A Flexible ServerNet-based Fault-Tolerant Architecture

William E. Baker  Robert W. Horst  David P. Sonnier  William J. Watson

Tandem Computers Incorporated
10555 Ridgeview Court
Cupertino, CA 95014

Abstract
This paper introduces a new fault-tolerant architecture that combines the best attributes of the software fault-tolerant Tandem NonStop systems with the hardware fault-tolerant Integrity systems. This architecture is based on the ServerNet System Area Network (SAN). ServerNet, formerly called TNet, is a packetized byte-serial multistage network that supports both I/O and interprocessor traffic in fault-tolerant systems. Dual-ported CPUs and I/O controllers connect to independent subnetworks in a variety of different network topologies. Systems can expand either through shared or distributed memory multiprocessing. A separate maintenance system controls system initialization, online configuration changes, and error reporting. The architecture's flexibility makes it suitable for a wide range of environments with varying requirements for performance, fault tolerance, and software compatibility.

1.0 Introduction

Reliable computer systems are key to addressing critical applications in diverse markets such as retail, finance, and telephone switching. These applications have wide-ranging requirements for scalability, open standards support, and fault tolerance [1,2,3]. Tandem currently addresses the diverse market needs for reliable computing with two different product lines: Tandem NonStop systems and Tandem Integrity Systems.

Tandem NonStop systems, first introduced in 1976, are based on distributed memory multiprocessing. These systems use message-passing between process pairs to allow recovery from hardware faults and software errors [4]. The shared-nothing NonStop SQL database has shown linear scalability for transaction processing in configurations of more than 100 processors. NonStop systems initially targeted online transaction processing (OLTP), but are now extensively used in messaging and decision support applications.

Tandem introduced its Integrity systems in 1990 to address the need for a fault-tolerant system to run existing UNIX applications without change. Integrity systems use hardware redundancy to mask faults and continue processing without impact to applications. Current systems process a single instruction stream, with fault tolerance implemented through triple-modular redundancy [5]. Integrity systems are frequently used in telephone and cellular networks, and as well as other commercial applications requiring reliable, open systems.

While the two product lines have distinctions, they also meet similar overall requirements and share many common technologies and components. All Tandem systems guarantee data integrity and fault-tolerance, and provide scalability and the ability to perform online upgrades. Both the NonStop and Integrity lines allow for online service through hot-plugging of boards, power supplies, and fans. The packaging of both systems allows installation in offices, standard computer rooms, or telephone company central offices. These two systems have many common components such as disk drives, memory components, and microprocessors.

In 1991, we began a program to combine the best attributes of both product lines into a single product line. This program addresses flexibility in a number of different dimensions: fault tolerance (software- and hardware-based recovery), scalability (distributed memory clusters and shared-memory multiprocessing), and operating systems (NonStop Kernel, UNIX, and Microsoft Windows NT).

The development of the ServerNet system area network provided the main vehicle for combining the architectures. ServerNet is a point-to-point byte serial multistage network used for both interprocessor and I/O communications. ServerNet includes extensive features for fault detection and isolation, and includes direct support for alternate communication paths to allow continued operation in the presence of network failures [6]. The introduction of this network offers new capabilities to both product lines, including greater scalability, interfaces to open bus standards, and improved support for multimedia applications.
2.0 NonStop Systems Architecture

Figure 1 illustrates the NonStop systems architecture. The basic architecture connects two or more CPUs through a pair of high-speed interprocessor buses. Each processor has one or more I/O channels connecting it to dual-ported disk and communications controllers. The first five generations of NonStop systems, the NonStop I, II, TXP, CLX and VLX, all have one I/O channel per processor and connect up to 16 processors through a pair of shared buses [7]. The NonStop Cyclone and Himalaya K10000/K20000 systems segment the interprocessor bus into four-processor sections to improve interprocessor bandwidth. Four sections can connect with fiber optic links to form a 16-CPU node. The NonStop II, TXP, VLX and Cyclone systems also support a fiber optic ring connection [8] that connects up to 14 nodes for fast interprocessor communications within a 224 processor domain. The Cyclone system can attach multiple I/O channels per processor, with four channels controlled by a pair of DMA engines.

After the development of the Cyclone system, Tandem shifted from the use of custom designed CISC CPUs to MIPS®-based RISC CPUs. The CLX/R, Cyclone/R and K200 systems use R3000™ processors, while the Himalaya K10000, K20000 and K2000 use the R4400™ processor. Himalaya systems also introduced the TorusNet interprocessor network to connect four-processor sections in a redundant two-dimensional, fiber-optic torus network [9].

All NonStop hardware components use fail fast designs in which each component must either function properly or stop. Early systems implemented fail fast operation with the use of parity, coding, or state machine checks in each logic function. Current designs rely on duplication and comparison to detect errors in complex logic. All systems with microprocessor-based CPUs compare the outputs of duplicated, lockstepped microprocessors to guarantee data integrity and fast fault detection [10]. In NonStop systems, software has the responsibility to recover from errors once they have been detected and contained by the hardware.

The operating system for NonStop systems, the NonStop Kernel, has evolved from a proprietary operating system to one that also provides open interfaces by adherence to industry standards. The operating system provides low-level checkpointing to ensure persistence of critical processes, and provides a layer on which to build both the proprietary Guardian environment as well as an open Posix-XPG/4 environment. The message-based NonStop Kernel also provides transparent scalability within a 16-CPU node, a 224-CPU TorusNet domain, or a 4000+ processor LAN- or WAN-connected network.

3.0 Integrity Systems Architecture

Tandem designed its Integrity systems to tolerate all single hardware faults while providing 100% application portability for standard UNIX applications. Integrity systems use triple-modular redundancy (TMR) to mask hardware faults in the processor, cache, and main memory (see Figure 2).

Three processors run identical code streams but run off of independent clocks. The processors synchronize during access to global memory and while servicing external interrupts. A pair of TMR controller modules votes all

---

![Figure 1. NonStop Systems Architecture](image1)

![Figure 2. Integrity Systems Architecture](image2)
accesses to global memory. The voters pass valid requests to the rest of the system and detect potential processor failures. The TMR controller and I/O processor designs use self-checking to detect failures. Each peripheral controller contains a standard VME board connected to a pair of parity-protected I/O buses through an adapter board that can switch the controller between the two I/O processors.

Integrity systems have an advantage in that main processor boards do not require designs with self-checking logic. However, this makes the processor board designs different from those in NonStop systems, even though both use the same microprocessors. The new system architecture incorporates the basic Integrity design requirements while remaining compatible with the Non-Stop system requirements.

4.0 System Architecture

4.1 Overview

The new ServerNet-based system architecture combines the features of Tandem NonStop systems and Tandem Integrity systems. While these systems share some common features, they use different approaches to fault tolerance. The new architecture meets the common goals by implementing a flexible method of connecting standard system building blocks, such as CPU/memories, storage subsystems and communications services.

Figure 3 shows the architecture of a typical ServerNet-based system. This system contains multiple CPU and I/O nodes connected through a ServerNet System Area Network. The system area network uses ServerNet links to connect these nodes with router ASICS. In typical configurations, most nodes have dual-ported ServerNet interfaces, with connections to independent X and Y subnetworks. Sets of up to four CPUs can also connect to an optional coherency bus that supports shared-memory image and cache coherency when running code designed for shared-memory multiprocessing.

When running operating systems that support software fault tolerance (such as the NonStop Kernel), CPU nodes execute independent code streams. Unlike previous systems, which used separate interfaces for interprocessor messages and I/O communications, the CPUs use ServerNet for all data transfers.

Operating systems that do not support software fault tolerance require configuration of fail fast CPUs as duplex pairs to provide hardware fault tolerance. The pairs of CPU nodes run identical code streams. If one CPU of a pair fails, the other continues to run. These CPUs share a common ServerNet node identification. All packets bound for this destination are duplicated and delivered to both CPUs simultaneously. In the absence of any faults, the paired CPUs create identical outbound packets. Therefore, the ServerNet routing logic can choose to forward either node's packets in the normal case. Faults are detected using ServerNet's extensive fault detection capabilities. Section 4.5 describes the protocol used to recover from CPU faults.

The Integrity system requires three processor chips and three memory arrays to provide hardware fault tolerance. The new architecture requires four processor chips (two per CPU module) and two memory arrays. The size of the memory array substantially affects the cost trade-off between these two approaches. For typical systems, the two methods have comparable costs.

4.2 ServerNet Overview

ServerNet is a fast, scalable, reliable system area network that has the flexibility to connect large numbers of CPUs and I/O peripherals. ServerNet is a wormhole-routed, packet-switched point-to-point network that focuses on reducing latency and assuring reliability. The ServerNet physical layer has independent transmit and receive channels, each with a 9-bit command/data field and a clock (see Figure 4). The command/data field encoding provides 256 data symbols plus up to 20 command symbols. ServerNet uses these command symbols for link-level flow control, initialization and error signaling. Encoding the commands and data into the same lines reduces pin counts and improves control logic fault detection.
ServerNet transfers data at the rate of 50 megabytes per second through backplane connections or copper cables. The initial speed of 50 MHz was chosen to perform well ahead of current peripheral needs, while keeping costs low.

A system uses ServerNet for CPU-to-CPU, CPU-to-I/O, and I/O-to-I/O communications but not for microprocessor-to-memory data transfers. Microprocessor-to-memory transfers stay local to CPU nodes.

When increased bandwidth is required, ServerNet links can be added to provide bandwidth into the GByte/sec range. Future generations of ServerNet will increase link performance as needed to keep up with improvements in silicon technology and peripheral speeds.

Future adapters will also extend ServerNet distances through serial fiber optic links. This conversion will be relatively straightforward because all control functions, including flow control, use the same command/data lines.

ServerNet packets consist of an 8-byte header, an optional 4-byte address, a variable sized data payload, and a CRC (see Figure 5). The header specifies the destination and source node IDs, the data length, and the operation to perform. The 4-byte address field provides a 4-gigabyte window into the destination node's address space. Typical I/O devices use a 1:1 mapping. CPUs map the address in a packet to a physical memory address with an Address Validation and Translation (AVT) table. The AVT validates remote accesses to memory. Each entry in the table provides one CPU or I/O node read and/or write access to any subset of a 4 kilobyte range of the local memory. Using multiple AVT table entries, a CPU node can provide one node with access to large segments of its memory or provide multiple nodes with access to the same segment of memory. A broken node can only corrupt the segments of memory to which the CPU has granted it access. The AVT also provides a mapping function to allow contiguous transfers to/from scattered pages in physical memory.

ServerNet transactions consist of a request and a corresponding response that must return before the expiration of a timeout counter. All ServerNet nodes support multiple outstanding requests to and from other nodes. ServerNet supports read and write transactions. The recipient of a specific transaction may elect to perform local actions beyond those required by the packet layer protocol. For example, current CPU and I/O nodes translate write transactions to specific addresses into local interrupts.

The ServerNet router ASIC uses a 6x6 crossbar and makes routing decisions based on the destination ID in the packet. Routers have FIFO buffers on inputs, logic for arbitration and flow control, a routing table implemented in RAM, and a crossbar switch (see Figure 6.). Software can modify the routing tables to initialize or reconfigure the network.
4.3 CPU Design

Figure 7 shows a block diagram of a fail-fast CPU module, which provides processor and memory resources to the system. Two ServerNet ports connect each CPU to other CPUs and to I/O devices through the system area network. For shared memory expansion, multiple CPUs may connect to each other through a coherency bus.

![CPU Block Diagram]

**Figure 7.** CPU Block Diagram

The CPU has two microprocessors, each with an independent secondary cache. Each microprocessor connects to the memory bus through a processor interface ASIC. These ASICs compare the outputs of the microprocessors on memory cycles to detect all microprocessor and cache errors. The CPU also contains memory protected by an ECC code that corrects all single bit errors and detects any error in a single DRAM or address line. Dual memory controller ASICs connect the DRAM array to the memory bus. These ASICs check each other’s outputs on memory cycles.

CPUs can have direct access to the memory of other CPUs through the optional coherency bus. This coherency bus provides the hardware support to run standard UNIX or Windows NT applications that use symmetric multiprocessing (SMP). Each CPU connects to the bus through a self-checked pair of coherency interface ASICs. These ASICs provide cache-coherent access to shared memory using duplicate tag stores and a standard cache invalidate protocol [11]. They also provide cache coherence for I/O accesses to memory. The coherency bus protects data transfers using ECC. Checking the ECC syndrome for data transferred across the bus and comparing ASIC outputs detects faults in either the bus or interface ASICs.

A shared-nothing operating system such as the Non-Stop Kernel can disable the coherency bus to increase fault isolation. In this mode, system software enforces cache coherency for memory targeted by ServerNet transactions. With the coherency bus enabled, a fault in one CPU connected to the coherency bus may result in the failure of all CPUs connected to that coherency bus.

The processor interface ASICs also provide two ServerNet ports for the CPU. The receive links of both ServerNet ports connect to both processor interface ASICs. Each ASIC creates transmit values for both ServerNet ports but only transmits one. The ASIC receives the data transmitted by the other ASIC, compares the value received with the value it creates, and signals an error on any disagreement.

4.4 I/O Design

The new I/O system provides massive scaling in connectivity and bandwidth. It efficiently supports distributed computing by allowing any CPU to access any I/O controller, and by allowing direct controller-to-controller communication.

ServerNet provides flexible attachment of custom controllers designed to take full advantage of ServerNet’s many features as well as the attachment of commodity controllers with standard bus interfaces. Figure 8 shows the block diagram of a typical ServerNet Bus Interface ASIC. All I/O adapters use an ASIC similar to this to convert I/O bus transfers to transmitted ServerNet packets and to convert received ServerNet packets to I/O bus transfers.

![ServerNet Bus Interface ASIC]

**Figure 8.** ServerNet Bus Interface ASIC
transfers. Different ServerNet Bus Interface ASICs support different I/O buses. Each ServerNet Bus Interface ASIC provides an additional measure of fault containment by allowing only specified ServerNet nodes to access each I/O controller.

Custom I/O controllers with a microprocessor bus such as 68040 or PCI can take full advantage of ServerNet’s many features. Figure 9 shows an example of a such a ServerNet-aware I/O controller. The controller firmware specifies the target ServerNet destination node ID for controller-generated requests by programming the ServerNet Bus Interface directly. ServerNet-aware controllers send interrupt requests to the host as ordinary ServerNet write requests, which the processor interface ASICs map to CPU interrupts. ServerNet-aware controllers can directly support a shared distributed programming model, where multiple loosely coupled processors share access to the controller. ServerNet-aware controllers reduce cost and increase performance for critical functions, such as SCSI, Ethernet and ATM.

The ServerNet bus interface can also support standard-bus I/O controllers. This allows the use of unmodified controller designs with minimal changes to device driver software. Figure 10 shows how a standard-bus I/O control connects to ServerNet. The ServerNet bus interface maps standard-bus data transfers and interrupts to appropriate ServerNet packets. Standard-bus I/O controllers do not support the shared distributed host model, but instead rely on host software to manage device ownership. Standard-bus I/O support allows maximum flexibility by supporting the widest variety of interfaces.

I/O Controllers can have either one or two ServerNet ports. SCSI devices use single-ported ServerNet controllers but have dual SCSI buses, as shown in Figure 11. Each SCSI adapter card has a single ServerNet interface, but has a pair of SCSI bus interfaces cross connected to the other card. This provides two completely independent access paths to all SCSI devices, even in the presence of most adapter faults. Disk mirroring or RAID provides

---

Figure 9. ServerNet-Aware, Multiple Function, I/O Controller

Figure 10. Standard Bus I/O Controller

Figure 11. SCSI Redundancy
fault tolerance for disk failures or for adapter failures that disable an entire SCSI string. The dual SCSI controllers minimize the amount of rem mirroring required after adapter failures.

Reliable ethernet implementations, as shown in Figure 12, use two single-ported ServerNet Ethernet adapters sharing a single MAC address on the Ethernet. Software implements a hot standby algorithm for Ethernet adapter recovery.

Most other ServerNet I/O adapters have two ports. The dual ports provide two independent access paths to the adapter. Failures in one ServerNet path cause requests to time out. Software reroutes and retries the request (see Figure 13).

4.5 Duplex Operation

A hardware fault-tolerant system is created by configuring two CPU/memory modules as a duplex pair. The system keeps the memory and cache arrays in the duplex pair identical by executing the same code on the same data on both CPUs, and by forwarding all inbound I/O to both memories. Both CPUs generate identical outbound I/O streams. The routers select one of the identical I/O streams to forward to the I/O controllers or to other processors.

Duplex operation affects three areas of the system: inter-module clock synchronization, ServerNet link-level synchronization, and ServerNet fault handling. Duplex operation requires two routers in different ServerNet subnetworks, and two CPUs connected to different coherency buses (for shared-memory operation). No single failure in the system can disable both CPUs or both routers. All CPUs that share a coherency bus have a common clock. We call this combination of CPUs, coherency bus, and clock a slice. A system configured for duplex operation has two slices.

Operation of the duplex system requires frequency locking the clocks in the two slices to each other. The CPUs in each slice execute identical instructions at the same clock count. Inbound I/O arrives at each memory interface on the same clock. In this way, no normal error-free operation can cause the slices to behave differently from each other.

The CPU modules have extensive fault checking. If any of the fault checking logic on either CPU detects a fault, that CPU halts. Halting the CPU forces both ServerNet ports to transmit an illegal pattern. In this way, a router detects a failed CPU by its ServerNet errors. A fundamental system rule states that all CPU faults must result in bad ServerNet transmissions.

When a router connected to a duplex CPU detects an error, it initiates an error recovery protocol. The protocol runs entirely without software interaction and results in choosing one CPU to disable and one to keep running. The protocol guarantees that a good CPU keeps running. However, cases exist where the disabled CPU has no faults. For example, faults in one of the routers or in one of the ServerNet links can result in the disabling of a CPU. In these cases, the maintenance system can disable the affected router and the bring the disabled CPU online.

If a router detects a fault on a ServerNet link from a CPU while forwarding a packet from that CPU, it marks the packet as invalid. Any ServerNet node that receives this packet will ignore it. This means that CPU, router, or link fault can result in the loss of one or more packets. In normal duplex operation, one of the two routers for the duplex processors forwards packets from each CPU. This limits the loss to packets in just one ServerNet subnetwork. The interface chips detect lost ServerNet packets with packet timeouts. I/O software performs the recovery by retrying through an alternate path.
4.6 Distributed Scalability

Scalability has long been an important attribute of NonStop systems. This scalability becomes even more important in new large-scale decision-support applications. Some NonStop applications already incorporate over a hundred processors and over a terabyte of disk storage in a single system. The increasing use of image and voice data will compound the need for high-bandwidth, scalable system interconnects.

While previous systems had well-defined scalability limits, ServerNet provides open-ended possibilities for scaling. Systems will typically have a few built-in ServerNet expansion cable connections available, as well as the option of installing router cards in ServerNet expansion slots. Also, each addition of a CPU provides more ServerNet links and effectively increases the total I/O bandwidth in the system. Unlike other massively parallel architectures, ServerNet networks are not restricted to regular topologies such as hypercubes or tori. The network supports the addition of ServerNet links anywhere that requires additional bandwidth. Applications with modest interconnect requirements can use low-cost networks, while data-intensive applications can have more fully connected networks.

Tandem designed and developed Guardian and the NonStop Kernel for distributed processing. Over the last twenty years, we have gradually discovered and removed software bottlenecks to scalability. We also have added new capabilities to the NonStop SQL database to make use of distributed hardware when executing complex database queries.

Standard UNIX systems also have started moving towards distributed scalability. Workstation clusters have begun to replace supercomputers in some scientific applications. This trend will fuel increased development of applications and operating environments for distributed computing. As distributed UNIX environments begin to emerge, ServerNet-based systems will be ideal for reliably running these distributed applications.

4.7 SMP Scalability

Unlike the NonStop Kernel, which was designed as a distributed operating system, most available operating systems do not scale in a distributed fashion. Instead, these systems typically use symmetric multiprocessing (SMP) as the scaling mechanism [11]. This mechanism has a single shared memory, shared I/O, and cache coherency.

The new architecture supports a flexible shared-memory implementation. As shown in Figure 3, multiple CPUs connect together via a coherency bus. Rather than using a single global memory, the system distributes memory across the CPUs. This results in a non-uniform access time to memory. However, the hardware provides a global address space and maintains cache coherency.

In order to improve performance, each CPU supports a private memory space. Private memory starts at physical address 0 on each processor and has a configurable size. Only the local processor can access private memory.

This memory architecture gives three distinct classes of memory as shown in Figure 14. Private memory has the shortest access time, access to global memory on the local CPU board takes longer, and access to global memory on another CPU board takes longer still. The private memory of each CPU contains a copy of read-only portions of the operating system's code and data. In addition, the operating system may place some processor-private data in private memory. Performance estimates predict that placing these items in private memory and randomly allocating global memory will provide performance comparable to a system with a single global memory. Use of memory affinity can achieve additional performance improvements.

Shared memory systems can function in either simplex or duplex mode. A software fault-tolerant system built out of SMP nodes can use simplex mode. A fully hardware fault tolerant system with SMP scalability requires duplex mode.

The coherency bus can also support hybrid distributed systems, where the processors use shared memory for interprocessor communication. In this case, the system maps most of memory as private and only places the shared data in global memory.

![Four-Way SMP Memory Map](image)

Figure 14. Four-Way SMP Memory Map
4.8 Maintenance System

The maintenance system has responsibility for system initialization, fault reporting, diagnostics, and environmental control. A pair of service processors (SPs) in each cabinet manage the maintenance system. The SPs function as ServerNet I/O controllers. SPs in different cabinets communicate with each other only via ServerNet. The SPs provide full local and remote support for both normal and down system operations, allowing for full lights-out installations.

The maintenance system uses dual system-maintenance buses. The maintenance buses form redundant trees, independent of normal system functional paths. The maintenance buses provide a path for two industry standard interconnects: Motorola's Serial Peripheral Interconnect (SPI) and IEEE 1149.1 JTAG scan. The Serial Peripheral Interconnect functions as a low-cost serial I/O bus for communication with all environmental status and control facilities. The maintenance system uses the scan functions to control, initialize, test, and monitor all ASIC operations. An extension to the IEEE 1149.1 standard developed by Tandem allows access to selected ASIC registers via the scan path. This access occurs without interferring with normal ASIC operation. This general mechanism provides a means for ASIC initialization, ServerNet topology determination, and error reporting.

4.9 Initialization

The service processors have responsibility for system initialization. At system startup, they create a full inventory of the resources available in the cabinet using the maintenance buses and serial EEPROMs which contain configuration information for all components in the system. The service processors then initialize the various components via the scan interface.

The service processors in the various cabinets then go through a dynamic ServerNet topology determination process. This process does not depend on any preconfigured knowledge of the system. It operates below the ServerNet packet layer since it precedes establishment of the ServerNet routing. Once the service processors have determined the topology, they assign ServerNet IDs and program the routing tables inside the ServerNet routers.

The service processors then start execution of the operating system bootstrap on the host processor. The service processors provide the bootstrap loader and the operating system with hardware inventory and configuration information, including the ServerNet address of every device in the system.

5.0 Software

The key attribute of this new architecture is flexibility. It fully supports distributed, software fault-tolerant systems such as the NonStop-Kernel. The hardware provides support for fault-tolerant SMP configurations ideal for operating systems such as UNIX or Windows NT. Systems can also support software such as Oracle Parallel Server, which uses a shared disk model of distribution.

The common interconnect of ServerNet-based systems allows construction of hybrid systems. For example, a Windows NT system can act as a front end for an Oracle Parallel Server database, all connected via ServerNet.

Support for a wide variety of operating systems requires keeping the porting effort to a minimum. The new architecture achieves this goal by keeping platform-specific code separate from the operating system and by ensuring that fault tolerance requires no core operating system modifications. Much of the platform-specific code runs on the service processor, independent of the main processor's operating system.

A portable suite of applications provides system management functions for all ServerNet-based systems. These applications monitor hardware states, report failures, take automated recovery steps when appropriate, and provide interfaces to remote support applications using industry-standard protocols.

6.0 Conclusions

ServerNet provides the reliable infrastructure for building both large and small fault-tolerant systems. The ServerNet interprocessor communications can scale to hundreds of processors while providing scalable bandwidth and redundant paths. By connecting both processors and peripherals through ServerNet, the CPU and I/O connectivity may scale independently. The support for any-to-any communications prevents unneeded intermediate transfers that waste processor and memory resources. ServerNet also gives built-in peripheral switching capability for load-balancing and standby redundancy.

ServerNet-based systems represent a major step forward in commercial fault-tolerant architecture. For the first time, a single architecture provides several different options for hardware or software fault tolerance. It also provides additional flexibility by support for shared-memory processing and distributed clusters. Using common components, this architecture supports the design of a wide variety of systems by selecting different operating systems, processors, and fault-tolerance mechanisms.
Acknowledgments

The authors wish to acknowledge the large team of engineers who have contributed to this architecture. In particular, we wish to acknowledge Doug Jewett and Timothy Chou, who were instrumental in starting this project, and Bill Bunton, Dave Garcia, John Krause, and Stephen Low, who have contributed many key ideas.

ServerNet, Tandem, Himalaya, Integrity, NonStop, NonStop V1X, NonStop CLX, and NonStop Cyclone are trademarks of Tandem Computers Incorporated.

Motorola is a registered trademark of Motorola Incorporated. 

MIPS and R3000 are registered trademarks and R4400 is a trademark of MIPS Technologies Incorporated.

UNIX is a registered trademark in the U.S. and other countries, licensed exclusively through X/Open Company Limited.

Windows NT is a registered trademark of Microsoft Corporation.

References


