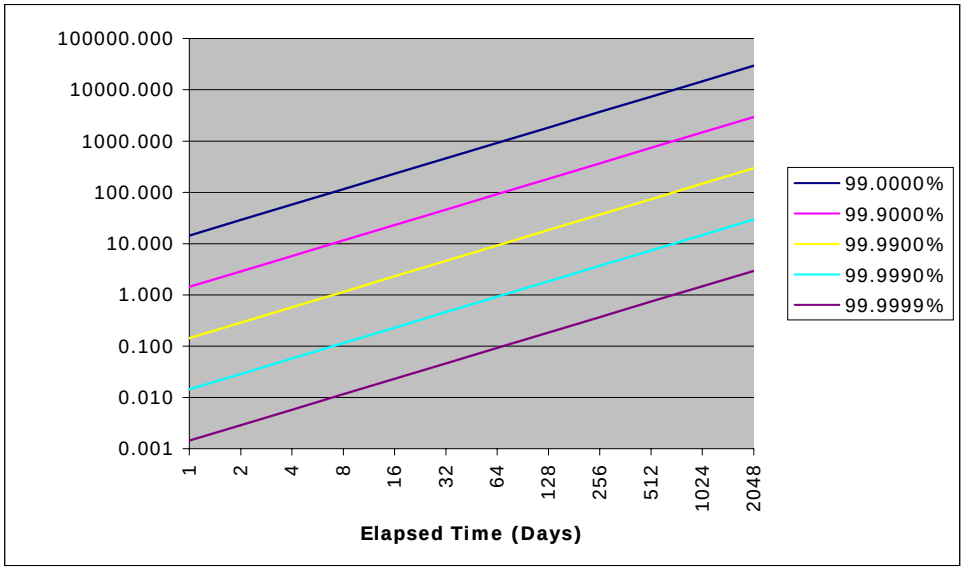
SKY COMPUTERS
THE MULTICOMPUTING EXPERTS

A SKY Computers White Paper

High Application Availability

By: Steve Paavola, SKY Computers, Inc.



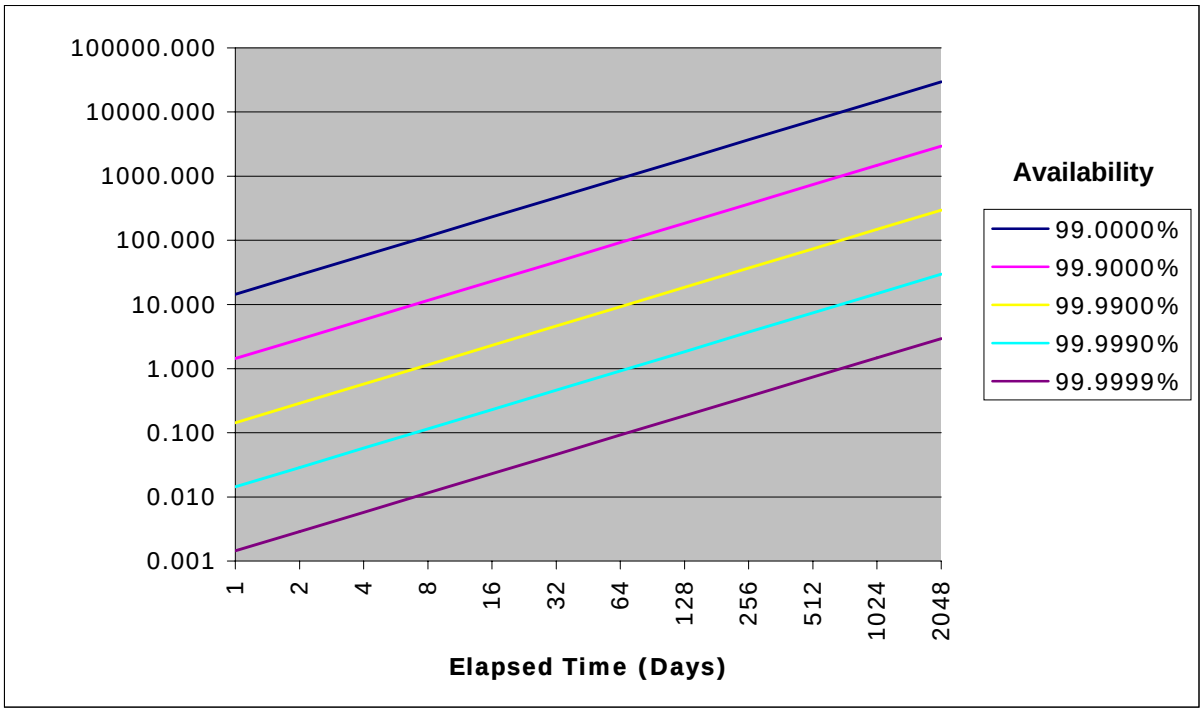
SKY Computers, Inc.
8 Centennial Drive
Peabody, MA 01960
978 977 3000
www.skycomputers.com

There is increasing emphasis on providing High Application Availability (HAA) for mission critical applications. The challenge of finding a cost effective solution increases as system complexity increases. HPEC systems implemented using RACEway or SKYchannel have limited capabilities to support HAA. However, the emergence of new fabrics such as InfiniBand is enabling truly HAA applications.

The usual metric for application availability is

$$\text{Availability} = \frac{\text{Mean Time Between Failures (MTBF)}}{\text{Mean Time Between Failures (MTBF)} + \text{Mean Time To Repair (MTTR)}}$$

The goal is to maximize this value, with a common requirement of 0.99999 or 99.999% (5 9's). This translates to about 5 minutes per year that the application can be unavailable, for whatever reason. From this formula, it is obvious that both system reliability (MTBF) and fault recovery



(MTTR) must be addressed in order to have a system solution that can be considered HAA.

Reliability

Certainly, there are the traditional hardware features that can be implemented to improve system reliability. Proper hardware design can improve hardware reliability. ECC on memory can correct many common failures. CRC on fabric transfers can detect errors, allowing the transfer to be re-tried. Modern systems also have sensors to monitor power supplies, fans and temperatures so that

these relatively low reliability components can be replaced before they cause a system failure. SNMP on IP networks and “baseboard management” on InfiniBand are used to report and manage these resources. But, reliability must be more than this. Features of the interconnect in a multi-processor environment can provide a safer application environment.

Using message passing to transfer data between processors can provide a much safer environment than using shared memory. A shared memory interconnect usually exposes the entire memory of the target system to accidental modification, either through application errors or a hardware fault. Operating system software can minimize the risk from application faults, but cannot control the hardware faults. Message passing hardware enables the target processor to restrict the memory visible to the source.

Additional reliability features available on some interconnects are keys put into the message header. A key can be established for each end-to-end connection such that if the key doesn’t match when a packet is received at the destination connection point, the packet is discarded. This enables much better security, and prevents unintentional access to a connection point. Keys can also be used to partition the fabric. For example, a partition can be established for communications processors, and another partition established for applications processors. Security processors can be established that are members of both partitions. As a result, the communications processors are unable to send packets directly to the applications processors. Their communication path is only via the security processors, which verify the validity of the transactions before forwarding them to the applications.

Fault Detection and Recovery

Even after the effort has been put into system and application reliability, faults will still happen. If the goal is “5 nines”, the application can’t wait for human intervention to fix the fault. Remember, the allowed application down time is only 5 minutes per year! Also note that a fault can be caused by a hardware failure, software failure, or human error. In actual implementations, the hardware is often the LEAST likely source of failures.

In order to support HAA, a system-wide approach must be taken. For each resource in the system, an analysis must be made to determine its impact on the system should it fail. In an HPEC (High Performance Embedded Computer) system, an n+m approach is normally taken for each resource. In this approach, if n copies of the resource are required to support the application, m additional copies are made available to provide redundancy. This applies to processors,

memory, fabric connections, power supplies, fans, name servers, and any other resource that might fail. If it's possible for the power cord to vibrate loose, it makes sense to provide two.

Once the system hardware has been configured to support HAA, the work still needs to be done to deal with faults. Because there is so little time available to recover from the fault the process must be automated, requiring support from both hardware and software. It's useful to define 5 steps involved in Fault Management:

- Detection – determine that a fault exists
- Diagnosis – identify the failing component
- Isolation – protect the rest of the system from the failing component
- Recovery – get the application running again
- Repair – replace the faulty component

There are a number of techniques used to detect a fault. These include a variety of hardware checks, including ECC or CRC checks, state changes on communications links, watchdog timers, missing communications responses, and application specific fault detection. The fault must be reported to a resource that is still reliable, and which can manage the fault.

The fault is then diagnosed to determine which resources are no longer usable at the moment. In the n+m design, if removing the unusable resources will leave enough to continue the application, then this will be done rather than attempting a repair at this point. A real repair won't be attempted until after the application is operational again. Often, the affected resources are self-evident from the nature of the failure. For example, if a sub-cluster failed to respond to a work assignment, the hardware and software resources in this sub-cluster are no longer usable. Sometimes a diagnostic will have to be run to localize the failure enough for the next stage.

Once the failure is localized, the unusable resources must be isolated from the application. In the case of a sub-cluster that failed to respond to a work request, the sub-cluster is simply removed from consideration for future work scheduling. InfiniBand provides significant additional capabilities. For example, a failed link would cause all communications through that link to fail. An HAA system would be designed with alternate links available. Taking advantage of these alternate links simply involves changing the routing tables in the fabric – a software task which can be completed in a fraction of a second. InfiniBand also supports automatic failover. When an application establishes a connection to another node in the fabric, it can request that a redundant path be established as well. During operation of the application, should a packet retry count be exceeded, the hardware will automatically fail over to the alternate path.

Once the system is re-configured, application recovery is possible. For some failures, the application may not notice that the failure occurred, with only an error report going to a management node. This would be the case in automatic path failover, or when a fan fails with the remaining fans picking up the load. Without automatic failover, a failed link would cause errors to be reported the application images. In an HAA application, these images could wait for the management node to reconfigure the fabric, then continue operation when notified by the management node. In other applications, restarting may involve recovering from a previously saved checkpoint. The recovery strategy is very much application dependent, and may depend on the resources that were lost.

At this point the application is running, but with less than the full set of resources that it started with. An HAA application usually requires that the failed resources be repaired and returned to service, without taking the application down. In the case of a non-responsive sub-cluster, the procedure might involve re-booting the nodes and re-loading the application into those nodes. The assumption would be made that there wasn't a hard failure on all of the nodes, so each node would be tested for correct operation. After all, the failure was probably induced by a software fault, not a hardware fault! Once correct operation is verified, they would be made available for scheduling. If a hard failure does exist, the good resources are still returned to service. An InfiniBand based system would then allow physical replacement of the failed component because InfiniBand does support live insertion.

Conclusion

Implementing HAA requires a system level view of the solution. There isn't any simple solution – hardware or software – that will make an HAA solution. System reliability is certainly important, but careful analysis for potential faults and the 5-step fault management process will focus planning for the inevitable.