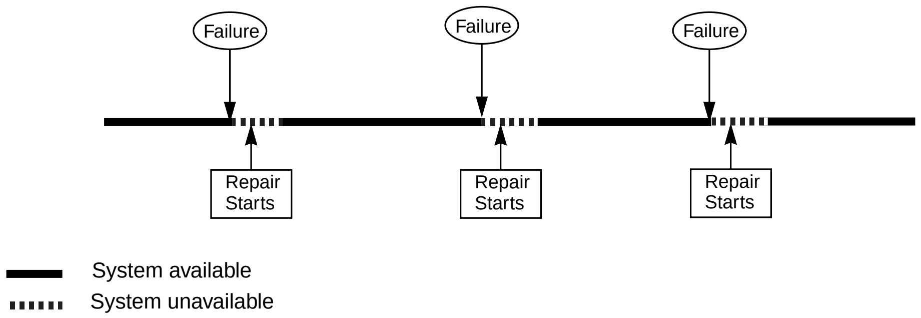

FIGURE 2 Solaris Operating Environment Availability Model Without Enhanced RAS Features

The Solaris OE contains enhanced RAS features including *Online Maintenance* and *Automatic System Recovery* (ASR). Because some of the failure cycles do not require any service intervention, it invalidates the old computer systems availability model. The ASR feature allows faulty components to be automatically fenced off after a component-failure reboot (component-failure reboots are invoked by the kernel or hardware to prevent data corruption). Hardware component redundancy is critical for system availability because ASR will not grant a system reboot if the remainder of the components cannot guarantee a functioning system.

Note – To enable the ASR feature, the OpenBoot™ Prom (OBP) `diag-level` environment variable must be set to `max`. The `diag-trigger` OBP environment variable must be set to `error-reset` or `soft-reset` to ensure that diagnostics are executed after a reboot.

FIGURE 3 illustrates possible failure scenarios for a Sun server:

- Failure 1 requires service intervention to bring the system online
- Failure 2 is detected by the ASR feature which brings the system back online after a reboot. The faulty component will be replaced during maintenance.

  As an example of the Solaris OE availability model, a CPU failure on a Sun Enterprise™ 6500 server can be characterized as Failure 2 in FIGURE 3. This failure induces an outage in the form of a component-failure reboot. The second outage occurs when the system board containing the bad CPU module is replaced during maintenance and the system is rebooted.

- Failure 3 does not bring the system down because the faulty component is bypassed using component redundancy. The faulty component will be replaced during maintenance.

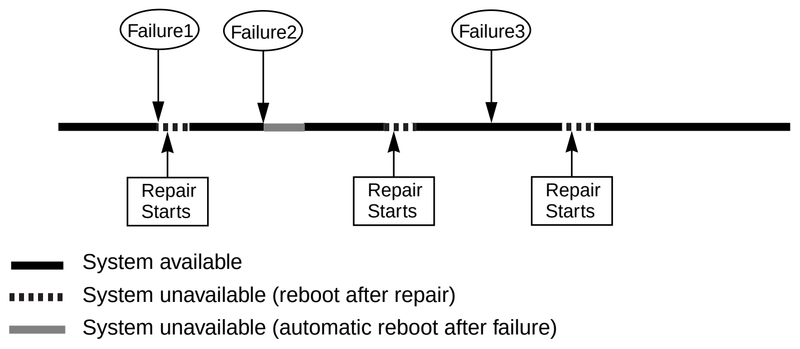
Failure1
Failure2
Failure3
Repair Starts
Repair Starts
Repair Starts
System available
System unavailable (reboot after repair)
System unavailable (automatic reboot after failure)

FIGURE 3 Solaris Operating Environment Availability Model With Enhanced RAS Features

The Solaris OE availability features (such as, component redundancy, ASR, and Online Maintenance) have made outage scenarios so complex that MTBF and MTTR figures can no longer be generated using basic statistical formulas—a random component failure simulation model is required.

Reliability versus Availability

There is a difference between reliability and availability. Depending on a customer's situation, benefits may favor reliability over availability or vice versa. For instance, reliability must have a higher precedence in cases where systems are placed in remote areas and maintenance is expensive. For additional information, see Chapter 2 in *Sun™ Cluster 2.2 Infrastructure* to assess the reliability of different hardware component choices for your datacenter needs.

Although availability is a function of reliability and serviceability, it is possible for a system with poor reliability to achieve high availability. For example, assume that a system averages four failures a year, and assume that for each failure, the system can be restored with an average outage time of one minute. Using the availability formula presented in section "Availability Fundamentals" on page 7, the availability of this system is calculated to be 99.99924%—four minutes of downtime out of 525,600 minutes in a year.

Reliability measures the ability of a system to function continuously without interruptions. A high MTBF for a computer system indicates that on average, the system can operate for extensive periods of time without component failure interruptions. On the other hand, availability, measures the ability of a system to support a committed application service level.

The following formula is an approximate conversion of reliability to availability:

$$Availability = \frac{t - (1 - R(t))dt}{t}$$

Where
R(t) = System Reliability over a time period *t*
dt = Average outage in hours

NOTE: The formula above is an approximation of a more complex statistical formula and is likely to be inaccurate for an *R(t)* figure below 90%. This formula was derived by the RAS Engineering group.

The reliability, *R(t)*, of a system represents the probability of the system not encountering any failures over a time period, *t*. The value of *1 - R(t)* represents the probability of a system encountering failures over a time period, *t*. Multiplying *1 - R(t)* by *dt*, (the average outage time), approximates the total outage time over the same time period (assuming *R(t)* is close to one).

The value of *t - (1 - R(t)) x dt* yields the total time a system is in working condition during the time period *t* (uptime). Dividing *t - (1 - R(t)) x dt* over the total time period *t* yields the total availability figure.

Serviceability Fundamentals

Serviceability, *S(t)*—is the probability of a service action being completed during a given time period, *t*. If a system is said to have a serviceability of 0.95 for three hours, then the probability of a service action being completed within three hours is 95%. Higher reliability reduces the frequency of system failures and higher serviceability reduces the duration of each repair incident while increasing overall availability.

A commonly used serviceability metric is the Mean Time To Repair (MTTR), which is the average length of time needed to complete a repair action.

Customers do not normally allow for the impact of the logistics time in the MTTR calculation. The logistics time is sometimes referred to as service response time, and it is the time needed for a service engineer —with the necessary spare parts—to arrive at a customer’s site. Availability calculated without the inclusion of a logistics time figure is called *intrinsic availability*.

Note – The logistics time is often influenced by a service option. We encourage customers to explore the Sun Enterprise Service offerings to select a service option that covers their availability requirements.

The following sections describe two software technologies developed by Sun Microsystems to simplify system serviceability, and enhance availability.

Dynamic Reconfiguration

The Solaris OE supports the Dynamic Reconfiguration (DR) software on Sun Enterprise™ 3000 to 6500, and 10000 platforms. The DR software enables the attachment or detachment of hardware resources without incurring system down time. For details on DR capabilities, see the Sun BluePrints™ book *Resource Management*. DR also enables online maintenance—which removes the need for a system reboot when defective hardware components are replaced.

Even though DR increases availability, the Sun™ Cluster version 2.2 (SC 2.2) infrastructure does not currently support it because of the time involved in suspending the operating system during a DR operation. The elapsed time spent in the suspended state by the operating system is dependent on system resources available during a DR-detach operation. A busy system can stay in the suspended state for a long enough time to trigger a non-responsive node alarm in the cluster management software.

Alternate Pathing

The Solaris OE supports Alternate Pathing (AP) software on the Sun Enterprise 3000 to 6500, and 10000 platforms. The AP software enables a manual fail over of an active network and an automatic fail over of disk I/O controllers to standby resources after a failure is detected. In the future, AP will provide full automatic fail over capabilities.

The SC 2.2 infrastructure does not allow the use of AP since it conflicts with its own bundled Network Adapter Fail Over (NAFO) software. NAFO provides automatic fail over to alternate network controllers when a network failure is detected.

Single System Availability Tuning and Configuration

This section discusses the key factors involved in configuring a single system to match your availability requirements. The most important thing before configuring a system for high availability is to understand the application RAS requirements and to select an appropriate computing platform (see "System Types—Availability Requirements" on page 2). Availability tuning is similar to performance tuning and is essentially a process whereby areas that contribute the most to system outages are identified and minimized —or eliminated—to improve overall system availability.

As previously mentioned, datacenter processes required to increase the high availability level are constrained by different factors:

- static

  System configuration, software choices, etc.
- dynamic

  Training, documentation, fire drills, etc.

The following sections address system configuration issues that can help increase system availability.

Configuring to Reduce System Failures

Sun Microsystems servers have been designed with enhanced system availability in mind. This has been accomplished through the reduction of component count, strict quality control, environmental monitoring, and the removal of active components from critical Field Replaceable Units (FRU). Memory error correction code has been adopted on all servers to minimize system downtime caused by faulty Single Inline Memory Modules (SIMMs) and Dual Inline Memory Modules (DIMMs).

Sun Microsystems servers also allow hot swapping of some components (for example, power supplies and cooling fans). Adding redundancy to subsystem components, such as power and cooling, provides an immediate availability improvement if faulty components can be hot-swapped without interrupting the operating environment.

Production system failures can be caused by a single point of failure (SPOF) or by multiple failures occurring within the same downtime period. Because multiple fault occurrences are rare, SPOFs are the dominant cause of system failures and should be targeted for elimination. SPOFs are unacceptable in a Solaris cluster environment since the basic premise of a cluster environment is total component redundancy at all levels of hardware and software.

Eliminating SPOFs can be accomplished by adding redundant components to the system. However, having more components in the system increases the probability of a component failure and can potentially result in additional system interruptions.

Component placement is critical for eliminating SPOFs. For example, if disk controller B provides redundancy for disk controller A, both controllers need to be placed on separate system boards to prevent inaccessibility after a system board failure.

Configuring to Reduce System Interruptions

A significant availability improvement can be achieved by reducing system interruptions caused by components such as CPU modules, memory modules, etc. Here are some basic approaches for increasing availability at the component level.

- Reducing the number of system interruptions:

  Minimize the number of system interruptions by reducing the number of components whose failure could cause system interruptions. A common approach is to replace a number of low-speed CPU modules with fewer high-speed modules. This provides equal processing power with a lower probability of a CPU module failure. At the same time, replace low-density DIMMs/SIMMs with fewer high-density DIMMs/SIMMs. This provides equal memory capacity with a lower probability of a memory module failure.

- Reducing the duration of system interruptions:

  There are several ways to reduce a system downtime window to help increase a system's availability (see next month's "High Availability Best Practices" article for additional details). File system logging—featured by Solaris 7 and VERITAS File System software—reduces system downtime by eliminating file system checks during a reboot.

Configuring to Reduce the Need for Maintenance Reboots

Most hot-pluggable FRUs in enterprise servers today can become hot-swappable with an appropriate operating system. Reduce the number of maintenance reboots by configuring the system for Dynamic Reconfiguration (see "Dynamic Reconfiguration" on page 11). DR software requires a lot of front end planning—

especially if the system boards to be added or removed involve I/O controllers. Although DR software may increase a system's availability level, the SC 2.2 infrastructure does not currently support it.

Configuring Highly Available Subsystems

A computer system (from a RAS perspective) is viewed as consisting of multiple subsystems in series, which means that if a subsystem fails, the whole system fails. For example, if the AC/DC power supply subsystem fails, the whole system will fail. However, computer systems based on RAS principles have redundancy built into each subsystem. For example, a RAS computer system with N+1 redundancy will have more than one power supply module in the subsystem (see FIGURE 4) therefore, the failure of one power supply module will not result in a total system failure.

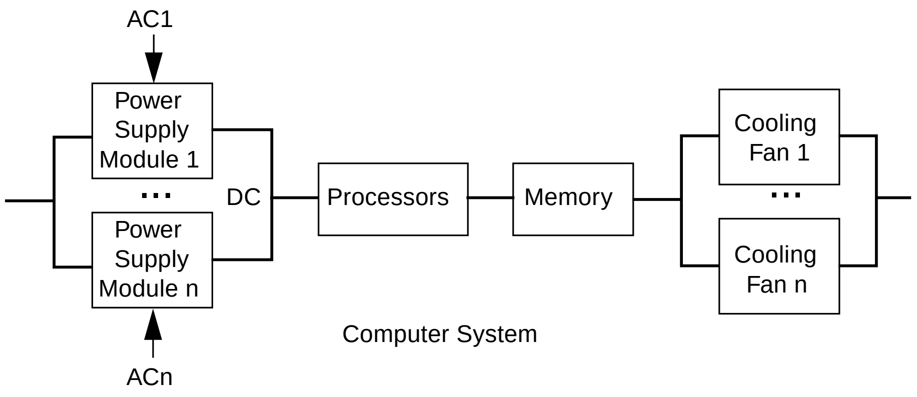

FIGURE 4 Serve Subsystems with N+1 Redundancy

The availability of the overall system is only optimized after the individual subsystems are optimized. The following sections highlight RAS-related issues for the main subsystems.

CPU Subsystem

Due to its complexity and high density, the processor module is a component that generally has one of the highest failure rates in a system. In most Sun servers, a failed processor will cause the system to invoke a component-failure reboot. However, if the ASR feature is enabled, the failed component will be fenced off after rebooting. There are few configuration choices for processor modules, however, the general rule of reliability applies—the fewer components there are, the less the chance of a failure.

Memory Subsystem

Memory errors can be single-bit or multiple-bit, however, the majority of errors are single-bit. A computer system is unaffected by single-bit errors, however, multiple-bit errors will cause the system to invoke a component-failure reboot. The general rule of reliability also applies here—the number of memory modules should be reduced by using high capacity memory modules.

Input-Output (I/O) Subsystem

There are a large number of I/O controllers available, consequently, methods of dealing with a failure can vary greatly. Some controllers support redundancy using AP software (see "Alternate Pathing" on page 11), load sharing with Sun Trunking™ software, or using the SC2.2 NAFO software.

Note – Neither AP software nor Sun Trunking software are currently supported by the SC2.2 infrastructure.

Boot Disk Subsystem

Without a boot drive it would be difficult to bring up the production version of the operating environment. The boot drive can be made highly available by mirroring its contents using a hardware or software volume manager.

Software volume managers such as the Sun Enterprise Volume Manager™ or Solstice DiskSuite™ (SDS) can be used to mirror the boot drive. Additionally, the use of hot-pluggable disk drives, may reduce the need to bring the system down to replace a failed half mirror.

Power Supply Subsystem

On most Sun servers, a power supply failure will not bring a system down, if it can be hot-swapped transparently. Configuring a power supply is relatively simple, however, keeping up the required N+1 redundancy is often overlooked in a fast-paced environment.

Cooling Fans Subsystem

On most Sun servers, a cooling fan failure will not bring a system down if it can be hot-swapped transparently. Most cooling related issues are external to the system—such as keeping the computer room temperature stable below 25 C(75 F)—high temperatures and temperature fluctuations are a form of stress to electronic components.

Disk Storage Arrays Subsystem

Storage array components are almost as complex as computer systems because they have their own power supplies, cooling fans, host-interface cards, processors, and memory. By mirroring the array or setting up Redundant Array of Independent Disks (RAID-5) volumes, it is possible to experience a storage array failure without bringing the system down. If hot-pluggable disks are supported by the disk array, failed disks can be replaced transparently.

Cluster Node Subsystem

In a SC2.2 configuration, we can view each of the computer nodes as a subsystem. A node failure in a correctly configured cluster will only cause a brief interruption. An interruption caused by this type of failure will usually be shorter than a failure incurred in a non-clustered system since the interruption only involves the application fail over time with no reboot time.

Summary

Datacenter availability is a critical business element which enables Sun customers to survive relentless competition and to satisfy stringent global business demands. This article's intent is to bring everyone on the same terminology playing field by focusing on basic mathematical concepts to define *reliability, availability,* and *serviceability.*

This article mainly emphasized configuration elements that impact a single server's availability to help system administrators arrive at a hardware configuration that best matches availability requirements.

We highly recommend the reader to review next month's "High Availability Best Practices" article to help increase their knowledge about availability in the datacenter.

Author’s Bio: Enrique Vargas

Enrique Vargas brings a wealth of large systems experience to Sun Microsystems and specializes in high-end UNIX® platform offerings, including the Sun Enterprise 10000. Enrique Vargas came to Sun Microsystems from Amdahl where he also focused on the high-end Solaris systems.