

# High-availability Computing in an Open Systems World

*High-Availability Computing and Open Systems have been concepts in conflict for years. The vast majority of applications for Open Systems do not require "5-nines" availability (systems available 99.999% of the time, implying no more than 5 minutes of down time per year). Thus, the design points for industry standard open systems emphasise other priorities-notably cost-over non-stop availability. Many telecom and internet applications, meanwhile, demand high-availability. This has led to the creation of numerous proprietary computing solutions for telecommunications.*

## INTRODUCTION

The difference between industry standard open systems and high-availability telecommunications systems can be seen just by looking at them. While typical PC systems consist of peripheral cards which plug into a horizontally mounted motherboard, typical telecom systems consist of vertically mounted cards plugged into passive backplanes. The PC solution is the lower cost design strategy, but high-availability demands systems where individual cards can be removed and replaced quickly, ideally without even stopping the rest of the system.

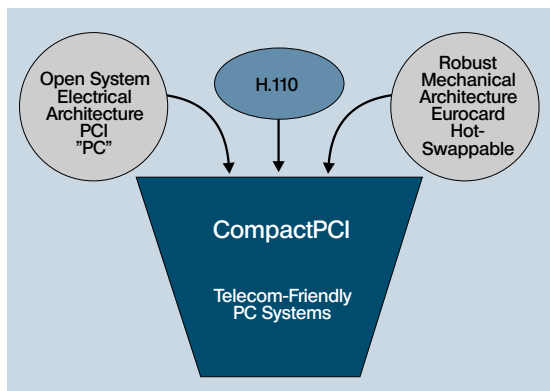


Figure 1.

CompactPCI has been designed to bridge this gap. It blends the mechanical characteristics needed for high-availability systems with the open architecture of PCI, bringing with it full PC-compatibility. However, to reach the availability requirements of telecom and internet applications, more than just a PC in a better mechanical package is required. Recognising this, the leading CompactPCI equipment providers, through the PCI Industrial Computers Manufacturing Group (PICMG®), are extending CompactPCI standards in several ways with the goal of specifying a standard high-availability computing platform.

In the first phase of CompactPCI standards development, the emphasis was on bringing together existing standards. On the electrical side, the key was use of the PCI bus. Not only is this a popular and highly func-

tional bus structure, it brings with it the capability of building completely standard PC-compatible systems.

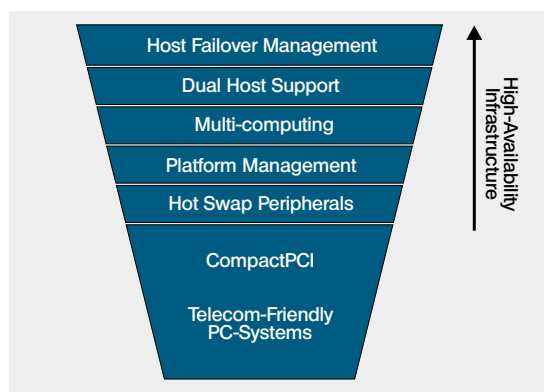


Figure 2.

On the mechanical side, the basis for CompactPCI was the Eurocard standard, widely used in VME systems. CompactPCI included a number of advances to the VME mechanical standard, most notably the higher-density connectors, and standardised transition I/O cards.

Add to this mix the H.110 bus standard-itself an adaptation of the existing H.100 standard used in traditional PC systems-and the result is a CompactPCI platform which is very telecom-friendly.

In Phase-2 of CompactPCI standards development, more sophisticated capabilities are being defined specifically aimed at building an infrastructure for high-availability. The first piece of this puzzle was the definition of a Hot-Swap capability for CompactPCI peripherals. The next piece will be the adoption of a standard platform management capability, the definition of which is well underway. Additionally, work on standardisation of multi-computing and dual-host capabilities are now ongoing.

Crowning phase-2 of CompactPCI will be the emergence of systems which implement host failover management. This is the ability to continue system operations after the failure of the system host. There are several approaches to that problem, and different techniques are suitable for different applications.

This paper will provide a survey of high-availability

technologies being developed for CompactPCI: where they now stand, and what they can accomplish, especially when combined together.

In developing these technologies, the challenge faced by platform vendors is in providing additional system availability while preserving the value of "openness" in the system. The primary payoff of having an open system lies in the transparent portability of application software. When software can be moved effortlessly from one platform to another then it is possible to significantly drive down the cost of both the hardware and software in the system. Conversely, when hardware and software are tied together, as in a proprietary system, the lack of economies of scale and ongoing meaningful competition keep prices high.

While the goal of effortless software portability is partially met just by having standards, the biggest benefit of CompactPCI systems will be realised by keeping them fully compatible with standard PC systems. The ideal is to allow application developers to write software without regard for the platform on which it will be running-i.e., having the same application run on the lowest priced desktop PC, on a super-high-performing and high-availability CompactPCI system, or any platform between these extremes. Thus, when building high-availability into CompactPCI standards, it is critically important to do it in a way that doesn't involve application software at all, and as little operating system software as possible.

## COMPACTPCI HOT-SWAP

The first layer of the CompactPCI high-availability infrastructure is the Hot-Swap standard. The mechanical structure of CompactPCI is a prerequisite and key enabler of the ability to Hot-Swap cards. The vertical orientation, front-panel access, card guides and so on make it possible to safely remove and insert cards while power is applied to adjacent cards and the backplane. The existence of rear-I/O transition cards also helps since it allows removal and replacement of

cards without touching the tangle of cables that may be connected to the system.

To take advantage of this capability though, sophisticated capabilities at the system level are required and are provided by the CompactPCI Hot-Swap standard.

The Hot-Swap standard defines three levels of capability which a peripheral card and system may provide:

- Basic hot-swap is the easiest to implement and provides only an environment where cards can be safely inserted and removed without causing physical damage or affecting other things going on in a running system. It doesn't provide any capabilities to help the system identify a newly inserted card and get it properly configured however.
- Full hot-swap adds that capability and defines an interface with the OS which allows for system software to help co-ordinate the insertion and removal of cards, so that the software can make sense of the changing configuration of the hardware.
- The third level of hot-swap support, called high-availability hot-swap, adds a few enhancements over full hot-swap functionality for use in high-availability systems.

Specifically, high-availability hot-swap gives the system control over cards that might be misbehaving, allowing them to be individually reset or powered off. It also includes a healthy signal coming from each card which can be monitored by a centralised Hot-Swap controller. Adding support to a peripheral card for high-availability hot-swap is trivial, but there are some significant issues with adding it to a CompactPCI system. The biggest issue may be that it is not defined in the standard how the host processor controls these features-for example, how the host processor orders the power-off of a particular card. This leads to the development of proprietary solutions in this space which is contrary to the goal of making CompactPCI systems compatible with each other and with standard PC systems.

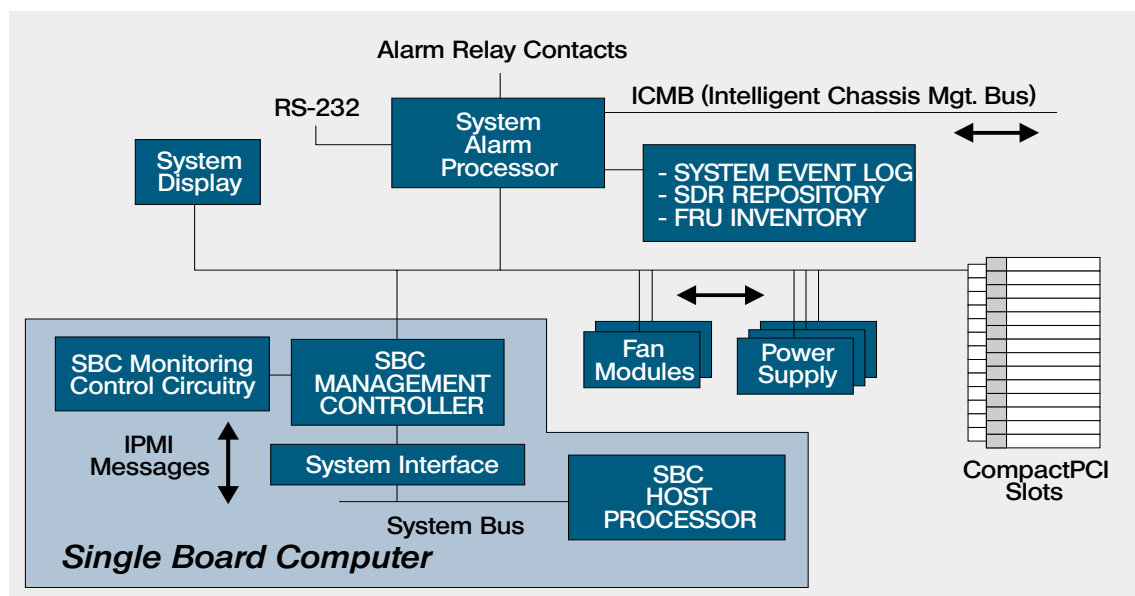


Figure 3.

***AD APPLIED  
COMPUTING  
CONFERENCE***

The capabilities provided by the high-availability hot-swap standard could be provided via a standardised platform management infrastructure-which is now being defined-and if done through that mechanism, the high-availability part of the CompactPCI Hot Swap specification may actually become less important as a standardised approach integrated with platform management provides the same functions.

## COMPACTPCI PLATFORM MANAGEMENT

The CompactPCI Platform Management subcommittee has adopted a standard named "IPMI" or the "Intelligent Platform Management Interface" for creation of a platform management infrastructure for CompactPCI systems. This standard is relatively new, and was originally developed by Intel, Dell, NEC, and Hewlett Packard for use in enterprise server systems.

One of the main problems that the developers of the IPMI standard were trying to solve was the lack of portability of management software across different platforms. In a typical server, there may exist lots of management software-most of it very standard and portable. However, at the bottom layer of the management software stack, there must be a platform-specific interface to platform-specific capabilities. Thus, management software must be customised for a particular platform. There have been attempts to define what all platform management features may exist, then standardise each one of them, but this is a difficult and potentially counter-productive activity in an area where innovation is continuing rapidly. IPMI solves the problem by defining a standard interface to whatever management capabilities are present in a platform. It does this by requiring platforms to include a Baseboard Management Controller which interacts with the actual platform hardware, and which, in turn, presents a standard, abstracted interface to software running on the host. Thus, whatever is unique to a particular hard-

ware platform is encapsulated in the Baseboard Management Controller-which is itself part of the platform-and out of the realm of management software running on the system.

Importantly, for CompactPCI, the IPMI standard also defines a management bus for communicating management data throughout a system. This management bus connects various management processors within the platform and provides a means for adding management capability to platform components-power supplies, peripheral cards, alarm processors, etc.-which can then be incorporated into platforms made by various manufacturers. The figure below shows a block diagram of the IPMI management subsystem in the RadiSys CP80 CompactPCI platform.

IPMI fits into a CompactPCI system by having a baseboard management controller embedded on the host SBC-so here it is called an SBC Management Controller. This controller provides the IPMI standard interface to whatever management software may be running on the host processor. In addition to providing access to management data specific to that SBC, the SBC Management Controller also provides access to the IPMI management bus, called the IPMB. This is a serial, multi-master bus based on an I2C physical layer, with IPMI defined addressing and message formatting. The IPMB is routed to all CompactPCI peripheral slots so any peripheral card can be managed.

Often, in CompactPCI systems for telecoms, there is an independent alarm processor in the system. With an IPMI architecture, this alarm processor is placed on the IPMI Management Bus to gain access to all the management data and control available in the system. By putting this capability onto an alarm processor which is included with the platform, no impact on any software running on the system is required to get the benefit of the capabilities provided through the presence of this management infrastructure.

The IPMB can also be used to interface with other

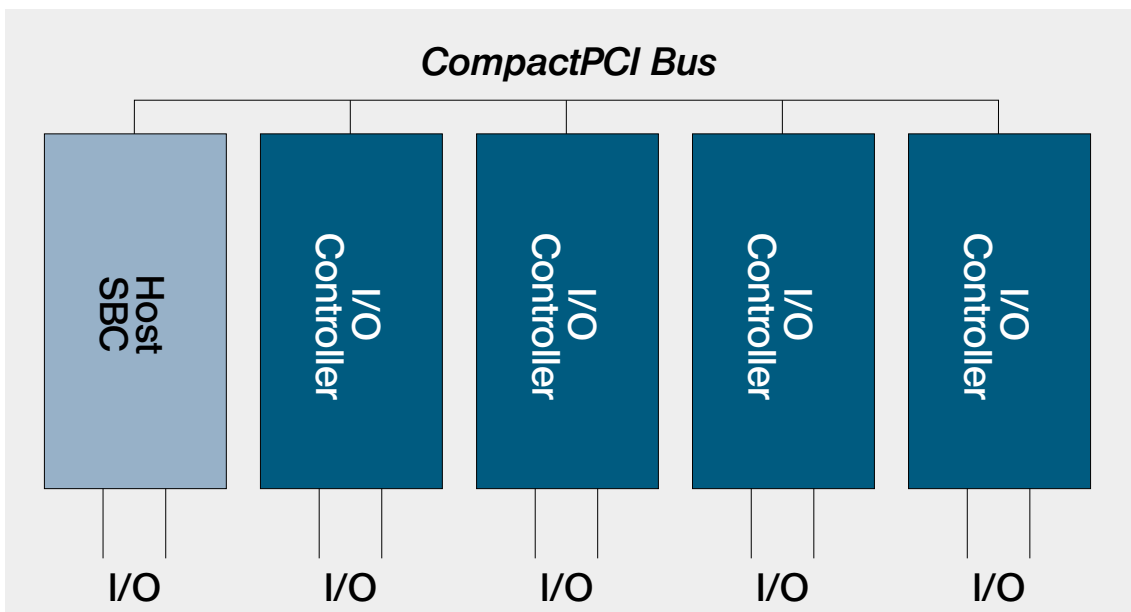


Figure 4.

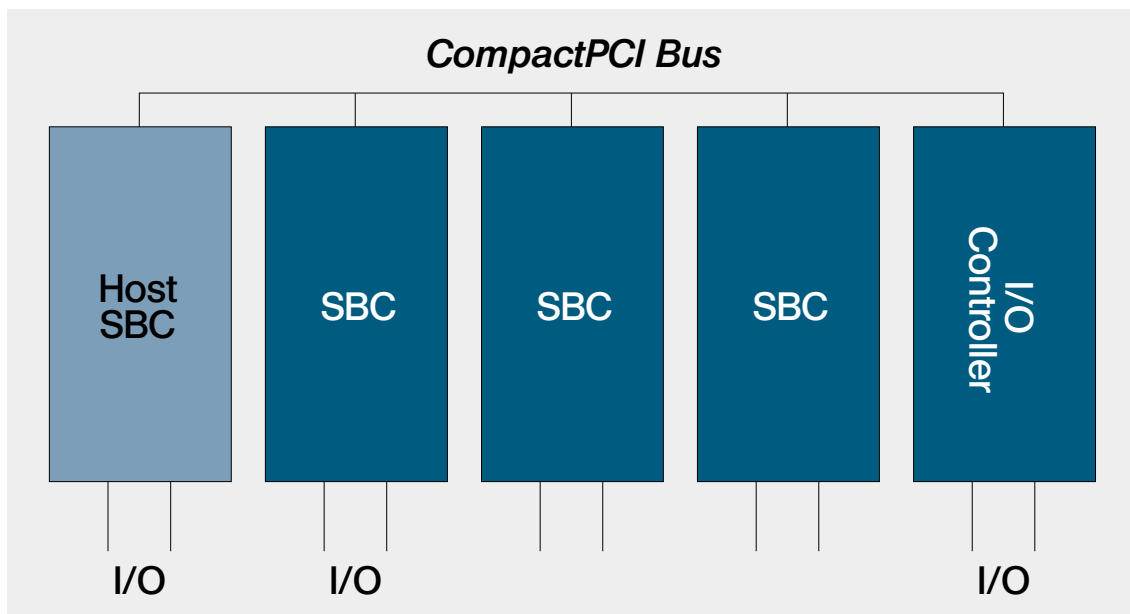


Figure 5.

devices in the system chassis such as power supplies, fans, or status displays. Finally, the PICMG subcommittee on platform management is considering designs with dual-host processors by defining a dedicated IPMB for use in inter-connecting two system slot boards.

To summarise the key features of CompactPCI Platform management:

- It is based on standards from the mainline PC server world.
- It complements the hot-swap standard both by being capable of having devices on the IPMB hot-swapped and by being able to use the IPMB to coordinate hot-swap activity, including the functionality of high-availability Hot-Swap systems.
- It can support independent management processors, so that host software does not have to be affected.
- It includes support useful in dual-host systems and multi-computing systems by providing an out-of-band communication channel which can be used to co-ordinate the activity of spare processors which may not yet even be configured on the main PCI bus.

### COMPACTPCI MULTI COMPUTING

Another advance in CompactPCI systems is the coming standardisation of multi-computing, or loosely-coupled-multiprocessing systems. Where a typical computer system has a host processor that is fully in charge of everything going on, with other cards in the system being only slave I/O controllers, a multi-computing platform will have several single-board-computer modules, each of which may (or may not) have I/O devices and, each of which, operates somewhat independently. This, of course, has been done for years in VME systems, so it isn't anything new except to PCI-based systems or to standard PC-compatible systems.

CompactPCI multi-computing systems still need to have a primary, host SBC in a special "system slot" on the backplane. In fact, using PCI bridge technology available today, the SBCs that are in slots other than the system slot, need to have a different type of connection to the PCI bus. This is provided by what has come to be known as a non-transparent PCI/PCI bridge. This device, such as the Intel 21554, makes a non-system-slot SBC appear to be a peripheral card to other devices on the CompactPCI bus. Whatever I/O devices are on the non-system-slot SBC itself are private to that local processor, and the only interface into the SBC is via shared memory, doorbell interrupts, or I2O message queues. Logically, this makes the non-system-slot SBC appear to be an intelligent peripheral device. Often, this is exactly what the SBC is-controlling some sort of specialised I/O either embedded on the board, or on a PMC module. In other applications, the non-system-slot SBC can be a generic "compute resource," receiving input over the CompactPCI bus, processing, then delivering output back over the bus.

Multi-computing relates to high-availability because it enables the creation of systems with distributed processing power configured in an N+1 manner. If any one of the non-system-slot SBCs fail, the work can be redistributed to surviving processors. The non-system-slot SBCs should also be hot-swappable and managed via IPMI, so that diagnosis, repair, and reconfiguration can take place while the system continues to run.

PICMG is involved with standardising inter-processor communications in CompactPCI multi-computing systems by defining a shared-memory message passing and interrupt generation protocol which will allow messages to be exchanged among the multiple SBCs in a system. This protocol will allow integration with OS networking support, so that the links "look" like a high-speed Ethernet connection even though they are actually implemented by passing data over the

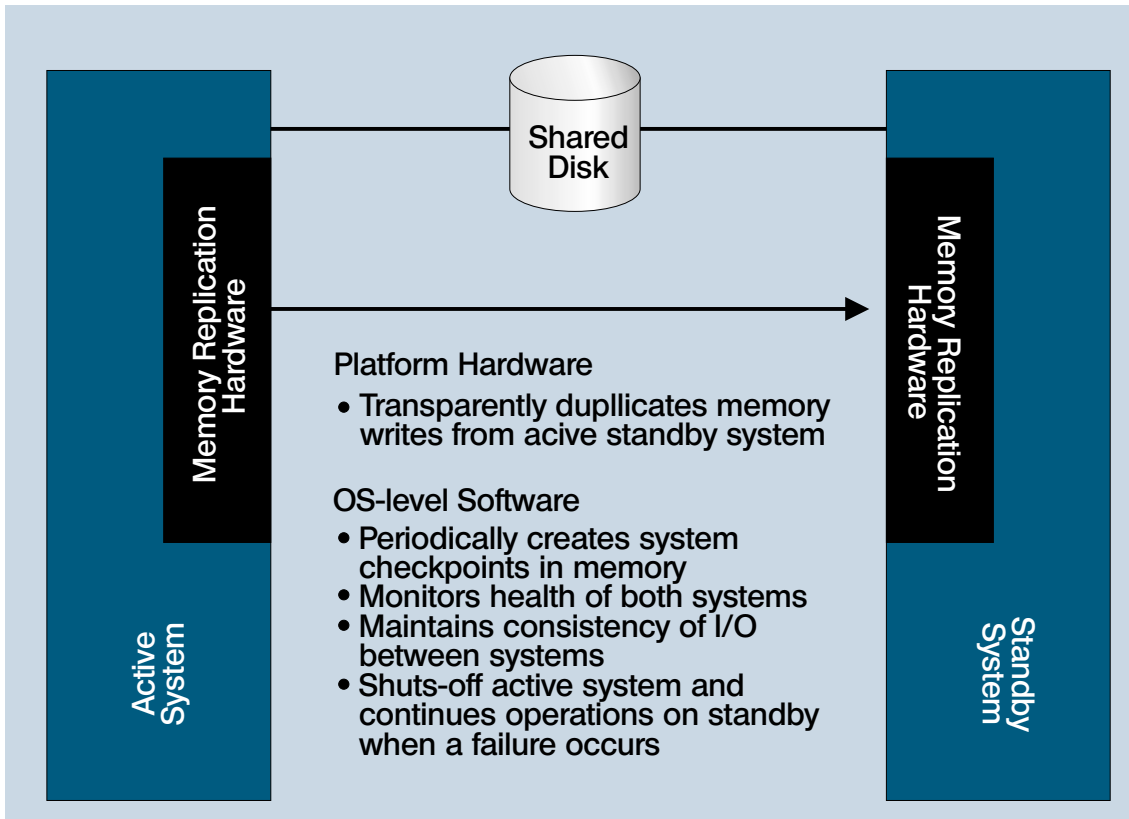


Figure 6.

CompactPCI bus. One of the by-products of defining interprocessor communication this way is that it will be possible to configure a multi-computing CompactPCI system to operate just like a cluster of standard PCs connected via a LAN. This can promote application portability by being able to use applications designed for standard clustered systems in high-performance and high-availability CompactPCI servers.

Note that use of the PICMG standard backplane communication protocol is not required to build a CompactPCI multi-computing system. At least two alternatives exist:

- Several real-time operating systems offer inter-processor communications capabilities today.
- Solutions are being designed today which incorporate actual Ethernet interconnects between SBCs.

The second of these alternatives is particularly interesting. With a private, switched Ethernet interconnect, bandwidth can scale linearly as more SBCs are added, ultimately eclipsing even the PCI backplane bandwidth. Emergence of gigabit Ethernet makes this even more attractive as an option. By putting two Ethernet ports on each of the SBCs, very high-availability, and very high capacity systems can be built.

In the extreme, a multi-computing system based on a switched Ethernet interconnect may not have a PCI bus running between SBCs at all -although it may still use all the rest of the mechanical standards of CompactPCI. In such a system, IPMI platform management becomes quite attractive as it provides an important mechanism to "bind" the loosely linked

SBCs together into a single managed entity.

## COMPACTPCI DUAL HOST SYSTEMS

Returning to more traditional CompactPCI systems which do include a PCI bus structure, building highly available systems also means addressing failures of the host SBC, that is, the SBC in the CompactPCI system slot. Because the system slot is special in CompactPCI-and PCI in general, something special has to be done to deal with the failure of host SBCs. This is the problem being addressed by the PICMG dual-host subcommittee.

Hot-swapping a failed host CPU is more demanding than Hot-Swapping a peripheral card, because the system cannot operate at all while a host SBC is removed. Thus, CompactPCI systems need to support switching host SBC duties from a failing board to another, already installed board. Making this transfer involves moving the clock source for the CompactPCI bus, and somehow handling I/Os which were in process between the failed host SBC and other peripherals when the failure occurred.

Currently, the PICMG dual-host subcommittee is working on standards which will define enough so that inter-operation between SBCs, backplanes, and peripheral cards can be maintained in dual-host environments. The subcommittee is likely to work primarily on making the bus and bridges behave correctly through a failover event. How the standby computer, when it takes over, knows the current context and state of the failed computer is likely to remain beyond the scope of this committee. There are many ways to do

this, and no one method is going to fit all applications. This is the top of the CompactPCI high-availability infrastructure. Host Failover Management refers to the problem of transferring state and context from a failed host processor to a standby. There are basically four ways to get a standby processor up to date in order for it to takeover operations. The first two of these methods, Lockstep Processing and Application Directed Checkpointing are the "traditional" methods which have been used in various systems for years. The second two, Clustering and System Directed Checkpointing are newer approaches and hold out promise for making the failover transparent to application software which is key in an open-system world.

### LOCKSTEP PROCESSING

Many approaches to lockstep processing have been implemented through the years. In lockstep systems, the standby processor executes right along with the active system so that at the moment of a failover it is already in step with the system that failed and can simply continue running. This can be implemented with hardware where multiple CPUs actually execute in true lockstep running the same instruction stream at the same time. This requires a significant amount of proprietary hardware though and is getting harder to do with advanced microprocessors which include deep pipelining, internal caches, and so on.

Something like lockstep processing can also be done by software by running the same application on two systems with both seeing the same input stream, but with only one of them actually creating output. With this method, when a failover occurs, the application software on the standby system is ready to continue operations. The main downside of this approach is that it requires an enormous amount of application software development, and unless the active/standby environment can be replicated with simple stand-alone PCs, it means breaking the open-system model by tying the application development to the platform.

### APPLICATION DIRECTED CHECKPOINTING & CLUSTERING

Application directed checkpointing is the most common traditional method for host processor failover. The

application running on the active system periodically sends information to the standby system in order to keep the standby system up-to-date. When a failure occurs, the standby system uses the latest information it has received to reset and then continue the application. This also requires a significant application development, but generally can be done with completely standard platforms, so at least it does not create a proprietary solution. The biggest problem for many telephony applications though, is that the fail over times, as the standby system uses the checkpointed data to bring itself up to date, may be long. The emphasis in the design of most of these systems is to minimise an outage rather than to keep the system running without any loss of service ■

---

*David McKinley is Director of Technology for the Computer Platforms Division of the RadiSys Corporation. In this position, he is responsible for development of specific high-availability system technologies, as well as other technology initiatives for RadiSys telecommunications and embedded computer systems. McKinley has over 25 years experience in developing computing systems and a long history of working with high availability and fault-tolerant systems. Prior to joining RadiSys in 1997, he was with Tandem Computers where he was responsible for managing communications and systems management software development for the Tandem Integrity family of fault-tolerant computer systems. McKinley possesses technical expertise in data communications, networking, systems management, and high-availability technology. He has been responsible for the architecture and design of high-throughput message switching systems, intelligent communication controllers for fault-tolerant computer systems, systems management software, and CompactPCI telecommunications systems. A native Texan, McKinley received a BS degree in Computing Science from Texas A&M University, and an MBA from the University of Texas.*



Let us know about your events to be included in Dedicated Systems Magazine  
and the on-line Dedicated Systems Encyclopaedia!  
Contact Nico Van Wijmeersch at +32-2-520.55.77 or [info@dedicated-systems.com](mailto:info@dedicated-systems.com)