



Intel[®] Technology Journal

Communications Processing

**AdvancedTCA*/CGL Advantage:
HA and TCO**

AdvancedTCA*/CGL Advantage: HA and TCO

Sharad Garg, Intel Communications Group, Intel Corporation
Raj Sistla, Intel Communications Group, Intel Corporation
Ramesh Caushik, Software and Solutions Group, Intel Corporation
Julie Fleischer, Software and Solutions Group, Intel Corporation
Rusty Lynch, Software and Solutions Group, Intel Corporation

Index words: Advanced Telecom Computing Architecture (AdvancedTCA), high availability (HA), Total Cost of Ownership (TCO), carrier grade, Linux, Carrier Grade Linux (CGL), Chassis Management Module (CMM), Telecom Equipment Manufacturer (TEM), Telco, five 9s

ABSTRACT

Carrier-grade systems in today's telecom market have high availability needs; however, current market conditions are forcing telecom equipment providers to look for lower-cost alternatives. In this paper, we discuss Advanced Telecom Computing Architecture (AdvancedTCA*), Carrier-Grade Linux* (CGL), and how these standards meet the requirements of carrier-grade equipment. We also describe how both of these standards can help increase high availability (HA) and reduce the total cost of ownership (TCO) in carrier-grade systems.

We define AdvancedTCA and its relationship to HA and reduced TCO. We also describe AdvancedTCA shelf management and show how AdvancedTCA hardware can enable HA middleware to perform HA functions. When discussing CGL, we focus on CGL requirements related to AdvancedTCA that increase HA and reduce TCO, as well as those that reduce TCO in general.

INTRODUCTION

In the current telecommunication climate there is a strong need to introduce increasingly complex voice and data services and at the same time reduce capital and operational expenditure. Next-generation telecommunication applications and carrier-grade equipment that provide these services have stringent availability (99.999% or greater, sometimes referred to as the five 9s), serviceability, and scalability requirements. In addition they need to have a low Total Cost of

Ownership (TCO). Carrier-grade systems were until now built around proprietary architectures that led to unacceptably high TCO for these custom-designed solutions. Alternative architectures that are based on open, standards-based communication infrastructure platforms and hardware and software building blocks help provide the needed high availability (HA) and performance at a reduced TCO. AdvancedTCA is a specification that defines such an open modular architecture. It defines a new board and chassis form factor, and an interconnect fabric and management schema that is tailored to meet the HA needs of next-generation converged applications. AdvancedTCA-based hardware requires management software, a carrier-grade operating system and HA middleware to provide the total solution. AdvancedTCA, a standard proposed by the PCI Industrial Computers Manufacturers Group (PICMG), defines carrier-grade hardware in an open, standards-based way. The AdvancedTCA standard also enables centralized, scalable, out-of-band management in a high-density modular chassis by incorporating the concept of a shelf manager that acts as the central management authority in the system. In this paper, we discuss the Intel® NetStructure™ AdvancedTCA Chassis Management Module, an implementation of this concept.

The CGL Working Group of the Open Source Development Labs (OSDL) has defined a series of Linux requirements [9] that enhance Linux [10] for carrier-grade systems.

* Other brands and names are the property of their respective owners.

® Intel NetStructure is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

In this paper, we discuss AdvancedTCA, how it combines with carrier-grade Linux operating systems (OS), HA middleware, and the management software to deliver on the promise of HA with reduced TCO. In addition, this paper describes how CGL requirements enable the HA needed for carrier-grade systems as well as how the requirements enable an open, standards-based solution that reduces TCO.

ADVANCEDTCA

Overview

The Advanced Telecom Computing Architecture (AdvancedTCA) is a specification targeted at the next generation of telecommunications equipment. It provides the high-availability (HA) requirements needed for carrier-grade equipment, and its open, standards-based approach enables a lower Total Cost of Ownership (TCO) for Telecom Equipment Manufacturers (TEMs).

AdvancedTCA Advantages

The AdvancedTCA specification defines carrier-grade communications equipment that will be highly available and can have a lower TCO than proprietary solutions. The key hardware-related advantages of the AdvancedTCA-based solutions are listed below.

Standards Based

The advent of rugged, standards-based architectures resulting from the AdvancedTCA standard is expected to steer the market away from proprietary designs. These architectures allow TEMs to choose the best components for the system and can help decrease cost as well as time to market.

Adaptable Interconnections

The AdvancedTCA interconnect framework is comprised of the physical connector used to mate boards and backplanes, the mapping of interconnections to that connector, and the routing of those signals among boards across the backplane. The AdvancedTCA supports five different communication interconnects that can be used for control, data, and management traffic. AdvancedTCA does not mandate any specific interconnect technology (e.g., 10Gbe, StarFabric, InfiniBand, PCI Express, etc.) and is capable of supporting all of them due to the way in which the specification deals with just differential pairs and not the protocols/technologies. This provides a great deal of flexibility and protection for the future and reduces the TCO.

Hot Swap Ability

In AdvancedTCA, almost all the entities are required to be hot swappable. This ensures operation on the order of 99.999% since a hot swap of one component would not interfere with the payload operation of another component.

Built-in Manageability Support

Downtimes of less than five minutes a year is the norm for these mission-critical systems for telecom applications. As such it is imperative that these systems be constantly monitored and managed to anticipate and prevent failures. It is also important that the system management mechanisms do not interfere with the operation of the payload: i.e., out-of-band management is essential.

AdvancedTCA devotes a great deal of attention to system management. About 30% of the specification discusses the technology and the requirements for achieving system management in AdvancedTCA systems. AdvancedTCA uses the Intelligent Platform Management Interface (IPMI) v1.5-based protocol for management and extends it in multiple ways to achieve all the requirements. The physical medium for management transport across the backplane is a two-wire serial bus capable of carrying Intelligent Platform Management Bus (IPMB) messages. AdvancedTCA mandates that every entity that connects to a management bus shall implement a management Intelligent Platform Management (IPM) controller to communicate via the IPMI. Also, to achieve HA, the specification mandates that the management bus be paired up with a redundant IPMB. These requirements go a long way to ensure a standards-based management methodology, something that was not proposed by earlier standards and something that increases Mean Time Before Failure (MTBF) and reduces costs in the process.

SHELF MANAGEMENT IN ADVANCEDTCA

The importance assigned to shelf management increased greatly between Compact PCI and AdvancedTCA. The AdvancedTCA specification mandated the implementation of a shelf manager in the shelf (chassis) as a central management authority in the shelf. The purpose of shelf management is to monitor, control, and ensure proper operation of boards and other components by maintaining a health counter that reports anomalies and takes corrective actions when required. The shelf manager is also capable of retrieving inventory information from the shelf components.

In AdvancedTCA there are multiple components of the shelf management system.

- There are individual management controllers on each Field Replaceable Unit (FRU) in the system that monitor the system's operation and health.
- The shelf manager is the central management authority in the shelf and communicates to the other controllers via the Intelligent Platform Management Interface (IPMI).
- There is a higher-level external system manager that is in charge of managing multiple shelves within the Telco office.

The shelf manager in AdvancedTCA performs the following major functions for the shelf components:

- Power management
- Hot-swap management
- Cooling management
- Electronic Key (E-key) management
- Inventory management

AdvancedTCA introduces the notion of "Intelligent FRUs." An intelligent FRU is one that contains an IPM controller that implements the IPMI protocol and that is capable of interfacing to the Intelligent Platform Management Bus (IPMB). Each intelligent FRU must negotiate power-usage needs with the shelf manager before it can be powered on. The shelf manager manages the power and cooling resources in the shelf at all times to ensure that the headroom is not exceeded. Also, boards must report their backplane interconnection types before any of them are enabled. This last functionality, known as E-keying, replaces the legacy model of using mechanical keys on the backplane to prevent incompatible interconnections.

The Intel Chassis Management Module

Since shelf management is such an important piece of AdvancedTCA, a robust solution to shelf management is key to a full AdvancedTCA solution. The Intel NetStructure Chassis Management Module (CMM) is one such solution for AdvancedTCA shelf management. It provides centralized, out-of-band management in a high-density modular chassis that is compliant with PCI Industrial Computers Manufacturers Group (PICMG) standards. It implements a radial IPMB topology whereby each intelligent FRU is connected to the shelf manager over dedicated, redundant IPMB links and has eventing/alarming capabilities for up to 16 node and/or fabric slots as well as for system power entry modules and fan trays. This ensures greater throughput, less chance of error, and greater security. In addition, the

active CMM may be paired with a backup CMM for redundant use in high-availability (HA) applications.

The CMM is capable of controlling, sequencing, and managing power to the FRUs in the chassis and maintaining a power budget. It manages the interconnections between the computing boards in the chassis ensuring that only boards using similar protocols are allowed to communicate over the backplane. The CMM maintains an inventory of the entire chassis to allow for easy field replacement of a module. The CMM employs different mechanisms to monitor and alarm, based on health changing events in the chassis. The adaptable designs of both the hardware and the software of the CMM enable Original Equipment Manufacturers (OEMs) and Telecom Equipment Manufacturers (TEMs) to build application-specific, standards-based modular platforms and extensions to the management software stack using the Intel CMM.

Given the critical nature of the applications using these modular platforms, it is unacceptable to have a single point of failure anywhere in the chassis. Since the CMM is the central management authority in the system, it is vital that there be a redundant CMM for HA failover.

The goals of redundancy between the CMM are twofold:

- (1) Ensuring a seamless failover to the redundant CMM in cases of emergencies thereby preserving the notion of a single management authority in the system.
- (2) Synchronization of the management information between the CMMs to maintain accuracy in cases of failover. The two CMMs operate in an active-standby model. The following components of the information are kept in sync between the CMMs:

- IPMB state information
- System event logs
- Health information
- CMM configuration information
- User names and passwords
- Power and hot-swap information
- Cooling parameters
- E-keying information

The physical media used for achieving the synchronization are the IPMB and the Ethernet. Two redundant and dedicated IPMBs connect the two CMMs. Each CMM has two network ports that can be configured

to provide an Ethernet connection between the two CMMs.

The active CMM will failover to the standby when it senses that it is not capable of acting as the shelf manager and the standby is better able to do so. When a failover occurs, the standby takes over seamlessly to ensure active shelf management. This redundant operation increases the HA of the entire shelf.

HIGH-AVAILABILITY MIDDLEWARE AND ADVANCEDTCA

Configuration and cluster management middleware is the key controlling entity in a high-availability (HA) configuration and is implemented by HA middleware. [Figure 1](#) shows the software stack and HA middleware that run on an AdvancedTCA blade. It maintains a system model of the components that comprise the cluster, defines how faults are detected in the cluster, and what action should be taken as a result. When failures occur it takes the appropriate actions to reconfigure the cluster and notify and/or restart affected parts of the application. The HA cluster configuration and management must be able to operate in a heterogeneous cluster. The member systems in the cluster may be of different types with different operating environments, and may have HA management middleware from different suppliers. The system model, the message protocols between cluster members, and application APIs must allow proper heterogeneous interoperability.

In this section, we discuss HA middleware requirements and show how HA middleware achieves application failover, including the fault management cycle. Then, we describe the hardware features of an AdvancedTCA platform that can assist HA middleware to achieve fast failover.

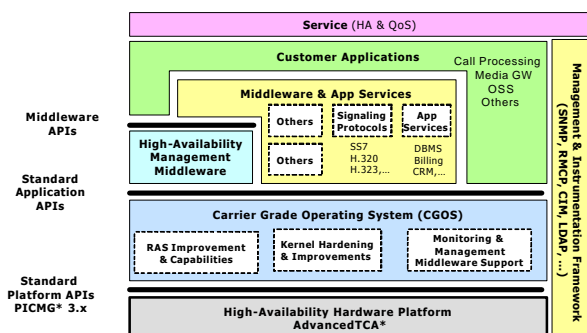


Figure 1: Software stack running on an AdvancedTCA blade

Telco HA Middleware Requirements

An application failover solution in a carrier-grade, HA environment must meet the following criteria:

- (1) *Efficient and fast.* The management software must be optimized for speed and efficiency. Given the stringent performance and availability requirements of telecom equipment providers, it is important that the management software itself carries a low overhead [1], [2].
- (2) *Self-managing.* The entire HA management system must be self-managing and self-reliant, and the failover cycle must operate automatically in real-time without need of human intervention. Self-management decisions are usually based on policies.

The management system must monitor and collect data about all critical system resources, including hardware, software, the operating system, and applications. Availability management software therefore must include an interface to these resources. System faults must be quickly detected, and fault data must be provided to an availability management service so that the service can quickly and appropriately respond by initiating actions that reconfigure the status and functioning of the resources as needed to maintain service.

- (3) *Configuration and maintenance of system-wide state model.* Service availability management requires a system-wide model that can represent all managed resources in the system, changing information and the intricacies of each resource's dependencies and interdependencies. The management software needs this information in order to make quick and appropriate reconfigurations when necessary.
- (4) *Automatic data check-pointing.* Redundancy is the key to HA. To provide continuously available services, the current application data and state must be maintained in a hot standby redundant resource. The HA management middleware must continuously copy ongoing transaction and application state data to a hot standby resource for a fast failover.
- (5) *Fault detection and diagnosis.* The HA management middleware must detect and analyze errors, and it must initiate an appropriate recovery action including switchover to a standby resource and send a message.
- (6) *Fast policy-based recovery.* The corrective action taken by the HA management middleware should be based on an application-specific policy. These actions can cover a wide range of activities from restart of a failed application, to failover, to a

standby hardware card. The recovery process is often multi-step in that several actions must be taken in a prescribed order. In some cases, the recovery process is multi-tiered in that, if a particular action does not recover the system, some alternative action must be taken.

- (7) *Dynamic management of system component and dependencies.* Cluster-aware availability management is responsible for initiating the actions and orchestrating the role assignments that maintain continuous availability. Availability management also controls service groups (the collections of managed objects in redundancy relationships such as 2N or N+1) and makes role assignments (active, standby, spare) within them. Availability management is responsible for the ongoing verification of component health via heartbeating protocols, for overseeing the check-pointing of data to redundant resources, for executing the switchover to redundant resources, and for dynamically reconfiguring the system.
- (8) *Administrative access and control:* Service availability middleware must provide a set of services for the ongoing maintenance and management of the system.

Fault Management Cycle

As just described in a previous section, fault detection and diagnosis, and a fast recovery are key requirements for a fast failover solution. The fault management cycle encompasses these steps, starting with the detection of a fault and ending with the recovery of the system [8]. A complete fault management cycle is described below, and the tasks a HA middleware solution should perform to ensure HA and a fast failover are also described.

Detection

The HA middleware will identify an undesirable condition by checking the health of every entity on a blade such as applications, the OS running on the blade, and hardware entities such as network port or memory faults, etc. The Intelligent Platform Management (IPM) device on each blade monitors several hardware/physical entities such as temperature and processor on that blade. In the case of hardware faults, the IPM device logs the events and faults in its local system event log (SEL) and informs the shelf manager. There can also be watchdog timers that might be monitoring some events, and in cases of time out, will inform the IPM device about such events. The HA middleware can access Intelligent Platform Management Controller (IPMC) event logs such as rising temperature, or a failed component to find out

about such events, thus avoiding going through layers of software.

Diagnosis

The HA middleware analyzes faults to determine their nature and location. It also performs root-cause analysis to identify the originating fault.

Isolation

The HA middleware contains the problem to ensure that a fault does not cause a system failure.

Recovery

To recover the system, the HA middleware does the following:

- Restores system to expected behavior.
- Performs policy-based actions such as
 - restarting a failed application or
 - switching over to a standby node.
- The shelf manager and the IPM device can assist by, for example, rebooting or restarting a node.

Repair

The HA middleware repairs the system by doing the following:

- Restoring a system to full capability including all redundancy.
- Replacing hardware or software components.

Hardware Assistance by AdvancedTCA Components to a Fast Failover

AdvancedTCA can be used to achieve a fast failover with a fast fault management cycle process. The AdvancedTCA components listed below are key pieces in a fast failover. These pieces are shown in [Figure 2](#) below [5].

- (1) *Update channel.* This low-latency board-board channel can be used to check the heartbeat of the board. The HA middleware can use this channel to synchronize application state and data with a redundant application on the standby node. The Switch Fabric Bypass interface of the update port can be used for checkpoint application data and state information to a standby application running on another board, thus bypassing the high-latency switching fabric [3], [4], [5].
- (2) *Shelf manager.* The shelf manager can use the health data provided the HA middleware to take

corrective action including generating alerts and e-mails, and reconfiguring a warm standby board.

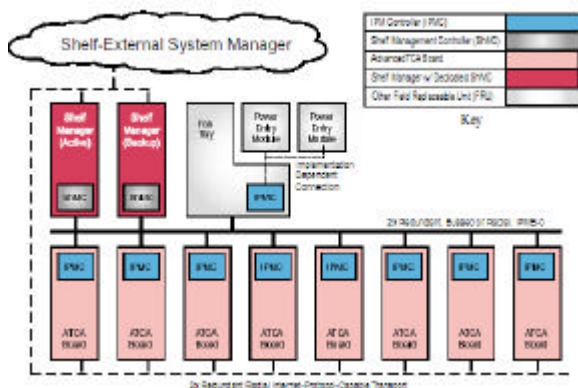


Figure 2: Key components of a fast failover

CARRIER-GRADE LINUX

Along with hardware and system management, robust operating systems and middleware are imperative to achieving a truly high-availability (HA) system. Carrier-Grade Linux (CGL) addresses the issues of HA and total cost of ownership (TCO) at the operating system and middleware level. The Open Source Development Labs (OSDL) CGL Working Group is the standards group that produces the CGL requirements, which define the additions to Linux that are needed by the telecom industry. CGL requirements ensure that the availability and service response characteristics needed in a carrier environment are fulfilled, thus ensuring a system that is truly HA. Subsequent sections describe how CGL enhances platform availability and reduces TCO as well as how CGL supports HA middleware services, such as those standardized by the Service Availability Forum (SAF).

Carrier-Grade Linux and HA

The OSDL Carrier-Grade Linux Working Group took into account the special considerations of AdvancedTCA when developing the CGL requirements. Because of this, the CGL requirements outline several key areas of operating system support for AdvancedTCA, enabling solution providers to create HA solutions on top of CGL-conforming operating systems running on AdvancedTCA blades.

Carrier-Grade Linux Support for Platform HA

This section describes some platform-specific strategies aimed at increasing HA and the support the OS provides to enable these strategies.

Redundancy Support

Node boards that are used in AdvancedTCA-based equipment will typically employ device-level redundancy to eliminate single points of failure on the board. Such redundancy, in combination with the HA features in the AdvancedTCA chassis and other associated equipment (like the fabric/switch boards) and OS and middleware support, ensure the HA of specific functionalities. Two key functionalities that we address in this section are (1) Ethernet connectivity between and to the nodes in the chassis and (2) disk data storage.

Ethernet Connectivity

To ensure HA Ethernet connectivity between nodes in an AdvancedTCA system, the node boards will typically include redundant Network Interface Controllers (NICs). These redundant NICs are combined to appear as a single virtual Ethernet interface by using the technique of “Ethernet bonding.” When one of the NICs in the bonded set fails, the other NIC (in the case of two bonded NICs) takes over transparently, thus preventing a single point of failure. A bonding driver combined with a system-level configuration facilitate Ethernet bonding. Support for Ethernet bonding is a key CGL requirement. Examples of conforming bonding driver implementations are the bonding driver developed by the SourceForge “Channel Bonding” project [16] and the Intel Advanced Networking Services Driver (iANS).

Bonding drivers can set up the bonded NICs to operate in a variety of modes ranging from the simple link failover mode where at any time only one NIC is active, to “link aggregation” modes where all the bonded NICs transmit and receive simultaneously, to varying degrees.

Disk Data Storage

To achieve HA of 99.999% or more, AdvancedTCA systems need to ensure that disk data corruption or disk failures do not lead to system downtime. Employing a Redundant Array of Inexpensive Disks (RAID) system is one way to meet this requirement. The OSDL CGL requires software RAID 1 [17] along with other user-space RAID tools (Disk Mirroring) support, which essentially provide for the maintenance of duplicate sets of all data on separate disk drives. In addition, the RAID 1 support will also allow booting off a mirrored drive if the other disk drive fails. In addition to mirroring the disk data, AdvancedTCA systems impose additional requirements on the OS support, specifically to handle hot swapping defective disk drives (or entire storage boards) and to re-synch the replacement drive or board after insertion, without having to take the system down.

The Linux Multiple Devices (MD) driver along with other user space RAID tools is an example of a conforming software RAID 1 implementation.

Hot Swap Support

In addition to managing redundant devices, the OS on AdvancedTCA systems should also support hot swapping of individual devices and even entire boards. A key requirement on the OS that arises as a result of devices and platforms being hot swapped is the device naming requirement described below.

Persistent Device Naming

A fully utilized AdvancedTCA solution will need to provide seamless removal and insertion of components without adversely effecting the larger telecommunications application. A CGL-conforming OS will provide a policy-based subsystem for naming of transient components. This will allow long-running HA applications to gracefully deal with changing resources without the need to restart applications, or reboot the OS.

Carrier-Grade Linux Support for HA Middleware

The HA middleware discussed earlier relies on a set of services provided by the OS. This section deals with some of those services and the requirements they impose on CGL.

Full Kernel and System Library Support for IPMI 1.5
Having Intelligent Platform Management Interface (IPMI)-enabled hardware is only half the story. Without full support for IPMI in the operating system, solution providers are still forced to depend on costly custom software stacks to create complete HA solutions.

A CGL-conforming OS provides both an IPMI driver exposing a direct connection to the blade's baseboard management controller, along with a user space framework designed for enabling system management software.

In addition to exposing programmatic access to IPMI, a CGL-conforming OS also provides kernel-level primitives for immediately signaling the shelf manager on a critical kernel error via the baseboard management controller over the Intelligent Platform Management Bus (IPMB). This enables the shelf manager to take appropriate action (such as forcing a fast cluster node failover) to maintain service availability targets.

SAF Hardware Platform Interface Support

The SAF is an industry consortium of major players in the telecommunication industry including leading Network Equipment Providers (NEPs), Original

Equipment Manufacturers (OEMs), platform and building-block providers, and independent software vendors. The goal of the SAF is to develop a framework and specifications for service availability to enable highly available carrier-grade systems to be developed using off-the-shelf platforms, middleware, and applications. The SAF has published the Hardware Platform Interface (HPI), which is the programming interface between the HA middleware and the platform. Members of the forum are working together to create multiple implementations of the specification. A CGL-conforming OS will provide such a HPI implementation, allowing standardized HA middleware and applications.

The HPI implementation provided with the CGL-conforming OS will provide the next level of complexity on top of the IPMI, utilizing multiple communication paths (IPMB, Ethernet, etc.) as needed to meet the extreme demands of a carrier environment.

Carrier-Grade Linux and TCO

In addition to defining a system that is HA because of the AdvancedTCA focus, CGL requirements also decrease TCO in a similar way to AdvancedTCA by being open and standards-based. CGL reduces the cost of obtaining hardware and software (OS and applications) and reduces maintenance time and unplanned downtime.

Reduction of Hardware and OS Costs

The openness of Linux means that using a Linux distribution with carrier-grade functionality can reduce both hardware and OS costs. Since the Linux kernel supports multiple architectures, customers have a choice of cost-effective architecture options. A CGL distribution is derived from the freely available Linux kernel, and there are multiple CGL distribution vendors competing for market share, so customers have additional freedom in choosing the distribution that meets their cost and feature needs.

Reduction of Application Costs

By detailing the list of features required for a carrier-grade operating system, the OSDL CGL requirements have increased the feature set available in an OS, reducing the need for users to develop and maintain custom solutions in order to implement a full carrier-grade solution. A large number of the CGL requirements are implemented as open source projects, furthering the technology by drawing documentation and bug fixes from the larger open source community. Other requirements are a more natural fit for proprietary implementations based on open interfaces, and are fueling a growing ecosystem of telecom independent software vendors (ISVs). The combination of these two complementing

software development models results in a longer lifetime for the technology.

Reduction of Development Costs

CGL requirements, like AdvancedTCA, are also standards-based, which allows for horizontal building-block solutions. The Portable Operating System Interface (POSIX) specification [11] defines a standard interface for an OS; porting from one POSIX-compliant operating system to another should not require costly rewrite, since the same standard interface is used. In addition, other stable Unix* interfaces such as Simple Network Management Protocol (SNMP), which is described on the next page, and all the individual interfaces specified in the SAF Application Interface Specification either are currently or are planned to be CGL requirements. In addition to source-level interfaces, the standard binary interface as detailed by the Linux Standard Base (LSB) [12] is also a CGL requirement. LSB's goal is to allow Linux software applications to run on any LSB-compliant system. Customers can therefore choose the LSB-compliant CGL distribution that meets their cost and feature needs and know that Linux software applications can be run on that system.

Reduction of Maintenance Time

In addition to being standards-based, CGL requirements also focus on serviceability, which reduces TCO by ensuring that CGL systems can be maintained, upgraded, and debugged quickly. The SNMP [13] is one such serviceability requirement. SNMP describes a simple, basic, and lightweight network management tool consisting of a Management Information Base (MIB) that contains information about the resources and activity on a node, managers that request these data, and agents that run on the network devices to be managed and that provide MIB data [14]. CGL defines a baseline set of MIBs that provide enough information for a CGL system to be more easily serviced. Another set of CGL requirements that assist in serviceability are those focused on ensuring that sufficient debug information is available after a kernel dump, thus allowing quicker debugging of system failures. CGL requirements focus on enabling the saving of kernel dumps to multiple targets (disks, memory, locations across the network) as well as producing configured dumps with only the data structures the user would like to see. CGL requirements also have been expanded to include the ability to take dump information from the system and applications as well as from the kernel.

*Other brands and names are the property of their respective owners.

Reduction of Unplanned Downtime

In addition to maintenance costs, unplanned downtime is also costly. CGL has specific requirements to reduce and prevent unplanned downtime.

Downtime Root-Cause Analysis Features

Regardless of how well a system is tested, unplanned downtime from software or hardware failure is going to happen. An AdvancedTCA-based solution lends itself to uninterrupted service availability via availability clusters instrumented for standard AdvancedTCA features, but the failover hardware is no longer redundant when a failure happens, and costly engineering hours spent tracing the root cause of a failure can quickly eat away at an operation budget.

A CGL-conforming OS provides many mechanisms designed to minimize root-cause analysis downtime by enabling debugging of a live system. Such features include the following:

- Kernel debugger
- Dynamic insertion of debug probes in both kernel and user space executing code
- Standardized kernel messages
- Kernel profiling support
- Kernel summary dump and dump analysis tools
- Live system application dump

Combining these advanced live debugging features that come with any CGL-conforming OS, together with the familiarity of these features to nearly every recent computer science graduate around the world, causes downtime (and therefore downtime expense) to be dramatically reduced.

CGL Security Features

A lack of security is another reason for unplanned downtime, and one CGL requirements section is devoted to security. One basic framework for security that CGL requires is Linux Security Modules (LSMs) [15]. LSM is a Linux kernel framework that supports security modules, primarily access control modules. Customers can create or choose an open source project implementing security modules with the functionality they would like. Access control modules prevent and permit access to objects and processes in a system, and therefore help prevent malicious users from executing programs on a node or from getting access to files on a node.

CONCLUSION

Today's market requires that carrier-grade solutions be highly available to meet the 99.999% demands on carrier-grade systems as well as have a low total cost of ownership to enable system owners to make a profit. The solution to increasing availability with a cost-effective system requires a hardware, middleware, OS, and application-level solution. AdvancedTCA provides an open, standards-based carrier-grade equipment solution that increases high availability (HA) by enabling a robust failover solution as well as having a robust shelf management definition. The Intel NetStructure Chassis Management Module implements the AdvancedTCA shelf management and can therefore be used to increase the HA of a system. Carrier-grade Linux provides an open, standards-based carrier-grade OS and application solution that increases HA by defining requirements geared at increasing the availability of a system at the OS and application level. Because both AdvancedTCA and Carrier-Grade Linux (CGL) are open and standards-based, they reduce total cost of ownership (TCO) by reducing the dependency on costly vertical, proprietary solutions. A carrier-grade solution involving AdvancedTCA equipment as well as a CGL distribution can help increase HA and decrease TCO.

ACKNOWLEDGMENTS

We thank the following people for their comments, useful suggestions, and support: Chetan Hiremath, Tisson Mathew, Udayan Mukherjee, Mark Skarpness, and Imad Sousou.

REFERENCES

- [1]. "Go Ahead. High Availability and More: Achieving a Service Availability™ Solution," <http://data.goahead.com/sr/FAQ-fault-management.pdf>.
- [2]. Intel® Corporation, "A Framework for System High Availability," June 2000, pp. 273389-001.
- [3]. <http://www.picmg.org>
- [4]. <http://www.advancedtca.org/>
- [5]. Intel® AdvancedTCA* website: <http://developer.intel.com/technology/atca/>
- [6]. IPMI information: <http://www.intel.com/design/servers/ipmi/>
- [7]. Service Availability* Forum: <http://www.saforum.org/home>
- [8]. HA Middleware: http://www.goahead.com/products/white_papers.htm
- [9]. CGL requirements: http://www.osdl.org/lab_activities/carrier_grade_linux/documents.html
- [10]. Linux kernel: <http://www.kernel.org>
- [11]. POSIX: <http://www.opengroup.org/austin/aardvark/finaltext/>
- [12]. LSB: <http://www.linuxbase.org>
- [13]. SNMP: <http://www.ibr.cs.tu-bs.de/projects/snmpv3/>
- [14]. Stallings, W, "Security Comes to SNMP: The New SNMPv3 Proposed Internet Standards," *The Internet Protocol Journal*, vol. 1, December 1998, pp. 2, (http://www.cisco.com/warp/public/759/ipj_3.pdf).
- [15]. LSM: <http://lsm.immunix.org/>
- [16]. Bonding: <http://sourceforge.net/projects/bonding/>
- [17]. RAID 1: <http://en.tldp.org/HOWTO/Software-RAID-HOWTO.html>

AUTHORS' BIOGRAPHIES

Sharad Garg is a modular-server architect at Intel. His research interests include modular storage servers, distributed computing, and distributed file systems. He received M.S. and Ph.D. degrees in Computer Science from the University of Connecticut. His e-mail is sharad.garg at intel.com.

Raj Sistla is a senior software engineer in the Embedded IA Division (ICG/NPG) at Intel. His interests include modular servers and server management. He received his M.S. degree from the State University of New York at Buffalo and is currently working on his M.B.A degree from Indiana University. His e-mail is Raj.Sistla at intel.com.

Ramesh Caushik is a staff software engineer at Intel. His interests include networking, telecom applications, and the Linux operating system. He joined Intel in 1993 and has a M.S. degree in Computer Science from Utah State University. His e-mail is Ramesh.Caushik at intel.com.

Julie Fleischer currently works as a software engineer in Intel's Telecom Software Program group and participates in the Carrier-Grade Linux Working Group. Her interests include software quality and software process improvement. She received her M.S. degree from Case

Western Reserve University. Her e-mail is Julie.n.Fleischer at intel.com.

Rusty Lynch is a software architect at Intel where he is engaged in various open source activities relating to the Linux operating system in carrier-grade environments. His interest is primarily in core operating system enabling, but he has worked in many other aspects of Linux since joining Intel in 1996. His e-mail is Rusty.Lynch at intel.com.

Copyright © Intel Corporation 2003. This publication was downloaded from <http://developer.intel.com/>.

Legal notices at <http://www.intel.com/sites/corporate/tradmarx.htm>.

For further information visit:

developer.intel.com/technology/itj/index.htm