High Application Availability

Steve Paavola
SKY Computers, Inc.

Abstract

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. The usual metric for application availability is

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

The goal is to maximize this value, with a common requirement in telephone applications of 0.99999 or 99.999%. 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. 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. A partition can be established for communications processors, and another partition established for applications processors, for example. 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 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.

There are usually 5 steps involved in Fault Management:

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

There are a number of techniques used to detect a fault. These range from 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 task of diagnosis is only to locate where the fault occurred so the application can continue operation. A real repair won't be attempted until after the application is operational again. Usually this is self-evident from the nature of the failure as detected. Sometimes a diagnostic will have to be run to localize the failure enough for the next stage.

Once the failure is localized, the failing component must be isolated from the application. To maintain HAA, this must be done using software to re-configure the communications fabric, the system software and the application. Some communication fabrics support automatic failover. If the hardware determines that a communications path is no longer functional, the communications interface will start using a pre-determined alternate path. Physically changing hardware will take too long.

Once the system is re-configured, the application recovery is possible. The application restarts, recovering any data that was saved. It will continue operation in a potentially degraded mode.

Repair usually requires human intervention. Many HAA systems are designed to support live insertion/extraction. Diagnostics may be run on the failed component while still physically still installed in the system to determine if the fault requires replacement of the hardware. If the fault was software induced, the problem may be intermittent and the component can be returned to service immediately.

Conclusion

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