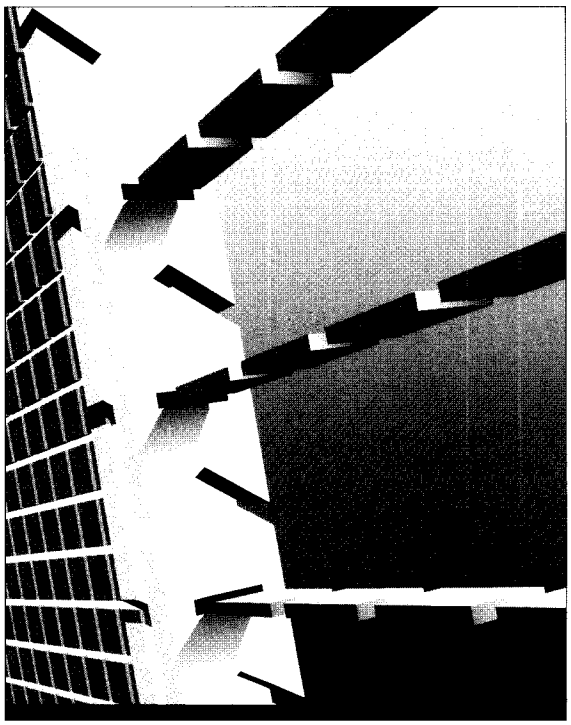
T A N D E M

SYSTEMS REVIEW

VOLUME 9, NUMBER 4 FALL 1993



NonStop NET/MASTER

Systems Management Program

Product Update

Index

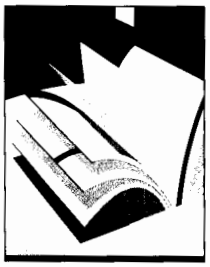
T A N D E M

SYSTEMS REVIEW

VOLUME 9, NUMBER 4

FALL 1993

TANDEM



Editor's Note

The four articles in this *Tandem Systems Review* focus on systems management. The three articles by Stockton are devoted to various aspects of NonStop NET/MASTER (NNM), an operations management tool that provides console services for Tandem operators, an environment for automated operations, and network connectivity for managing Tandem and IBM systems. These articles describe NNM's event management architecture; provide guidelines for configuring NNM and optimizing performance; and present event-processing costs, sizing calculations, and memory and disk space requirements.

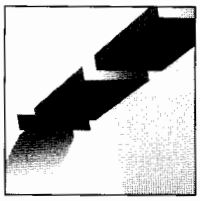
The concluding article by Dagenais describes a case study in which an OM organization carried out a systems management improvement program. By following a sequence of planned steps, the OM organization effectively introduced tools such as automated operations software and improved their systems management processes. They were able to reduce outages, raise the quality of services delivered to end users, enhance OM productivity, and build the foundation for future improvements.

Included with this issue is a brief form requesting address confirmation for existing subscribers and contacts for new readers. We would sincerely appreciate your providing the appropriate information and returning the form at your earliest convenience.

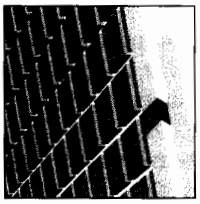
–AL

Tandem Systems Review is published quarterly by Tandem Computers Incorporated. All correspondence and subscriptions should be addressed to *Tandem Systems Review*, 10400 Ridgeview Court, Loc 208-65, Cupertino, CA 95014.

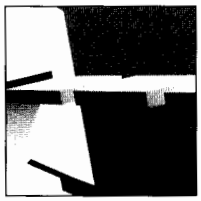
Subscriptions: $75.00 per year; single copies are $20.00. Detailed subscription information is provided on the subscription order form at the end of this book.



SYSTEMS MANAGEMENT



DEPARTMENTS



Product

Systems Products

NonStop Himalaya Range

July 1993

The NonStop Himalaya range comprises a new class of open, reliable, parallel-processing servers. These servers deliver outstanding price/performance for a broad range of online transaction processing, client/server, messaging, and decision support applications. The Himalaya range consists of three server series, the K100 series, the K1000 series, and the K10000 series.

The K100 servers are entry-level parallel servers available with CISC or RISC processors. The K100 CISC processor comes standard with 16 megabytes of onboard error-correcting memory, 192 kilobytes of cache, and dual interprocessor buses. The K100 RISC processor comes with 32 megabytes of onboard memory, 512 kilobytes of cache, and dual interprocessor buses. From its smallest to its largest configuration the Himalaya K100 server offers a seven-fold performance growth range.

The K1000 servers constitute a cost-effective platform for applications requiring midrange performance and a wide range of scalability. These servers combine the use of RISC processors with Tandem's parallel system architecture to provide all of the benefits of NonStop systems at excellent price/performance levels. Each K1000 processor comes with 32 megabytes of onboard error-correcting memory (expandable to 128 megabytes), high-speed cache memory of 512 kilobytes, and dual interprocessor buses.

The K10000 servers are Tandem's most powerful, massively parallel servers designed for high-volume commercial applications. The K10000 has a scalability of from 2 to over 4000 processors and uses a RISC processor based on the MIPS R4400 microprocessor technology. Each K10000 processor comes with 64 megabytes of onboard error-correcting memory (expandable to 256 megabytes), high-speed secondary cache memory of 4 megabytes, and dual interprocessor buses.

The Himalaya servers are fully application compatible with each other and with existing NonStop systems. The servers can be installed and serviced by nontechnical users. System features that support remote access for diagnostics and repair are standard. For maximum configuration flexibility, the Himalaya servers use a compact system design that minimizes floor space requirements and allows operation in a noncomputer-room environment.

Information Access Servers

July 1993

Tandem's Information Access Servers, based on the Himalaya K100 and K10000 servers, provide a range of capabilities aimed at information access applications, with excellent price/performance. The Information Access Servers use RISC processors and come packaged with disk storage devices providing from 17 to 208 gigabytes of usable storage. Each server also comes with the NonStop Kernel operating system, Tandem's NonStop SQL relational database management system, and open data access gateways.

Most parts of the Information Access Servers, including processor, I/O controller boards, devices, power supplies, and fans, are designed to be installed and serviced by nontechnical users and without tools. Virtually all of the field-replaceable units (FRUs) can be installed and serviced online without disrupting normal operation of the server. The servers also include, as standard, system features that support remote access for diagnostics and repair.

Operating System Software

Tandem NonStop Kernel

July 1993

The Tandem NonStop Kernel, Tandem's next generation of operating system software, provides new features and benefits to support open, reliable, highly scalable computing with lower costs of ownership and management. The NonStop Kernel is a new, layered operating system that consists of low-level functions such as interprocess communication; I/O interface procedures; and memory, time, and process management. These core services, in turn, support a number of system services that manage different application program interfaces and unique Tandem features. The system services provide fault tolerance and data integrity and perform system tasks on behalf of user programs. These services include formatting, I/O drivers support, performance measurement, NonStop process-pair support, file management, standard security, support for single-system image, and low-level transaction management. These services have been designed to increase the capacity of the operating system and allow users to execute more transactions and configure more devices on their Tandem systems.

The new operating system architecture improves system reliability and scalability by extending process, device, and concurrent message limits. Key areas of the operating system have been restructured to simplify the access to low-level functions and improve cross-product coordination. These changes improve the overall reliability and availability of the operating system. They also reduce the time and effort needed to introduce future products and to support open environments and client services.

The NonStop Kernel is the foundation on which all open systems development is based. The NonStop Kernel is vital to establishing open interfaces, enabling heterogeneous system interoperability, and allowing applications to port to NonStop systems. The NonStop Kernel provides client/server-based transaction services and extends the Tandem architecture to massively parallel commercial computing environments. This new functionality supports the application interfaces, formats, and protocols required to implement formal and de facto industry standards and to provide gateways for easy access to Tandem fundamentals.

Client/Server Computing Products

Windows NT

June 1993

Tandem will offer Windows NT on its PSX and NDX platforms when Microsoft makes this operating system available for delivery. Windows NT provides full 32-bit operations, preemptive multitasking, advanced security and reliability, built-in networking, and complete scalability across hardware platforms for users of powerful PCs, workstations, and network servers.

Communications and Networking Products

Tandem Message Integrator

September 1993

The Tandem Message Integrator is a new system application solution package for integration of electronic mail with NonStop systems. The Message Integrator combines a state-of-the-art E-mail integration software product with the NonStop Himalaya range servers to offer a standards-based, reliable alternative to proprietary electronic mail integration.

Built on an advanced messaging architecture, the Message Integrator automatically synchronizes user directories, converts office system objects, and performs multinational character set translation between multiple electronic mail systems. The operations are fully transparent to end users, who continue to use the mail formats and addressing conventions of their local systems. The Message Integrator supports the interoperation of major electronic mail systems, including IBM OfficeVision/VM (PROFS), IBM OfficeVision/400, DEC All-IN-1, HP DeskManager, MicroSoft Mail, and cc:Mail.

SNAX/APN Software

June 1993

SNAX/APN software provides added flexibility in connecting Tandem NonStop systems in SNA networks. Based on the Low-Entry Networking (LEN) Node definition of IBM's Advanced Peer-to-Peer Networking (APPN) standard, SNAX/APN enables users to connect NonStop systems to a wide range of IBM computers. These include AS/400 systems, ES/9000 hosts, and other system platforms that support the APPN architecture. SNAX/APN offers all the advantages of APPN architecture, including parallel session support and dynamic session configuration, while preserving users' existing investments in Tandem and IBM Logical Unit (LU) 6.2 applications.

SNAX/APC, Release 3

June 1993

Release 3 of Tandem's SNAX Advanced Program Communication (SNAX/APC) software incorporates a number of new features designed to make the product easier to use and more efficient. The new features include parallel session support; support for Mapped Conversation and Change Number of Sessions (CNOS) verbs; improved online manageability; full Distributed Systems Management (DSM) support; and improved utilities.

Tandem OSI/MHS, Release 3.0 and OSI/GPI, Release 2.0

July 1993

Release 3.0 of Tandem's OSI/MHS (OSI Message Handling System) and Release 2.0 of OSI/GPI (an XAPIA standard Application Program Interface to X.400 services) offer a number of new features. These include closed user groups; distribution list support; enhanced manageability; significant increase in configuration limits; and special enhancements for Japanese markets.

In addition, Tandem now supports X.435 protocols in OSI/MHS. X.435 is an advanced client/server protocol for the merger of Electronic Data Interchange (EDI) and electronic mail technologies over X.400 backbone services. X.435 allows trading partners using electronic commerce applications to manage communications more efficiently and securely.

16-Bit Network Adapter Card

July 1993

The 16-bit network adapter card from Ungermann-Bass is a software-configurable network interface card optimized for use with Ethernet networking environments. This new adapter card supports a wide variety of industry-standard drivers, including NetWare and LAN Manager. Because the new Ethernet card is fully software configurable, it provides a more efficient and fail-safe method of configuration than conventional jumper/switch methods.

LAN Manager 2.2

July 1993

LAN Manager 2.2, a new and improved version of LAN Manager, makes it easier for network administrators to manage growing, complex networks. LAN Manager is a network operation system developed by Microsoft and enhanced with services by Ungermann-Bass.

The new features of LAN Manager 2.2 include network administration for Windows; interoperability with Windows for workgroups; Windows network applications starter; LAN Manager print station; TCP/IP extensions; enhanced IBM connectivity; and Data Link Control (DLC) protocol support. In addition, value-added features provided by Ungermann-Bass include printer services, DOS remote boot service, and statistics logger and reporter.

3615 Ethernet Controller

May 1993

The 3615 is a new, high-performance Ethernet controller. It is significantly faster than the 3611 and 3613 models it replaces and has the potential for providing increased maximum throughput for TCP/IP and other LAN protocols. The 3615 controller supports multiple protocols and communications operations simultaneously, including Expand over LAN, Multilan, OSI/TS, TCP/IP, AppleTalk, customer-written native Tandem LAN Access Method (TLAM) protocols, and third-party protocols.

MasterLAN ISA Network Adapter Card

June 1993

The MasterLAN ISA card is a high-performance, 16-bit network adapter utilizing bus master architecture for superior performance in Ethernet networking environments. This new Ethernet adapter supports connectivity via twisted-pair wiring (RJ-45) and standard Ethernet coaxial cable (AUI).

EISA 32-Bit Network Adapter Card

The new EISA network adapter card from Ungermann-Bass is an Ethernet adapter that delivers full 32-bit performance for servers or high-speed workstations. Driver support is provided for major network operating systems including NetWare, LAN Manager, and SCO UNIX. The new Ethernet adapter is fully software-configurable for quick and easy installation.

EtherNext MicroHub

June 1993

The EtherNext MicroHub from Ungermann-Bass provides the ultimate in 10BASE-T connectivity and performance. The MicroHub's small size, combined with its high-powered 10BASE-T performance, makes it ideal for small LAN applications needing full 10BASE-T power, or as a workgroup concentrator for a subsystem in enterprise-wide LANs.

Series 4000 Family

June 1993

The Series 4000 family of products fulfills the requirement for efficient delivery of network applications. Each Series 4000 component, including hub chassis, Ethernet host modules, Token Ring modules, and management modules, is designed to maximize networking performance. Through its modular components, the Series 4000 family allows users to customize solutions that best meet their particular networking needs.

TCPnext

June 1993

TCPnext is the new release (16.62) of the Ungermann-Bass TCP/IP protocol stack. Release 16.62 frees up conventional RAM on a DOS or Windows workstation through its UMB and HMA support, enhanced memory management, and enhanced configuration options. TCPnext provides users with a standards-based mechanism for developing mission-critical applications. It also offers up to 254 NetBIOS sessions per stack.

Tools and Utilities

Micro Focus COBOL Workbench

May 1993

Micro Focus COBOL Workbench, now available for Tandem systems, enables programmers to develop and maintain COBOL85 and SCREEN COBOL programs on any PC running under MS-DOS, Windows, or OS/2. Using the workbench software, programmers can edit, compile, execute, and debug applications, including Pathway applications, on their PCs.

The Micro Focus COBOL Workbench provides significant productivity and quality gains. These gains are achieved through a set of integrated advanced tools; faster compilation; an improved user interface; freeing of host resources for production usage; faster, predictable execution times; and developer portability.

Printer Products

5575 Laser Printer

June 1993

The 5575 laser printer is a 600-dpi, 8-ppm printer that uses Resolution Enhancement technology and microfine toner to produce outstanding print quality. It is designed for distributed use by small departments and work groups and is suitable for print volumes of up to 20,000 pages per month. The 5575 printer supports industry-standard PCL 5E and prints graphics faster with the use of a RISC processor. It also supports Adobe PostScript Level 2, and automatically switches between the PCL and PostScript languages.

The printer comes with 2 megabytes of standard memory, expandable to 32 megabytes. It has a total of 45 internal scalable typefaces in both Intellifont and TrueType formats. The 5575 employs the Transparent Tandem Asynchronous Protocol (T-TAP) interface to ensure that printing is completed correctly, even if the printer runs out of paper or if a power outage occurs. This printer is compatible with all NonStop systems, and can also be connected to Integrity systems, PCs, and PC LANs.

TCP/IP LAN Print Spooler

August 1993

TCP/IP LAN Print Spooler enables users to print to a wide variety of printers connected to a TCP/IP network. With this product, users can even print from their Tandem NonStop server to printers attached to UNIX and other platforms, including Macintosh and OS/2, that can serve as Line Print Daemons. TCP/IP LAN Print Spooler is a print process that works with the user's Tandem NonStop server's Spooler. It sends print jobs to LAN-based printers via Tandem TCP/IP, Tandem LAN Access Method (TLAM), and an Ethernet controller.

X.25 WAN Print Spooler

August 1993

X.25 WAN Print Spooler allows users to print files from their Tandem host to printers located on X.25 WANs. X.25 WAN Print Spooler permits the use of X.25 switched virtual circuits rather than permanent virtual circuits. This means that a printer can be accessed by multiple circuits. If one circuit is busy, another can be used to reach the printer for maximum printer availability. It also means that a small number of circuits can service a large pool of printers for minimum communications costs. X.25 WAN Print Spooler is a print process that works with the user's Tandem NonStop server's Spooler. It sends print jobs to WAN-based printers via X.25 Access Method and a synchronous controller.

XNS LAN Print Spooler

August 1993

XNS LAN Print Spooler lets users send print jobs from their Tandem system to Xerox high-speed printers located on an Ethernet LAN. Users can locate their Xerox printer remotely and access the printer from any Tandem NonStop system in the LAN/WAN environment. In addition to offering remote printer command and control, XNS LAN Print Spooler provides data integrity and print data error correction. It thus makes printing faster and more CPU efficient. XNS LAN Print Spooler is a print process that works with the Tandem NonStop server's Spooler and sends print jobs to the Xerox printer via Tandem LAN Access Method (TLAM), an Ethernet controller, and an Ethernet cable or LAN. Compatible printers include the 4050, 4850, 4090, 4890, 4135, 8790, and 9790.

Workstation and Terminal Products

High-Capacity Tape Drives for NDX ST-Series Servers

June 1993

Tandem now offers two new, high-capacity tape drives for its NDX ST-series servers, the 4-GB Digital Audio Tape (DAT) drive and the 1-GB Quarter Inch Cartridge (QIC) drive. The 4-GB DAT drive provides offline data storage of up to 4 gigabytes (8 gigabytes compressed). In addition to the high storage capacity, the new DAT drive provides high reliability through read-after-write, third-level error detection and correction circuitry. The drive uses data compression to produce an increase in capacity of 2 to 4 times over a standard DAT drive.

The 1-GB QIC drive provides offline data storage of up to 1 gigabyte. With its error detection and correction capability and an MTBF of 80,000 hours, this drive is particularly suitable for users who require a reliable, low-cost, data storage device. Each tape drive is shipped with a tape cartridge and Sytos Plus software.

MS-DOS 6.0 International Upgrades

August 1993

Tandem now offers MS-DOS 6.0 upgrades to its existing international users. German, French, and USASCII versions are now available in Europe. All international language upgrades are available in 3.5-inch media only.

Storage Products

5190 Cartridge Tape Subsystem

July 1993

The 5190 is a new, lower-cost, high-performance cartridge tape subsystem for NonStop Himalaya K100, Himalaya K1000, Himalaya K10000, Cyclone, and CLX 800 systems. The 5190 is fully compatible with the following cartridge tape subsystems: Tandem 5180, STK 4480, IBM 3480, and 18-track models of the IBM 3490.

The 5190 is a compact, customer replaceable unit (CRU) consisting of the formatter, one cartridge tape drive, power supply, and the optional seven-cartridge automated cartridge loader (ACL). One or two 5190 CRUs can be installed in a single Tandem Modular Storage System module. The 5190 ACL uses an internally mounted mechanism that eliminates the external cartridge magazine common to other products in its class. This makes it possible to install up to three 5190 modules in a single Modular Storage System stack. A single stack can thus contain up to six 5190 subsystems in only 6.3 square feet of floor space.

Product Programs and Services

Tandem System Access Guarantee

June 1993

Tandem System Access Guarantee is a Tandem support service that guarantees user access to an operational Tandem system in the event of a disaster. An important feature of the Tandem System Access Guarantee is a pre-service technical review to ensure that Tandem's disaster recovery system will meet the user's disaster recovery needs. Once the suitability of Tandem's predefined disaster recovery system is established, the service guarantees that an operational Tandem system with operating software will be installed and ready for application loading where and when the user needs it. The service also provides use of the installed operational Tandem system for 30 continuous days. If the system is needed longer, Tandem will arrange for rental of the system beyond the initial 30-day period.

NonStop NET/MASTER: Event Management Architecture

The new Tandem™ NonStop™ NET/MASTER (NNM) product is an operations management tool that provides console services for Tandem operators, an environment for automated operations, and network connectivity for managing Tandem and IBM systems. NNM's operations management functions include event monitoring, security services, and command processing. NNM can be installed on a single Tandem node or on multiple systems for event management in a multinode environment.

Two important features of NNM for automated operations are its Network Control Language (NCL) interpreter and its Rules Management Services (RMS) product. NCL is a scripting language for writing operations management procedures. Using NCL, a user can write customized management procedures for execution under NNM. RMS provides a set of automated recovery rules and an environment for their execution. The rules are predefined NCL procedures for the automated recovery of Tandem components such as Pathway terminals, Expand™ network communication lines, and CPUs.

This article presents an overview of the design and implementation of NNM event management functions and describes automated recovery under RMS. It provides background information necessary for approaching two further articles on NNM in this issue of the *Tandem Systems Review*: "NonStop NET/MASTER: Configuration and Performance Guidelines" (Stockton, 1993a) and "NonStop NET/MASTER: Event Processing Costs and Sizing Calculations" (Stockton, 1993b). All three articles apply specifically to NonStop NET/MASTER Release 1.5. Readers of this article should possess a basic knowledge of Tandem systems and have some familiarity with the Tandem Subsystem Programmatic Interface (SPI) message format and the Tandem Event Management Service (EMS) product.

Design and Implementation of NNM Event Management Functions

NNM is designed to provide basic event management functions:

- Receive and evaluate events.
- Forward events to local NNM modules and to remote systems.
- Display event information to an operator.
- Log events.
- Initiate automated operations procedures.

In addition, NNM provides security functions, memory management services, and procedures for converting event messages from Subsystem Programmatic Interface (SPI) format to Mapped Data Object (MDO) format. MDO is NNM's internal formatting structure for the evaluation and manipulation of event messages.

NNM implements the preceding features by providing a core product and by giving the user a means to develop additional customized functions. The core product is written in the C programming language and provides basic functions such as event receipt, conversion of SPI format to MDO format, interprocess communication, multitask management, and terminal and file I/O. Functions in the core product are packaged in cohesive program modules and made available in two configuration modes (see "Configuration Modes," later in this article.).

NNM consists of several logical processing areas called regions. Within each region, individual tasks, or threads, execute to perform NNM functions such as logging an event or displaying it to an operator. This article focuses on

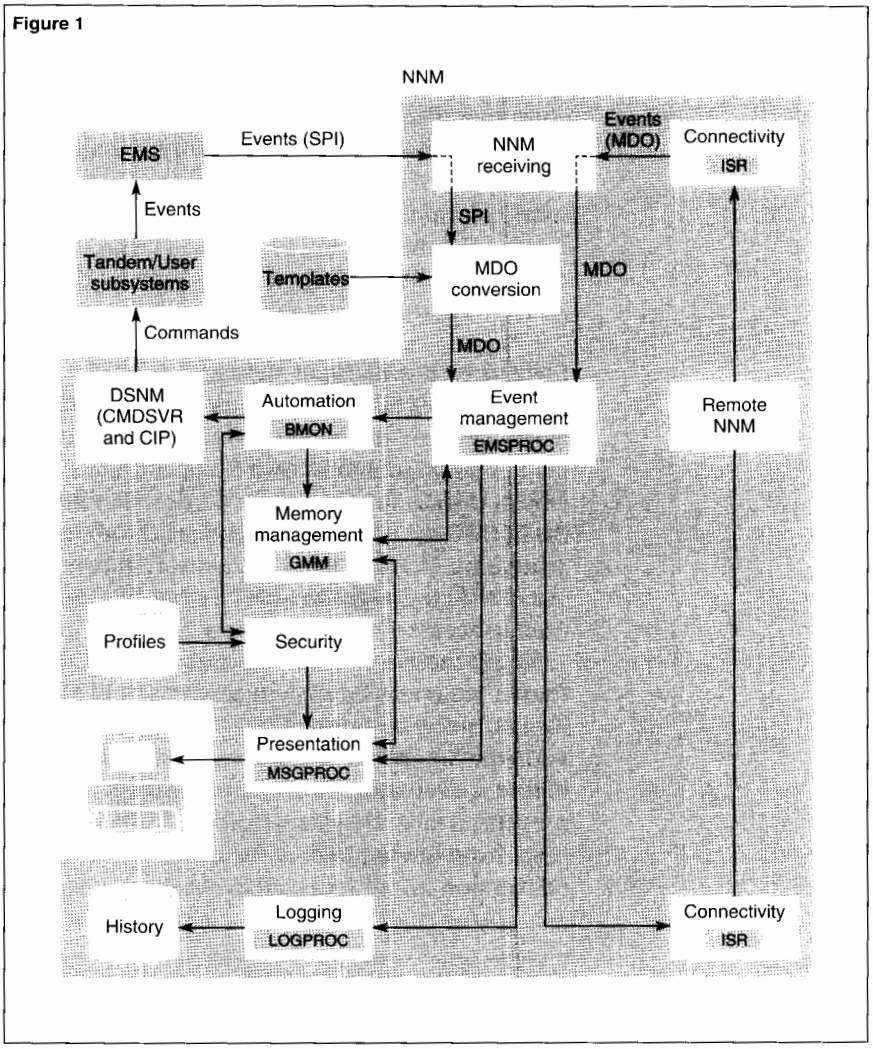
- The EMSPROC task, which runs in the EMSP region.
- The LOGPROC task, which runs in the LOGP region.
- MSGPROC tasks, which run under an Operator Control Services (OCS) region.
- The Background Monitor (BMON) region, which provides asynchronous, background execution for NCL procedures.

Using NCL, users can write procedures, or scripts, to customize NNM functions. For event-driven execution, such procedures can be loaded into the appropriate region to execute as the EMSPROC task, the LOGPROC task, or a MSGPROC task. For asynchronous, background execution, NCL tasks can be submitted to one of three *background* regions: BMON, BLOG, or BSYS.[1] In addition, an operator can invoke NCL procedures directly for immediate execution.

Figure 1 illustrates the design of event management functions in NNM. It is discussed in the following section.

[1] BLOG and BSYS are not discussed in this article. For further information, see the *NonStop NET/MASTER Management Services (MS) Operator's Guide* (1992).

Figure 1



Event Management Components of NNM

In Figure 1, elements in the shaded area are components of NNM. Within NNM, unshaded boxes represent functional components. The shaded boxes within functional components identify elements in the implementation of NNM. The following discussion covers the major functional components represented in Figure 1.

Figure 1.
Design of NNM event management functions.

Receiving Events. In Figure 1, NNM can receive events from either EMS or from a remote NNM system. NNM receives EMS events from its EMS consumer distributor. The user can write a filter for the distributor so that it only forwards designated types of events to NNM. When NNM receives events from remote NNM systems, the events are sent over Inter-NET/MASTER Connection (INMC) links using the SNA, X.25, or Expand process-to-process network protocols.

Conversion of EMS Event Messages to MDO Format. When NNM initially receives an EMS event message, it is in SPI format. NNM converts the SPI message into its own Mapped Data Object (MDO) format and forwards the event to the EMSPROC task for evaluation and further processing. As illustrated in Figure 1, in converting SPI messages to MDO format, NNM may need to access a template file on disk.[2]

Event messages received over an INMC link from a remote NNM system are already in MDO format, since the originating NNM system converts the SPI events it receives to MDO format before it evaluates them for forwarding. This prevents a central system with multiple INMC links from having to bear the costs of MDO conversion for all received event messages.

[2]Outside of NNM, the conversion of SPI messages to text messages is usually carried out through the use of templates maintained in system memory. In some cases, there may be nonresident templates, which are stored in a template file on disk. The templates used for SPI-to-text conversion are also used by NNM for SPI-to-MDO conversion. When nonresident templates are necessary for conversion to MDO, NNM must access them from disk.

Event Processing (EMSPROC). Once an event has been received and converted to MDO format, it is forwarded to the EMSPROC task. If RMS or a user-written NCL procedure executes as the EMSPROC task, it can apply evaluation logic to determine whether or not to forward the event to other NNM tasks. In forwarding an event, the evaluation logic can

- Send the event only to the presentation (MSGPROC) and logging (LOGPROC) tasks.
- Forward the event only to another NNM node.
- Forward an event both to the MSGPROC and LOGPROC tasks and to another NNM node.
- Submit RMS or user-written NCL procedures to the BMON, BLOG, or BSYS regions for background execution.

Automation. NNM supports automated operations. As an important part of this, it provides RMS, a set of NCL procedures and management functions for evaluating events and initiating recovery actions (see "RMS Automated Recovery," later in this article). RMS procedures issue commands to subsystem objects through the Tandem Distributed Systems Network Management (DSNM) product (described later), which is supplied with NNM. In addition, users can write their own NCL procedures to provide automated-operations logic specific to the needs of their site. These procedures can also make use of the DSNM command interface.

The automated recovery functions provided by RMS can be configured in the EMSPROC task, the LOGPROC task, or a MSGPROC task for event-driven execution, or they can be configured in the BMON, BSYS, or BLOG region for background execution.

As shown in Figure 1, automated recovery makes use of NNM's memory management and security components. The memory management facility maintains status information on components during recovery procedures. Security is provided through NNM User ID/Management Services (UMS) profiles and the SECEXIT process.

Memory Management (Global Memory Manager). NNM provides memory management services for information that may need to be shared across NNM tasks or processes. Memory management services are implemented in NNM's Global Memory Manager (GMM) processes.

Users can configure multiple GMM processes, each of which contributes approximately one megabyte of global memory that is available to all NNM processes and user-written NCL. Global memory maintains status information on components for which RMS recovery procedures have been initiated. It also retains critical event messages, called non-roll-delete (NRD) messages, until they are purged by an operator or meet an NNM condition for NRD deletion. Users can write NCL procedures to define and access global memory data.

Logging (LOGPROC). NNM maintains an activity log on disk for logged NNM events. Logging is provided through NNM's LOGPROC task. There can only be one LOGPROC task in an NNM system. The LOGPROC task receives event messages from other tasks, such as the EMSPROC task or MSGPROC tasks (see the next section, "Presentation"). These event messages can be stored in the NNM log file. The LOGPROC task also receives internal NNM messages, such as security actions, which it can insert into the system log. If RMS or a user-written NCL procedure is loaded into the LOGPROC task, it can apply evaluation logic to determine whether or not to log an event. If no evaluation logic is provided (LOGPROC = FLUSH), the LOGPROC task logs all events that it receives.

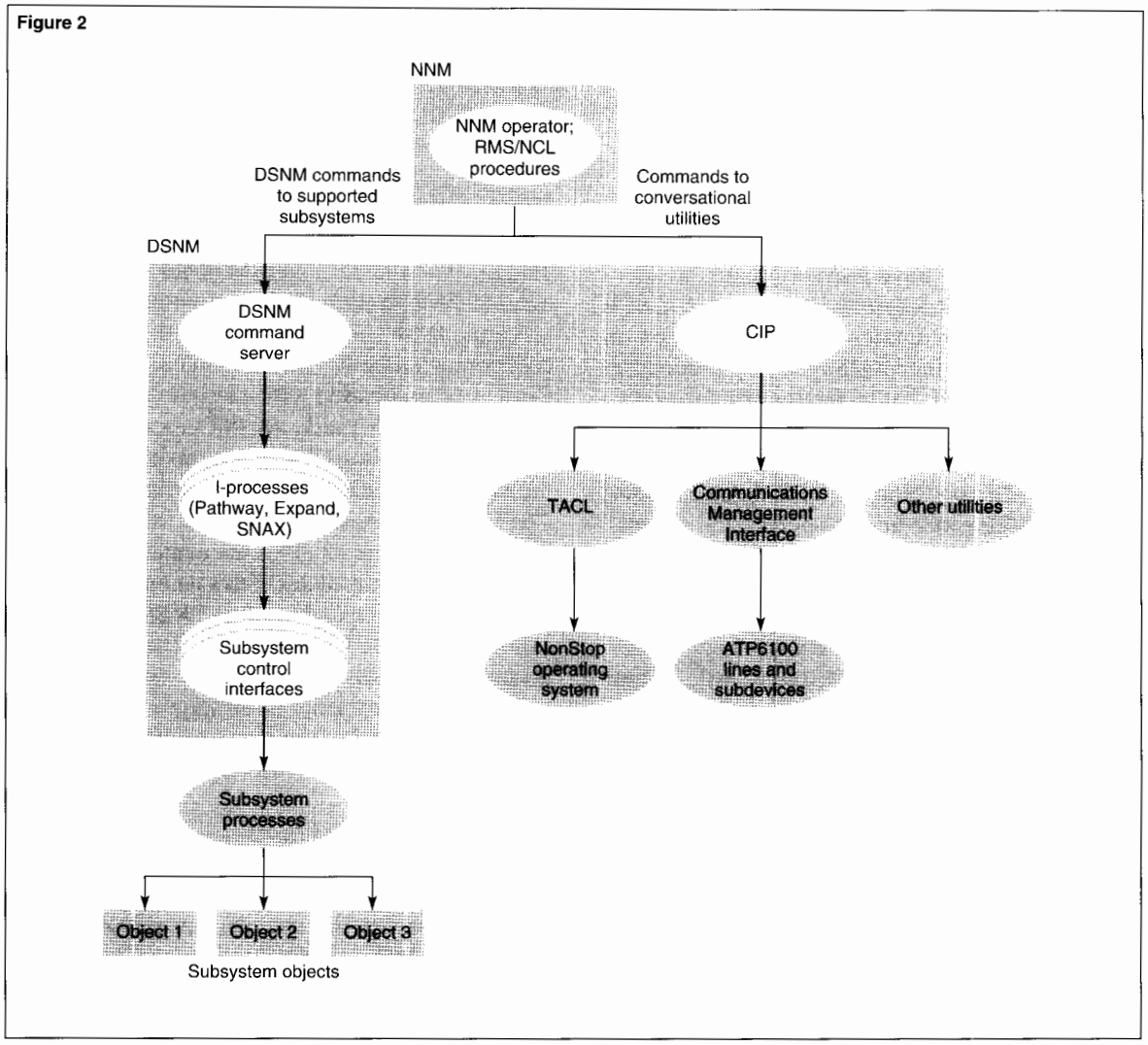
Presentation (MSGPROC). Event messages are displayed in an Operator Control Services (OCS) window. The display of event messages in an OCS window is controlled by a MSGPROC task. There can only be one MSGPROC task per OCS window. However, there may be many active OCS windows in an NNM system, each with its own MSGPROC task.

If a user-written NCL procedure is loaded into a MSGPROC task, it can apply evaluation logic to determine whether or not to display an event. The procedure can also specify display attributes, such as the use of color for highlighting different types of event messages. If no evaluation logic is provided (MSGPROC = FLUSH), the MSGPROC task passes all events that it receives and uses default display attributes.

Connectivity (ISR). Intersystem communication between Tandem NNM nodes or between Tandem NNM nodes and IBM nodes is implemented through Inter-System Routing (ISR) tasks and INMC links. An INMC link is a logical connection between two systems. For each INMC link associated with a system there is a corresponding ISR task. An ISR task receives events from the local EMSPROC task and sends the event across its INMC link to a remote system. The ISR task corresponding to the INMC link on the remote system receives the event. Events can be transported over INMC links using SNA, X.25, or the Tandem Expand product. When NNM's Remote Operator Control (ROC) component is configured, INMC links can also transport commands. (For further information on NNM's ROC component, see the *NonStop NET/MASTER Management Services [MS] Operator's Guide* [1992].)

Unlike other NNM tasks such as the LOGPROC and MSGPROC tasks, ISR tasks cannot carry out evaluation logic on individual events. However, the user can specify whether a given ISR task is to forward or receive either solicited or unsolicited events. A solicited event is an event message generated in response to an explicit request, such as the output of a command. An unsolicited event, such as the event message generated when a terminal fails, is not the result of an explicit request. Regulating the transmission of solicited and unsolicited events is discussed in "NonStop NET/MASTER: Configuration and Performance Guidelines" (Stockton, 1993a) in this issue of the *Tandem Systems Review*.

Figure 2.
DSNM command server and CIP under NNM.



DSNM Command Server and Conversational Interface Process

The Tandem Distributed Systems Network Management (DSNM) product is provided with NNM. NNM uses the DSNM Command Server (CMDSVR) interface to issue commands to Tandem subsystems such as the Pathway transaction processing subsystem and the Expand network communications product. NNM uses the DSNM conversational interface process (CIP) as an interface to system utilities, TACL™ (Tandem Advanced Command Language) routines, and third-party utilities. CIP also supports a security mechanism for controlling user access to system utilities and commands. All RMS rules use DSNM to execute automated recovery actions. User-written NCL procedures can also make use of DSNM.

DSNM processes supporting the command server and CIP are automatically started when NNM is initialized. Other DSNM processes, not automatically started by NNM and not discussed here, can be configured.[3]

[3]For further information on the DSNM product, see the *Distributed Systems Management Solutions (DSMS) System Management Guide* (1992). Information on the use of DSNM within NNM is provided in both the *NonStop NET/MASTER Management Services (MS) System Management Guide* (1993) and the *NonStop NET/MASTER Management Services (MS) Operator's Guide (1992).*

The DSNM command interface provides a set of nine commands and a simple command syntax that can be used with objects in a wide range of Tandem subsystems.[4] For example, the DSNM START, STOP, and ABORT commands all have the same syntax and can be used with any object in the Tandem subsystems supported by DSNM, even if the subsystem uses a different expression and a different syntax to achieve the same result.

DSNM commands are implemented through a command server, the DSNM CMDSVR process, and DSNM interface processes (I-processes). For every supported subsystem, DSNM maintains an I-process that converts DSNM commands and command syntax to the subsystem's own command terminology and syntax. Figure 2 illustrates the use of the DSNM command server and CIP under NNM. Elements in the shaded area are components of DSNM. RMS uses the DSNM command server for the recovery of Pathway, Expand, and SNAX™ (SNA Communication Services) objects. RMS uses CIP for the recovery of CPUs, ATP6100 lines and subdevices, and other objects.

Configuration Modes

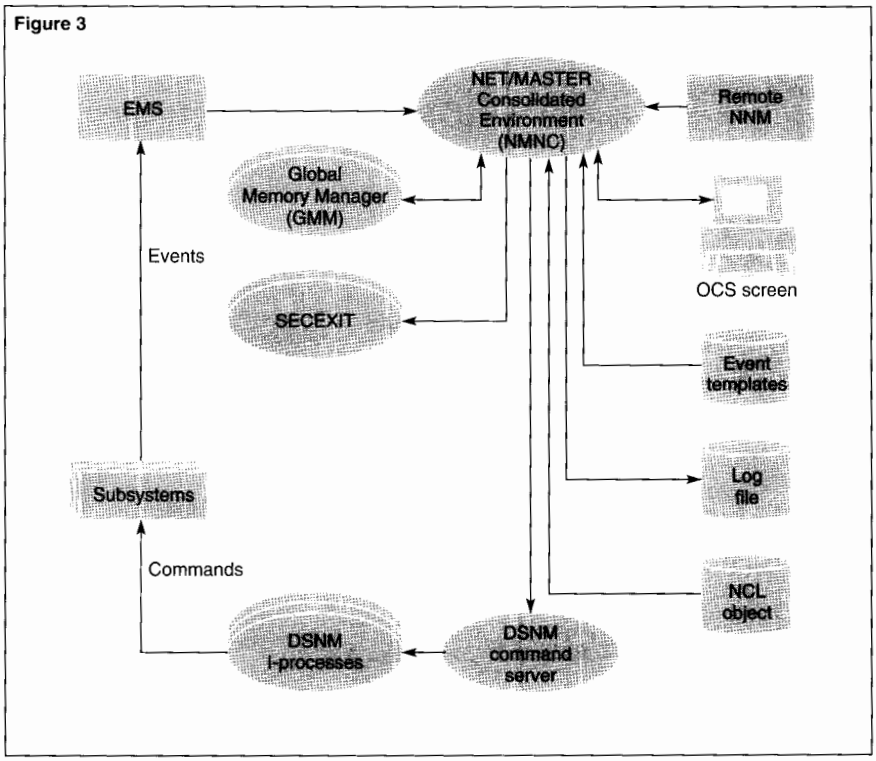
NNM packages its event management functions in two configuration options, the basic configuration mode and the advanced configuration mode. The basic configuration mode is designed for smaller sites with lower event rates and fewer operators or for satellite sites with unattended node management in a distributed NNM network. The advanced configuration mode is for sites with greater system management demands or for sites that centralize the management of a distributed network. Both configuration modes provide the same functions.

Components of the Basic Configuration Mode

In the basic configuration mode, a single process, NMNC (NNM Consolidated Environment), provides almost all event management functions.

[4]Under NNM, the DSNM command server supports the following Tandem subsystems and products: Expand, Pathway, the NonStop Kernel (only for use of the DSNM STATUS command on disks and CPUs), Spooler, SNAX/CDF, SNAX/XF, X25AM, AM3270, TR3271.

Figure 3

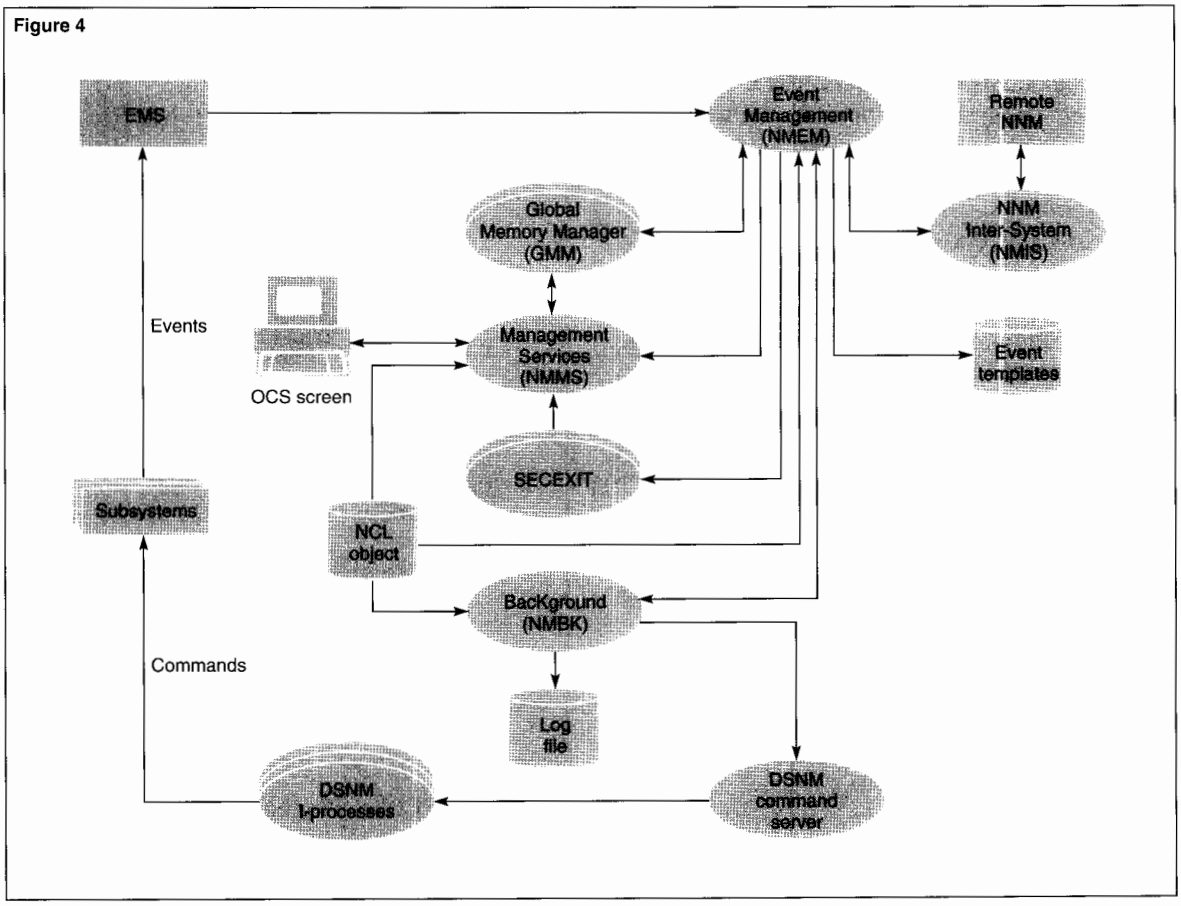


Aside from NMNC, the basic mode includes Global Memory Manager (GMM) processes, SECEXIT, the NET/MASTER Control Process (NCP), and DSNM. Figure 3 shows the structure of NNM under the basic configuration mode.

In Figure 3, the NMNC process manages event interfaces, logs events, presents them to operators, forwards them to remote systems, and executes any automated recovery actions. GMM stores unacknowledged critical messages (NRDs) and maintains the status of RMS recovery objects. The SECEXIT process provides security functions. DSNM provides interfaces for issuing commands to Tandem subsystems and interacting with NonStop Kernel utilities. Not shown in Figure 3 is the NET/MASTER Control Process. NCP acts as a system monitor that creates all NNM processes at startup and restarts NMNC and DSNM processes if there is a failure.

Figure 3.
NNM basic configuration mode.

Figure 4.
NNM advanced configuration mode.



Components of the Advanced Configuration Mode

The advanced configuration mode divides the functions in NMNC between four different processes:

- The NNM Event Management (NMEM) process receives events from ISR tasks and the EMS distributor for NNM, executes the EMSPROC task, and forwards events that pass EMSPROC filtering to other processes.
- The NNM BacKground (NMBK) process executes the LOGPROC task for logging events and any BMON functions, such as RMS recovery scripts.
- The NNM Management Services (NMMS) process provides the navigation, presentation and command interface used by NNM operators. It executes MSGPROC tasks associated with active OCS screens.
- The NNM Inter-System (NMIS) process contains the ISR task and the INMC connectivity functions.

Figure 4 shows the structure of NNM under the advanced configuration mode. The arrow from the NMBK process to the DSNM command server signifies that RMS or a user-written NCL procedure executes as the LOGPROC task or runs in a background region. Other alternatives are possible; RMS or a user-written NCL procedure that invokes DSNM commands can execute as the EMSPROC task in the NMEM process or as a MSGPROC task in the NMMS process. Not shown in Figure 4 is the NET/MASTER Control Process (NCP), which creates all NNM processes at startup and restarts the NMEM, NMBK, NMMS, NMIS, and DSNM processes if there is a failure.

RMS Automated Recovery

NNM Release 1.0 contained a set of NCL procedures, the NET/MASTER Automated Rule Set (NARS), that provided automated recovery for failed system components. In NNM Release 1.5, NARS is replaced by RMS, which contains an enhanced set of NCL procedures and a management environment. RMS supports automated recovery for

- CPUs.
- Expand lines.
- SNAX lines, SNA physical units (PUs), and SNA logical units (LUs).
- ATP6100 lines and subdevices.
- Pathway terminal control processes (TCPs) and terminals.
- TACL command interpreters.

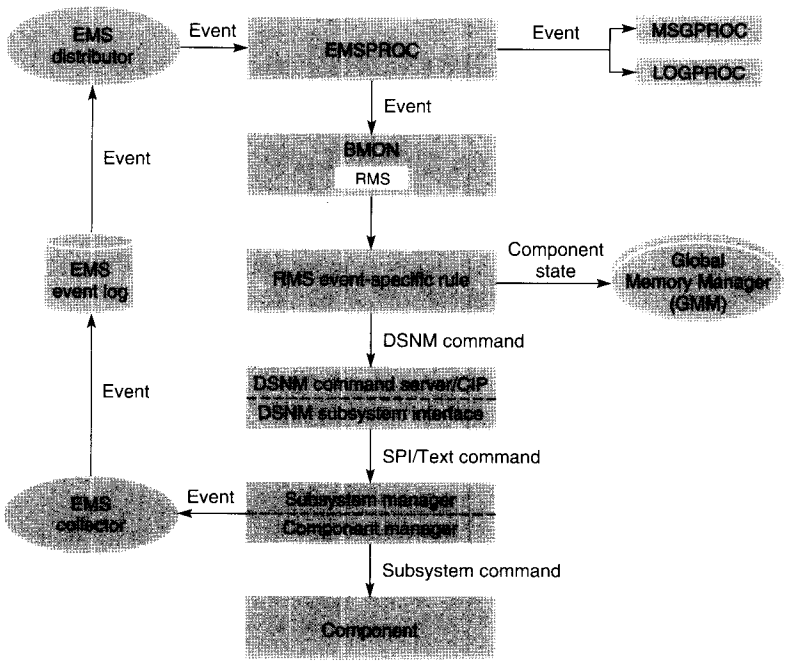
In providing automated recovery, RMS receives an event and checks to see if the event's subsystem and event number match the set of failure events it handles. If there is a match, RMS checks global memory to make sure the application of recovery rules for the failed component is not *frozen*.[5] If rule application for the component is not frozen, RMS invokes component-specific recovery actions using DSNM commands. Figure 5 shows automated recovery processing with RMS placed in the BMON region. The user has the option of executing RMS in any one of the primary NNM tasks, EMSPROC, LOGPROC, or MSGPROC, or in one of the background regions.

The following sequence of steps uses the example of a failed Pathway terminal to show automated recovery according to the diagram in Figure 5.

1. A Pathway terminal aborts and Pathway sends an event reporting the failure to an EMS collector.
2. The EMS consumer distributor for NNM gets the event from the EMS event log and forwards it to NNM, where it is converted to MDO format and evaluated in the EMSPROC task.

[5]A component is *frozen* when an operator explicitly selects the RMS *freeze* option for preventing the application of an RMS rule to a component.

Figure 5



3. EMSPROC forwards the event to other tasks; RMS receives it in the BMON region.
4. RMS evaluates the event and finds that the subsystem (Pathway) and the event number for an aborted terminal match its rule set.
5. RMS checks global memory to see if RMS recovery of the object has been frozen.
6. If RMS rule application for recovery of the object has not been frozen, RMS stores the terminal's ID and current state in global memory and initiates recovery actions.

Figure 5.
RMS recovery processing with RMS in BMON.

7. RMS submits a DSNM START command for the failed terminal to the DSNM command server.
8. The DSNM command server forwards the START command to the appropriate DSNM interface process (I-process), in this case, the Pathway I-process (PWI).
9. The Pathway I-process converts the DSNM START command into an SPI command and forwards it to PATHMON, the Pathway subsystem manager, for execution.
10. PATHMON confirms receipt of the command by generating an event reporting that the terminal was started. If the command fails, an event reporting the failure is generated.
11. The event generated at step 10 goes through EMS and the EMSPROC task back to RMS. If the restart was successful, RMS updates the state of the terminal in global memory and issues a message to OCS screens for operators to see. If the restart was not successful, RMS issues a message to OCS screens, waits 60 seconds, and if there is still no change in the terminal's state, attempts to restart it again.

RMS makes repeated recovery attempts at 60-second intervals until recovery is successful or an operator freezes the failed object. This delay can be useful.

For example, in some cases, a component repeatedly restarts and then fails within a few seconds. With each failure, RMS waits until the end of the 60-second interval. This avoids the problem of rapid recovery looping which would result from instantaneous recovery efforts.

The 60-second delay is also useful in cases where there are operational dependencies. All RMS recovery logic is rule based and is triggered by a specific combination of a subsystem ID (SSID) and an event number. There is no linkage or dependency-checking between rules and events. However, in some cases, the operation of one object is dependent on the operation of another. For example, if a Pathway terminal is running on an ATP6100 subdevice and the subdevice fails, the Pathway terminal will also fail. Each of these failures invokes a separate recovery rule.

Since each RMS rule executes independently, it is possible for the recovery rule for Pathway terminals to execute before the rule for recovery of ATP6100 subdevices. If this happens, the PATHCOM START command will arrive to restart the terminal while the ATP6100 subdevice is still down, and the recovery attempt will fail. However, 60 seconds after RMS is notified of the failed recovery attempt, it will try again. This allows time for the ATP6100 subdevice to recover, after which restarting the Pathway terminal should also be successful.

Conclusion

NNM can provide operations management functions for a single Tandem system, a network of Tandem systems, or a heterogeneous network of Tandem and IBM nodes. NNM event management functions are implemented through NCL procedures executing as the EMSPROC, LOGPROC, and MSGPROC tasks, and through NCL procedures executing under a background region such as BMON.

Users can write their own procedures for customized event management, and they can use the NNM-supplied Rules Management Services (RMS) product. RMS provides NCL procedures and management services for automated recovery.

The EMSPROC task provides the initial evaluation of events received by NNM and forwards events to other tasks. The LOGPROC task filters events received from the EMSPROC task and logs them. MSGPROC tasks filter the events they receive, customize video attributes, and display them on individual OCS screens. The NNM background regions provide environments under which NCL procedures can execute as asynchronous background tasks. Operators can manually initiate NCL procedures for immediate execution.

References

Distributed Systems Management Solutions (DSMS) System Management Guide. 1992. Tandem Computers Incorporated. Part no. 68090.

NonStop NET/MASTER Management Services (MS) Operator's Guide. 1992. Tandem Computers Incorporated. Part no. 57247.

NonStop NET/MASTER Management Services (MS) System Management Guide. 1993. Tandem Computers Incorporated. Part no. 95565.

Stockton, M. 1993a. NonStop NET/MASTER: Configuration and Performance Guidelines. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Computers Incorporated. Part no. 89807.

Stockton, M. 1993b. NonStop NET/MASTER: Event Processing Costs and Sizing Calculations. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Computers Incorporated. Part no. 89807.

Acknowledgments

Thanks are due to the entire NET/MASTER group. Special acknowledgments are given to Tony Bond, Nick Cabell, Jim Collins, Larry English, Rod Falck, and, most emphatically, Ted Hollman, for their understanding of NNM internals and design and their contributions in reviewing this article.

Mark Stockton is a principal of the consulting firm Zen Systems. He has worked with Tandem as a consultant for over eight years. Initially, Mark worked with Tandem developers on the requirements, definition, and design of the DSM products. In 1989, he began working with the DSM Applications group as a performance consultant. In conjunction with the DSM Applications group, for the past two years Mark has worked on evaluating the performance of NNM.

NonStop NET/MASTER: Configuration and Performance Guidelines

The Tandem™ NonStop™ NET/MASTER (NNM) product provides a comprehensive basis for operations management. However, configuring its many features and functions, specifying appropriate parameter values, and ensuring that system resources are available for optimal performance requires careful planning and evaluation. This article discusses the considerations involved and provides guidelines for configuring NNM, setting its parameters, and managing NNM memory requirements. A further article in this issue of the *Tandem Systems Review*, "NonStop NET/MASTER: Event Processing Costs and Sizing Calculations" (Stockton, 1993b), describes how to evaluate the CPU, memory, and disk space resources necessary for using NNM at an individual site.

This article assumes familiarity with the design and use of NNM. For a discussion of the design of NNM, see the companion paper in this issue of the *Tandem Systems Review*, "NonStop NET/MASTER: Event Management Architecture" (Stockton, 1993a). The guidelines presented in this article apply specifically to NNM Release 1.5.

Configuration Modes

In planning for the installation of NNM, one of the first decisions that must be made is the choice of a configuration mode. NNM supports two primary configuration options, a basic mode and an advanced mode. The basic configuration mode combines most of the features of NNM into a single NMNC (NNM Consolidated Environment) program. This configuration streamlines the execution of NNM, simplifies the management of NNM by reducing the number of processes necessary, and requires a minimum of system resources. However, because almost all major functions are packaged in a single program, NNM throughput and performance are more constrained than in the advanced configuration mode.

The advanced configuration mode places the major NNM functions in four programs. The NMEM (NNM Event Management) program supports the EMSP region and the EMSPROC task. The LOGPROC task and the background regions, BMON, BSYS, and BLOG, are packaged in the NMBK (NNM BacKground) program. Presentation services, command processing, and MSGPROC tasks are configured within the NMMS (NNM Management Services) program. The INMC (Inter-NET/MASTER Connection) and ISR (Inter-System Routing) functions are in the NMIS (NNM Inter-System) program. The two configuration modes are described in greater detail in "NonStop NET/MASTER: Event Management Architecture" (Stockton, 1993a).

There can only be one active EMSPROC task and one active LOGPROC task in either configuration mode. There can also only be one active instance of each of the background regions. Although users can configure multiple NMNC processes in the basic configuration mode, or multiple NMEM and NMBK processes in the advanced configuration mode, the additional EMSPROC tasks, LOGPROG tasks, or background regions will not be used.

Each connection with another NNM system is supported by a process containing the INMC and ISR functions. If the basic configuration mode is used with only one link to a remote NNM system, these functions are contained in an initial NMNC process. If there are links to more than one NNM system, multiple NMNC processes provide these services. If additional links to other systems are defined, additional NMNC processes will provide these services. When the advanced configuration mode is used, each intersystem link is serviced by a separate NMIS process.

A single NMNC or NMMS process can usually provide presentation services for several operators. Additional NMNC or NMMS processes can be configured to support more operators.

The basic configuration mode with a single NMNC process requires less disk space and memory than the advanced configuration mode. However, because event management functions are concentrated in the NMNC process, there are also more likely to be bottlenecks in the basic mode than in the advanced mode. For example, with high event rates and multiple operators, there are likely to be slowed response times as operator actions contend with the flow of events through the EMSPROC task. In the advanced configuration mode, this type of contention is of less concern, since these functions are in separate processes that can execute in different CPUs.

In general, the basic configuration mode is intended for smaller sites with low event rates and few operators. The basic mode is likely to be a good choice on an unattended site or a satellite node in a distributed network. The advanced configuration mode is likely to provide significant performance gains when there is frequent use of automated recovery procedures, two or more INMC links need to be supported, event rates are high, or more than one process for supporting operator activities is called for. A characteristic example is a central node that manages multiple satellite nodes.

The number of operators a single NMMS or NMNC process can effectively support depends on the rate at which events are displayed, how often the operators need to issue commands, and the types of commands they issue. If too many operators are serviced by the same process, response times will suffer. As a rule of thumb, plan on four operators per process, although in some cases a single process can support twice this number.

Event Filtering

Much of the cost of operating NNM is associated with processing events. First, system events that pass the Event Management Service (EMS) consumer distributor for NNM are converted to NNM's Mapped Data Object (MDO) format. Next, the EMSPROC task evaluates every event that reaches NNM and determines whether or not to pass it to other NNM tasks. These tasks evaluate the events that reach them and determine whether to carry out further actions. Additional processing costs are incurred as an event is logged, displayed to operators, used in automated recovery procedures, or forwarded to remote systems. Finally, in both the advanced and basic configuration modes, every event adds to the message traffic between NNM components. The filtering and forwarding of events is often the most critical determinant of NNM performance. The next two subsections offer some guidelines.

Filtering Events Before They Reach NNM

The cost of filtering events in NNM is an order of magnitude greater than the cost of filtering at an EMS consumer distributor. In addition, before events can be filtered within NNM, there are further costs for receiving and converting them to MDO format. The first guideline for reducing filtering costs is to use the EMS consumer distributor for NNM to filter out unnecessary events before they reach NNM.

Filtering Within the EMSPROC Task

All events that arrive at NNM must go through the EMSPROC task. If a user-written Network Control Language (NCL) procedure is executing as the EMSPROC task, it may contain a number of tests, or filtering conditions, to determine whether an event is to be deleted, submitted to a background region, or sent to other NNM tasks. The more filtering conditions there are, the more CPU resources are needed for the EMSPROC task to evaluate an event (see "Calculating Filter Costs," Stockton [1993b]). To prevent the EMSPROC task from becoming a bottleneck, the instruction pathlength should be kept to a minimum.

A major consequence of this is that in most cases, user-written filtering logic for logging and Operator Control Services (OCS) display should be placed in NCL procedures that execute as LOGPROC or MSGPROC tasks, rather than in a procedure that executes as the EMSPROC task.

However, in some cases it is useful or necessary to have filtering conditions in the EMSPROC task. It is necessary if events need to be evaluated for forwarding to a remote NNM system, because the ISR task cannot carry out its own filtering. It is also necessary when the EMSPROC task has to count the number of times an event occurs for an action that is contingent on a threshhold number of occurrences.

It is often useful to put user-written NCL procedures for triggering automated recovery and statistics collection in the EMSPROC task. If an event meets the filtering conditions for invoking such a function, the procedure can submit execution of the function to the BMON task for asynchronous background processing. This strategy is supported by both the NCL Automated Rules Set (NARS) and the Rules Management Services (RMS) applications.[1]

Filtering and execution of an NCL procedure within the EMSPROC task can be useful when an event invokes operations that apply across many tasks. For example, suppose there are a large number of MSGPROC tasks and all OCS screens are to display security events in reverse highlight. Rather than require each MSGPROC task to evaluate a filtering condition for security events and turn on reverse highlighting, it might be preferable to have the EMSPROC task evaluate events and turn on reverse highlighting for all OCS screens.

Placement of RMS

RMS can be configured in any of the major tasks, EMSPROC, LOGPROC, or MSGPROC, or in one of the background regions. Based on experience from performance testing, it is recommended that users configure RMS in either the EMSPROC task or the BMON region. Deciding between the EMSPROC task and the BMON region involves a tradeoff between fault tolerance and performance.

The EMSPROC task executes in the EMSP region. EMSP is the only region that provides a mechanism for event recovery in the case of a process failure. The recovery procedure provides fault tolerance if the process containing EMSP, either NMNC or NMEM, restarts within one minute of a failure. If RMS is configured in the EMSPROC task, it benefits from the EMSP recovery procedure. If RMS is configured elsewhere and the process containing it fails, events that were in queue for it before the failure may be lost.

When RMS is configured in the BMON task, it executes recovery functions as an asynchronous background task and does not interfere with the timely delivery of events to operators. Configuring RMS in the EMSPROC task has two disadvantages. First, it forces all events that enter NNM to be evaluated for RMS recovery. This increases the potential for the EMSPROC task to become a bottleneck during high event rates. (As discussed in the preceding section, to avoid a bottleneck, filtering in the EMSPROC task should be kept to a minimum.) Second, if RMS executes as the EMSPROC task, the user cannot execute a customized NCL procedure as the EMSPROC task. If RMS is configured in the BMON region, a user-written NCL procedure can execute as the EMSPROC task.

Configure RMS in either the EMSPROC task or the BMON region.

[1]Both RMS and NARS are provided with NNM Release 1.5. Previously, only NARS was available.

Executing RMS in the BMON Region

If RMS is configured to execute in the BMON region, BMON must be added as a user under the NNM User ID Management Services (UMS) subsystem.[2] This is necessary so that BMON's default parameters can be overriden.

In adding a user definition, a user ID and password must be provided. The specific password entered does not matter, but the user ID does. As a virtual user, BMON already has a user ID.[3] This ID must be used in defining BMON under UMS. To obtain BMON's user ID, use the command SHOW NCL=ALL. This will generate a list of user IDs for all executing NCL, including BMON.

In order for RMS to provide automated recovery, BMON must receive EMS messages. Once BMON is defined as a user under UMS, set Receive EMS Messages to Y on the the UMS OCS Details screen. Then restart NNM so that the new EMS-receive setting takes effect.

Once BMON starts receiving EMS messages, operators may begin to see duplicate event messages on their OCS screens. This will happen if an operator's user definition specifies both Receive EMS Messages as Y and Monitor Status as Y. To correct the problem, the operator should set Receive EMS Messages to N and keep Monitor Status as Y on the OCS Details screen.

[2]For instructions on adding user ID definitions, see the *NonStop NET/MASTER Management Services (MS) System Management Guide* (1993).

[3]For a discussion of virtual users, see the *NonStop NET/MASTER Management Services (MS) Operator's Guide* (1992).

Networking Strategies

Several tradeoffs and considerations are involved when developing operations management strategies in multinode environments. NNM can be configured as a standalone operations management tool on each node without regard to other NNM systems. It can be defined as a set of peer environments that manage their own domains and cooperatively manage shared resources. It can be configured as a central network control node that is connected to multiple satellite nodes. The management of the satellites can be distributed to each local node or it can be consolidated within the central system. Several other variations on these themes can be implemented. Some of the considerations involved in developing a multinode management strategy include

- The availability of system resources at each of the nodes.
- The cost of licenses for multinode environments.
- The effect of consolidating operations management at a central site versus distributing operations functions among remote nodes.
- Event processing considerations such as event rate and conversion costs.
- The placement of automated operator functions.
- The impact on operations staffing.

When a central node with NNM is to be connected to a satellite node, there may be some question about whether the satellite should run NNM and forward events over an INMC link or whether it should use an EMS Distributor to forward events, and not run NNM. The following considerations apply.

If an EMS Distributor forwards all events from a satellite node to a central node, it uses two-thirds the resources needed for sending the same event information over an INMC link. However, once received at the central NNM node, EMS event messages will be converted from Subsystem Programmatic Interface (SPI) format to MDO format. Events sent over an INMC link are already in MDO format, so there are no costs for MDO conversion on the central node. Total MDO conversion costs for EMS and INMC transmission are the same, the only difference is where the operation occurs. However, this can be important. When a central node receives events from two or more satellites, conversion costs are concentrated at the central site. At high event rates, this may affect performance.

In addition to event management considerations in a multinode environment, users should also consider the placement of automated operations functions. RMS provides automated recovery either at the node of origin or at the central site. If NNM and RMS are configured on each node, event processing and recovery can take place locally, without impact on the central network control node. If the Distributed Systems Network Management (DSNM) product is used on the remote nodes, RMS executing on the central node can recover failed objects on both the remote and central systems.

In most cases with multiple nodes, an effective strategy is to use INMC links and distribute management functions between the central node and its satellites. In this approach, satellite nodes run NNM and carry out some management functions and NCL automated recovery procedures locally. Other functions and escalated events are handled by the central node. This approach distributes MDO conversion costs to satellite nodes and provides multiple NMIS processes for handling multiple connections. It reduces the demands on the single EMSPROC task that processes all events received by the central node and it reduces network traffic for recovery operations.

Regulating Inter-Node Messages

To allow unsolicited event messages to move across an INMC link, it is necessary to use the UNSOLICIT option of the ISR ENABLE command. In situations where a central node needs to receive unsolicited event messages from all of its satellite servers, enter the following command on the central node:

ISR ENABLE UNSOLICIT = INBOUND

On a satellite connected only to the central node, enter the following command to send unsolicited messages:

ISR ENABLE LINK=<*link name*> UNSOLICIT = OUTBOUND

The preceding pair of commands sets up one-way traffic so that all unsolicited event messages from satellites go to the central node, but no unsolicited event messages from the central node are received at satellite nodes.

Displaying Roll-Delete Messages

Roll-delete messages are typically only of passing interest to an operator and do not require any operator acknowledgment. As the area on an OCS panel for displaying roll-delete message fills, new messages overwrite old messages. The length of time between the display of successive messages is determined by the PROFILE ROLL command.

If PROFILE ROLL is set to a value greater than zero and NNM is flooded with messages that need to be displayed on an OCS panel, a backlog of undisplayed roll-delete messages can build up. These messages are held in extended memory until displayed or the NRDLIM limit (see "GMM Processes," later in this article) is reached. A backlog of roll-delete messages can affect NNM performance and increase the time needed to recover from flooding.

To reduce backlogs of roll-delete messages, set PROFILE ROLL=0. During an event flood, there will be no delay between message displays, and large backlogs are less likely to accumulate. When the event flood subsides, buffered roll-delete messages will be displayed more quickly than they would at higher PROFILE ROLL settings, and the time needed for recovery will be reduced.

Starting NNM Processes Dynamically or Statically

Like Pathway, NNM allows the user to configure processes either statically or dynamically. Statically configured processes are created in specified CPUs at startup time. Dynamically configured processes are started in CPUs as needed, based on the relative parameter value of CPUWEIGHT. When they are no longer needed, dynamic processes stop. In general, statically configuring NNM processes to meet expected demand is likely to be preferable to configuring processes dynamically. Although static processes stay in memory and claim process control blocks (PCBs) even when they are not needed, this is often less significant than having processes created unpredicatably when the system is under stress.

Setting NCLUMAX

The parameter NCLUMAX specifies the maximum number of concurrent RMS and user-written NCL tasks. Set NCLUMAX higher than the maximum number of objects that might conceivably require automated recovery at the same time.

Setting KEYEXTR for Key-Sequenced Files

NNM supports the use of user-defined database (UDB) files. Often, users will want to define records in such files as structured, or *mapped*, records so that they can access elements or fields within the records. When using mapped, key-sequenced UDB files, be sure to set KEYXTR to NO when issuing the FILE OPEN operator. Otherwise, the record will not be mapped correctly. For further information, refer to the *NonStop NET/MASTER Network Control Language (NCL) Programmer's Guide* (1993).

Assigning CPU Priorities to NNM and DSNM Processes

NNM, RMS, and DSNM are sensitive to priority settings. This is particularly the case in respect to recovery processing, which is serial in nature and requires some processes to wait for others.

To avoid some of the problems associated with stress conditions and queuing, follow the general rule that servers should be given higher priorities than requesters. Figure 1 shows the specific priority settings used for NNM and DSNM processes during testing. The values are intended to be illustrative only, subject to modification based on the needs of individual installations.

In setting CPU priorities for NNM processes relative to application processes, there is a tradeoff to consider. Giving higher priorities to operations management tools may increase the overall availability of the application, but it will also reduce the application's access to CPU cycles and may affect the application's performance and response times, particularly during stress or failure conditions. On balance, however, it is usually preferable to assign higher CPU priorities to NNM processes than to application processes. During stress testing with RMS, it was found that a large number of failures (over 100 objects) can be successfully recovered if this priority schema is implemented.

NNM Memory Utilization

All NNM processes, with the exception of GMM processes, use standard C library procedures for the management of their extended-segment memory areas. The major uses of these process-specific memory areas are caching and buffering, which are discussed later. GMM processes use tailored procedures to map and manage their own extended memory allocations.

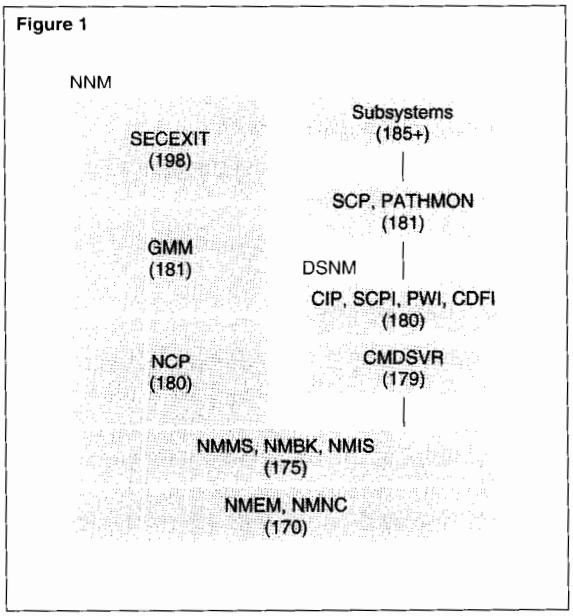


Figure 1.
Priority settings used for NNM, DSNM, and subsystem processes during performance testing.

GMM Processes

GMM processes use extended memory to maintain information that can be shared across NNM processes. Each GMM process maps and manages approximately one megabyte of extended memory. Slightly less than this amount is functionally available as global memory for all NNM processes. It is important to configure enough GMM processes to meet global memory requirements. If global memory is full when a GMM process requests space, the process fails. As a rule of thumb, it is advisable to have one GMM per CPU.

The information stored and managed by GMM processes includes:

- User-defined global variables that can be shared by all NCL procedures.
- Non-roll-delete (NRD) events that have not been resolved or deleted.
- EQUATE statements contained in INIT and READY files.
- SYSPARMS settings.

Examples of NCL procedures that share global variables are NARS and RMS recovery procedures, which use global variables to maintain the status of the objects they manage.

NRD event messages are held in global memory until resolved or deleted. The NRDLIM operand of the SYSPARMS command specifies the maximum number of NRD event messages that can be queued in global memory before older messages are deleted to make space for new messages.

EQUATE statements allow an operator to use a short character string in place of a full NNM command string. EQUATE statements in READY and INIT files are held in global memory and apply to all OCS panels and operators.

Setting EMSNRD. At a remote node with unattended operations, no OCS panel is active and there is no need to queue NRD events in global memory. To prevent all NRD message traffic and the use of global memory for NRDs, set SYSPARMS EMSNRD=NONE.

EQUATE Statements and GMM. EQUATE statements can be placed in READY and INIT files to globally apply to all OCS screens and operators. They can also be used in NCL procedures, such as individual MSGPROCs and LOGPROCs. To reduce global memory usage, unless an EQUATE statement is to apply globally, it should be placed in an NCL procedure for a MSGPROC task.

Caching

NNM event management processes such as NMNC and NMEM use caching techniques to optimize performance by reducing disk I/O. Each process caches frequently used information in its own extended memory and checks for the presence of cached values before retrieving them from disk. NNM allocates separate cache space for

- Panel caching.
- NCL caching.
- Template caching.

In setting aside cache space, there is always a tradeoff between reducing disk I/O in order to improve CPU utilization and contention for memory resources that might be needed by other processes.

Panel Caching. The formats for NNM screens are maintained in panel libraries on disk. Each screen has its own file in a library. When an operator requests a particular screen, NNM first looks in the panel cache for the NMNC (basic configuration mode) or NMMS (advanced configuration mode) process providing presentation services. If the screen is not there, NNM opens the appropriate file on disk, reads the contents, and closes the file.

To reduce disk I/O for screens, users can specify the number of panels for which cache space is to be allocated by setting the MAXPANEL operand of the SYSPARMS command. MAXPANEL applies across all NNM presentation processes; it allocates panel cache in NMNC and all NMMS processes.

The optimal value for MAXPANEL depends on the memory resources and response time requirements at an individual site. If operators frequently navigate between NNM panels, MAXPANEL should be large enough to encompass all frequently used panels.

NCL Caching. Four of the primary NNM processes, NMNC, NMEM, NMBK, and NMMS, allow memory space for preloading and caching NCL procedures. When one of these processes is to execute an NCL procedure, it first checks its NCL cache for the presence of the routine. If the routine is not in cache, the process opens and reads the routine's NCL object file. If the routine is new, and no object file exists, the process opens the routine's NCL source file, reads and compiles the NCL code, stores the resultant object in an NCL object file, and executes the procedure.

To reduce disk I/O for NCL procedures, a user can specify values for the PRELOAD and NCLPRSHR operands of the SYSPARMS command. The SYSPARM PRELOAD command specifies the particular NCL procedures that are to be preloaded and, if necessary, compiled at the time of the command, which is typically executed at startup. The preloaded procedures stay in memory cache and are always available without further disk I/O or compilation. The NCLPRSHR operand allows users to specify the amount of NCL cache space that is to be allocated for procedures that were not preloaded. Both PRELOAD and NCLPRSHR are system-wide parameters. All NMNC, NMEM, NMBK, and NMMS processes must preload the procedures specified with SYSPARMS PRELOAD and must set aside the NCL cache space required by SYSPARMS NCLPRSHR.

For optimal performance, all frequently used NCL procedures should be preloaded. In addition, SYSPARM NCLPRSHR should be set to a value that can accommodate as many nonpreloaded NCL procedures as are likely to run concurrently.

Users can test NCL procedures by setting PROFILE NCLTEST=YES. When NCLTEST=YES, NCL caching is disabled. Before executing an NCL procedure, NNM checks the timestamp on a procedure's source file with the timestamp compiled in its object file. If the timestamps are the same, NNM executes the procedure from the object file. If the timestamp on the source file is later, NNM gets the procedure from its source file and compiles it. Once testing is completed, it is important to set PROFILE NCLTEST=NO so that NCL caching is enabled and NNM can use preloaded NCL procedures.

For optimal performance, all frequently used NCL procedures should be preloaded.

Template Caching. Most events from Tandem subsystems are converted from SPI to MDO format through the use of event templates. When an NMNC (basic configuration mode) or NMEM (advanced configuration mode) process converts an SPI event to MDO format, it uses template services to format the text of the event and then caches the template in its own extended memory. In converting an event to MDO format, the process first checks its own template cache to find out whether the necessary template is there. Next, it attempts to retrieve the template from system memory. If the template is not in system memory (a nonresident template), the process must retrieve it from disk.[4] In contrast to panel caching and NCL caching, NNM does not provide a user parameter for controlling the amount of memory allocated for template caching.

Buffering

During stress conditions, NNM uses buffering in memory to minimize the impact of event floods and to optimize message traffic between NNM processes. In the first case, when a process receives events faster than it can fully process them, it buffers the events in extended memory queues. In the second case, to reduce queues as rapidly as possible, NNM buffers multiple events targeted for the same process into a single interprocess message (IPM). In estimating the resources required for NMM, capacity planners should consider both uses of buffering.

Queuing During Event Flooding. NNM responds to event floods by buffering unprocessed events in queues within extended memory. This applies to events that are to be forwarded to other processes. For example, events that pass through the EMSPROC task and are destined for delivery to MSGPROC tasks for presentation are queued until they can be forwarded for display on an OCS screen. Queuing applies in the same way for events that are to be delivered to the LOGPROC task for logging to BMON for automation, and to ISR tasks for intersystem communication.

The use of extended memory to cushion the effect of event floods can result in the allocation of additional memory and the increasing use swap file space. Depending on memory pressures from other applications, it can result in increased swap rates and reduced system performance.

Multiple Events in an IPM. If multiple event messages are queued for delivery to a given process or task, NNM attempts to block them into a single IPM as soon as the destination is available. This helps to reduce the demand on extended memory during event flooding and makes it easier for tasks to catch up with the event flow when flooding subsides. The effect of multiple-message buffering is more important in the advanced configuration mode than it is in the basic mode, where the interprocess communication of queued event messages only occurs between the NMNC process and the GMM and terminal I/O processes.

[4]For further discussion of templates and MDO conversion, see "NonStop NET/MASTER: Event Management Architecture" (Stockton, 1993a).

NNM buffers up to one screenful of messages in an IPM intended for OCS terminal display. For transmission across an INMC link, the number of messages that can be buffered into a single IPM depends on the message-unit size configured through the IOSIZE operand of the UNIT DEFINE command.

Conclusion

Fully installing and configuring NNM requires decisions about the configuration mode that is to be installed, the use and placement of event filtering, the extent to which functions are to be centralized or distributed between nodes of a network, the amount of global memory necessary, the utilization of extended memory for caching panels and templates, and many other matters. This article has attempted to provide guidelines in all of these areas or, where specific guidelines could not be given, to clarify some of the issues and tradeoffs involved. In some cases, decisions will need to be based on information about NNM processing costs and NNM's CPU, memory, and disk space requirements. Much of this information is provided in "NonStop NET/MASTER: Event Processing Costs and Sizing Calculations" (Stockton, 1993b).

References

Stockton, M. 1993a. NonStop NET/MASTER: Event Management Architecture. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Conputers Incorporated. Part no. 89807.

Stockton, M. 1993b. NonStop NET/MASTER: Event Processing Costs and Sizing Calculations. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Conputers Incorporated. Part no. 89807.

NonStop NET/MASTER Management Services (MS) Operator's Guide. 1992. Part no. 57247.

NonStop NET/MASTER Management Services (MS) System Management Guide. 1993. Part no. 95565.

NonStop NET/MASTER Network Control Language (NCL) Programmer's Guide. 1993. Part no. 94891.

Acknowledgments

Thanks are due to the entire NET/MASTER group. Special thanks go to Tony Bond, Rich Difeo, Rod Falck, Ted Hollman, and Robin Maybury.

Mark Stockton is a principal of the consulting firm Zen Systems. He has worked with Tandem as a consultant for over eight years. Initially, Mark worked with Tandem developers on the requirements, definition, and design of the DSM products. In 1989, he began working with the DSM Applications group as a performance consultant. In conjunction with the DSM Applications group, for the past two years Mark has worked on evaluating the performance of NNM.

NonStop NET/MASTER: Event Processing Costs and Sizing Calculations

In planning for the installation of the Tandem™ NonStop™ NET/MASTER (NNM) operations management product, it is important to have an estimate of the resources that will be needed. The first part of this article gives CPU processing costs for NNM event management, logging, presentation, intersystem communication, and memory management functions. The second part of the article demonstrates the use of these processing costs in sizing NNM. Detailed sizing calculations for an example of the NNM advanced configuration mode on a VLX™ system are presented. The final portion of the article gives representative CPU costs for Rules Management Services (RMS) recovery actions and provides data on NNM memory and disk space requirements.

The reader should be familiar with the basic architecture of NNM and its implementation in the basic and advanced configuration modes. This information is provided in "NonStop NET/MASTER: Event Management Architecture" (Stockton, 1993a) in this issue of the *Tandem Systems Review*.

The performance data presented in this article apply to NNM Release 1.5 and were gathered under Release C30.09 of the Tandem NonStop Kernel.

NNM Processing Costs

Tables in this section show CPU demand for executing individual NNM event management functions such as logging or displaying an event (see Table 1 for a list of these functions and their definitions) and for evaluating Network Control Language (NCL) if-then test conditions. The CPU cost of an individual NNM function is given on a per-event basis, such as the amount of CPU time it takes to convert a single event message from Subsystem Programmatic Interface (SPI) format to Mapped Data Object (MDO) format. For NCL test conditions, the statistics show the CPU cost of evaluating a single token in an MDO event message.

Values in tables are given as atomic (single event or single occurrence) demand costs to simplify their use in sizing calculations at sites with very different NNM configurations and operating characteristics. For example, if the cost of forwarding one event over an InterNET/MASTER Connection (INMC) link is 22.6 milliseconds in the basic configuration mode on a Cyclone/R™ system, at a site that expects an event rate of 4 events per second on a given INMC link, the expected CPU demand on the link is 90.4 milliseconds per second, or 9 percent CPU busy.

The preceding calculation may suggest that NMM performance costs are essentially linear. This is not true at higher event rates, but is a reliable approximation up to rates of one or two events per second.[1] In general, the values presented in this article are averages derived from performance testing and should be treated conservatively.

NNM Functions Included in Performance Statistics

During performance testing, a number of NNM functions yielded relatively constant CPU demand values. These functions are the basis for the atomics provided in the following sections. The functions are listed and defined in Table 1.

[1]When events are queued in memory for delivery to a target process, NNM buffers multiple events into a single interprocess message for the destination. At higher event rates, this lowers event processing costs. Maximum throughput is probably achieved at about 10 events per second, although this will vary, depending on the hardware platform. NNM's use of event queueing and message buffering is discussed in "NonStop NET/MASTER: Configuration and Performance Guidelines" (Stockton, 1993b).

Table 1.
NNM atomic functions

Function*	Definition
Receive an event (Receive)	The cost of processing the receipt of an event. Receive costs vary, depending on the process that receives the event. In the basic configuration mode, a receive cost is only charged to event processing in the NMNC process. In the advanced configuration mode, event processing, log processing, and display processing are carried out in separate processes and incur separate receive costs.
MDO conversion (MDO)	The cost of converting an event from SPI to MDO format. The cost depends on the number of tokens in the event. The statistics reported are for an event containing four user tokens in addition to standard header tokens.
Forward an event (Forward)	If an event passes EMSPROC filter criteria, there is a cost associated with forwarding the event to other NNM components. This cost varies depending on whether multiple event messages are buffered into a single interprocess message (IPM). The statistics reported here assume one event message per IPM.
Log processing (Log)	The cost of logging an event once it has reached the LOGPROC task and passed any LOGPROC filtering.
Display processing (Display)	The cost of displaying an event once it has reached a MSGPROC task and passed any MSGPROC filtering.
INMC	The cost of sending an event across an INMC link to another NNM system. (The cost of receiving an event across an INMC link is the cost of receiving an event at an NMNC or NMEM process and is given under "Receive an event".)
RMS filtering (RMS)	When RMS is configured as the EMSPROC task, each event reaching the EMSPROC task is evaluated to determine whether it should invoke RMS recovery. The cost of this evaluation is reported as the RMS filtering cost.
GMM-Primary (RMS) (GMM_P [RMS])	When RMS is active as the EMSPROC task, all events that pass the EMSPROC filter result in an IPM to a GMM process.** The processing cost at the primary GMM process is reported as GMM_P (RMS).
GMM-Backup (RMS) (GMM_B [RMS])	Same as GMM_P (RMS), except that the cost is for the GMM backup process.
NRD event (NRD)	The cost of processing a non-roll-delete (NRD) event within the EMSPROC task.
GMM-Primary (NRD) (GMM_P [NRD])	The EMSPROC task forwards NRD events to a GMM process that stores events in global memory and checkpoints them to its backup. The processing cost at the primary GMM process is reported as GMM_P (NRD).
GMM-Backup (NRD) (GMM_B [NRD])	Same as GMM_P (NRD), except that the cost is for the GMM backup process.

*Expressions in parentheses are shortened forms used in tables and formulas.

**This is because RMS NCL issues a global VARTABLE READ to see whether the subject of the event is defined within the GMM.

Table 2.
Event processing atomics for the basic configuration mode on CLX™, VLX, and Cyclone/R systems (times in ms/event).

Function	CLX	VLX	Cyclone/R
Receive	102.4	61.3	20.4
MDO	42.0	24.4	14.4
Forward	19.0	11.4	4.0
Log	10.9	7.3	3.4
Display	64.2	41.9	17.5
INMNC	76.1	42.7	22.6
RMS	166.3	102.5	35.7
GMM_P (RMS)	15.7	9.2	4.8
GMM_B (RMS)	2.0	1.2	0.6
NRD	73.2	37.3	23.9
GMM_P (NRD)	82.3	49.1	14.7
GMM_B (NRD)	4.0	2.3	1.3

Table 3.
Event processing atomics for the advanced configuration mode on a VLX system.

Function	VLX (ms/event)
Receive	
NMEM	31.8
NMBK	
LOGPROC (no NCL filter)	6.1
LOGPROC (with NCL filter)	11.1
NMMS	
MSGPROC (no NCL filter)	6.2
MSGPROC (with NCL filter)	11.2
MDO	52.1
Forward	
No NCL filtering in EMSPROC, INMC inactive*	6.9
NCL filtering in EMSPROC, INMC inactive	11.7
NCL filtering in EMSPROC, INMC active	14.7
Log	7.4
Display	41.9
INMNC	38.4
RMS	102.5
GMM_P (RMS)	9.2
GMM_B (RMS)	1.2
NRD (EMSPROC)	24.6
NRD (MSGPROC)	44.9
GMM_P (NRD)	49.1
GMM_B (NRD)	2.3

*This is because an INMC link is active when a session is in progress and the link is available to transmit messages. The link is inactive when no session is in progress or it is unable to transmit messages to a remote system.

Processing Costs for the Basic and Advanced Configuration Modes

Table 2 presents atomic CPU costs for functions in the basic configuration mode. Table 3 presents atomic costs for functions in the advanced configuration mode on a VLX system. Direct performance measures for the advanced mode on other platforms are not available. To calculate approximate atomic values for other platforms from VLX atomics, refer to the article "Measuring DSM Event Management Performance" (Stockton, 1992).

Processing Costs for System I/O Services

In addition to processing costs attributable to NNM components, there are performance costs for several system I/O services invoked by NNM. Table 4 shows CPU costs for the following services:

- Read an event template from disk (Read template).
- Write an event to a log file on a primary disk (Write log_P).
- Write an event to a log file on a backup disk (Write log_B).
- Read or write an event for terminal I/O (Terminal I/O).
- Read or write an event over a Tandem Expand™ network (Expand).

Processing Costs for NCL Filtering Expressions

Table 5 presents examples of CPU costs for processing NCL filtering conditions in the EMSPROC and MSGPROC tasks. The examples are selected to show the processing costs associated with event tokens of different data types. In general, expressions containing the same data type require similar amounts of CPU processing time.

Sizing NonStop NET/MASTER

This section presents procedures for sizing NNM. Sizing calculations are applied to the specific example of a VLX system running the advanced configuration mode. The calculations are presented according to processes in the advanced configuration:

- Event processing (NMEM).
- Log processing (NMBK).
- Display processing (NMMS).
- Intersystem routing (NMIS).
- Global memory (GMM).

In sizing the processes that make up NNM, it is important to include sizing costs for system services associated with NNM. In the following, system costs for accessing templates are included under NNM event processing, costs for writing to primary and backup log files are included under log processing, and terminal I/O costs are calculated with display processing. One additional system cost, interrupt handling, should be added as overhead to NNM sizing costs. As a conservative rule of thumb, interrupt handing can be estimated as 20 percent of the sizing cost calculated for NNM alone.

Sizing calculations for the basic configuration mode are carried out in the same way as for the advanced configuration mode. The only differences are that there are no receive costs for log processing and display processing in the basic mode, and that the sizing costs for the four event management processes in the advanced mode (NMEM, NMBK, NMMS, and NMIS) must be combined to yield a single cost for the NMNC process in the basic mode.

Sizing NNM requires information from the tables presented earlier in the article and estimates of site-specific characteristics such as event rates and the processing costs of user-written NCL filters in the EMSPROC, LOGPROC, and MSGPROC tasks. Collection of most of the required data is straightforward. However, estimating NCL filtering costs is more complicated and is treated separately in the next subsection.

Table 4.
System I/O services for NNM on CLX, VLX, and Cyclone/R systems (times in ms/event).

System service	CLX	VLX	Cyclone/R
Read template	13.3	7.5	3.9
Write $\log_P$	15.3	9.7	4.6
Write $\log_B$	3.2	2.4	1.0
Terminal I/O	20.5	10.1	6.2
Expand	27.5	14.7	8.2

Table 5.
Processing costs for sample NCL filtering expressions.

	EMSPROC			MSGPROC		
Filtering expression	**CLX**	**Cyclone/R**	**VLX**	**CLX**	**Cyclone/R**	**VLX**
If ZEMS^TKN^EMPHASIS	9.8	1.6	3.5	7.3	1.5	2.9
If ZSPI^TKN^SSID = *nnnn*	8.8	2.1	3.9	7.9	1.7	3.2
If ZEMS^TKN^SYSTEM = *nnnn*	10.7	1.5	3.9	8.6	1.6	3.2
If ZEMS^TKN^EVENTNUMBER.# = *nnnn*	11.4	1.8	4.5	9.0	1.9	3.8
If SUBWORD(ZEMS^TKN^CRTPID,1,1) = *nnnn*	13.1	2.1	5.2	10.0	2.2	4.7
If C2D(LEFT[ZEMS^TKN^USERID]) = *a* AND C2D(RIGHT[ZEMS^TKN^USERID]) = *b*	27.7	3.3	9.7	17.9	3.4	7.9
If user^TKN^FILENAME32 = *nnnn*	8.7	1.5	3.8	7.4	1.7	2.9
If user^TKN^DEVICENAME = *nnnn*	16.0	1.7	2.8	9.4	1.7	3.0
Additional tests on the same token*	4.3	0.8	2.6	4.3	0.8	2.6

*Values show the cost of repeating a test on the same token within an event.

Calculating Filter Costs

An important part of sizing NNM is the calculation of CPU costs for user-written NCL filters within the EMSPROC, LOGPROC, and MSGPROC tasks. Given atomic costs for individual NCL filtering conditions (see Table 5), the cost of filtering is determined by two factors:

- The order of test conditions in the NCL code for the filter.
- The probability of tokens meeting individual test conditions.

Table 6.
Calculation of average filtering costs for MSGPROC 2.

Test condition	CPU cost (ms)	Cum cost (ms)	Frequency (%)	Cost per 100 events (cum cost × freq. × 100) (ms)
IF ZSPI-TKN-SSID = *aaa*	3.2	3.2	25	80.0
OR SSID = *bbb*	2.6	5.8	20	116.0
OR SSID = *ccc*	2.6	8.4	15	126.0
OR SSID = *ddd*	2.6	11.0	10	110.0
IF SUBWORD(CRTPID,1,1) = *zzz*	4.7	15.7	5	78.5
Event does not satisfy any test condition*			25	392.5
Totals	**15.7**	**44.1**	**100%**	**903.0**
Average filter cost = 903.0 ms/100 events = 9.0 ms/event				

*Events that do not satisfy any test condition are evaluated at the cumulative CPU cost (cum cost) of the last condition in the filter.

Figure 1.
User-written filter for MSGPROC 2.

Figure 1

```
MSGREAD
  IF $&MSG.TANDEM.SPI THEN
    IF ZSPI-TKN-SSID = aaa
    OR SSID = bbb
    OR SSID = ccc
    OR SSID = ddd THEN MSGCONT else
    IF SUBWORD(CRTPID,1,1) = zzz
      THEN MSGCONT
    ELSE
      MSGDEL
ELSE
  MSGDEL
```

For demonstration purposes, consider the simple filter for MSGPROC 2 given in Figure 1 and the data and calculations in Table 6. The outcome of the calculations will be used later in the VLX sizing example.

The first column in Table 6 lists the test conditions in the filter of Figure 1. Column 2 shows the atomic cost of testing each condition. The Frequency column gives a percentage representing the likelihood of a token meeting individual test conditions.[2] The last column shows the average CPU cost contributed by a test condition to the total cost of the filter for every 100 events.

As shown in Table 6, 75 percent of events meet one of the test conditions in the filter. 25 percent of events do not meet any condition. Such events get tested on every condition in the filter, and therefore must be given the same cumulative processing cost as the last test condition in the filter.

In the table, the sum of all contributions to CPU costs per 100 events is 903 milliseconds. This is the average cost of the filter for 100 events. Thus, the average CPU cost of the filter per event is 903/100, or 9.0 milliseconds per event.

As pointed out earlier, the order in which expressions are executed in a filter can have a considerable effect on the cost of the filter. For example, suppose the positions of the first and last conditions in Table 6 were reversed. In this case, the condition

IF SUBWORD(CRTPID,1,1) = *zzz*

would show a CPU cost of 16 milliseconds/(100 events), instead of 78.5 milliseconds/(100 events), and the condition

IF ZSPI-TKN-SSID = *aaa*

would show a cost of 392.5 milliseconds/(100 events) instead of 80 milliseconds/(100 events). The result would be a total CPU cost of 1153.0 milliseconds/(100 events). The average cost per event would thus be 11.5 milliseconds instead of 9.0 milliseconds, a 28 percent increase in CPU cost.

[2]Event frequencies can be obtained through the Tandem Event Management Service (EMS) Analyzer product.

A shortcut in calculating filtering costs is to simply take the cumulative CPU cost (Column 3) of the last test condition as the average filter cost per event. This is equivalent to assuming that 100 percent of events do not meet any test condition. When the cumulative CPU cost of the last test condition is relatively low and it is known that most events do not meet any test condition, the result may be a sufficient approximation.

Sizing the Advanced Configuration Mode on a VLX System

The sizing calculations that follow are based on atomic costs in Tables 3, 4, and 5 and the data in Table 7.

Sizing Costs for Event Processing. Sizing costs are given as the CPU cost of a function, such as event processing, for each second of NNM operation. The CPU cost is given in milliseconds per second. The sizing cost of event processing is the sum of the sizing costs of its components (for component definitions, see Table 1):

- Receive an event.
- MDO conversion.
- RMS filtering (or user-written NCL filtering).
- NRD event.
- Forward an event.

During MDO conversion, an additional system cost may be incurred for accessing the template disk file (Read template).

Table 8 shows sizing formulas in milliseconds per second for the components of event processing. The formulas apply to both the advanced and basic configuration modes.

Applying the formulas in Table 8 to the example of a VLX system with the advanced configuration mode yields the following results:

- NMEM

  Receive ms/sec
  = (receive rate) × (receive cost)
  = 0.5 event/sec × 31.8 ms/event
  = 15.9 ms/sec

  MDO ms/sec
  = (receive rate) × MDO
  = 0.5 event/sec × 52.1 ms/event
  = 26.05 ms/sec

Table 7.
Sample data for sizing NNM on a VLX system.

Receive rate (NMEM)	0.5 events/sec
Receive rate (NMBK)	0.5 events/sec
Receive rate (NMMS)	0.5 events/sec
EMSPROC filter	RMS
EMSPROC pass rate	0.5 events/sec
NRD rate	0 events/sec
LOGPROC filter	FLUSH (no event filtering)
LOGPROC pass rate	0.5 events/sec
Number of operators	2
MSGPROC 1 filter	FLUSH (no event filtering)
MSGPROC 1 pass rate	0.5 events/sec
MSGPROC 2 filter	9.0 ms/event (see Table 6)
MSGPROC 2 pass rate	0.38 events/sec
ISR	Not used (no INMC links)

Table 8.
Sizing formulas for components of event processing.

NNM		
Receive ms/sec	=	(receive rate) × (receive cost)
MDO ms/sec	=	(receive rate) × MDO
User filter	=	(receive rate) × (filter cost)
RMS filtering ms/sec	=	(receive rate) × (RMS cost)
NRD event ms/sec	=	(critical event rate) × (NRD cost)
Forward ms/sec	=	(pass rate) × (forward cost)
System		
Read template ms/sec	=	(receive rate) × (read template)

  RMS event ms/sec
  = (receive rate) × (RMS event)
  = 0.5 event/sec × (102.5 ms/event)
  = 51.25 ms/sec

  NRD event ms/sec
  = (critical event rate) × (NRD cost)
  = 0 event/sec × 24.6 ms/event
  = 0 ms/sec

  Forward ms/sec
  = (pass rate) × (forward cost)
  = 0.5 event/sec × 11.7 ms/event
  = 5.85 ms/sec

Table 9.
Sizing formulas for components of log processing.

NNM	
Receive ms/sec	= (receive rate) × (receive cost)*
User filter	= (receive rate) × (filter cost)
RMS ms/sec	= (receive rate) × (RMS cost)
Logging ms/sec	= (pass rate) × (logging cost)
System	
Write log_P ms/sec	= (pass rate) × (write log_P cost)
Write log_B ms/sec	= (pass rate) × (write log_B cost)

*Advanced configuration mode only.

- System

  Read template ms/sec
  = (receive rate) × (read template)
  = 0.5 event/sec × 7.5 ms/event
  = 3.75 ms/sec

The sizing cost of event processing in the NMEM process is

Event processing ms/sec
= 15.9 ms/sec + 26.5 ms/sec + 51.25 ms/sec + 0 ms/sec + 5.85 ms/sec
= 99.50 ms/sec (10% CPU busy)

Sizing Costs for Log Processing. Log processing consists of receiving an event, filtering it, and logging it. A system cost is incurred for writing the event to log files on primary and backup disks (Write log_P, Write log_B). Table 9 shows sizing formulas for the components of log processing. Applying the formulas in Table 9 yields

- NMBK

  Receive ms/sec
  = (receive rate) × (receive cost)
  = 0.5 event/sec × 6.1 ms/event
  = 3.05 ms/sec

  [no user filter or RMS]

  Logging ms/sec
  = (pass rate) × (logging cost)
  = 0.5 event/sec × 7.4 ms/event
  = 3.7 ms/sec

- System

  Write log_P ms/sec
  = (pass rate) × (write log_P cost)
  = 0.5 event/sec × 9.7 ms/event
  = 4.85 ms/sec

  Write log_B ms/sec
  = (pass rate) × (write log_B cost)
  = 0.5 event/sec × 2.4 ms/event
  = 1.2 ms/sec

Total sizing cost of log processing in the NMBK process is

NNM log processing ms/sec
= 3.05 ms/sec + 3.7 ms/sec
= 6.75 ms/sec (0.7% busy)

Sizing Costs for Display Processing. The sizing cost of Operator Control Services (OCS) event displays is the sum of the sizing costs for individual MSGPROC tasks. A system cost is incurred for terminal I/O. Table 10 shows sizing formulas for display processing.

Applying the formulas in Table 10 to MSGPROC 1 in the VLX example yields

- NMMS

  Receive ms/sec
  = (receive rate) × (receive cost)
  = 0.5 event/sec × 6.2 ms/event
  = 3.1 ms/sec

  [no user filter or RMS]

  Display ms/sec
  = (pass rate) × (display cost)
  = 0.5 event/sec × 41.9 ms/event
  = 20.95 ms/sec

- System

  Terminal I/O ms/sec
  = (pass rate) × (terminal I/O cost)
  = 0.5 event/sec × 10.1 ms/event
  = 5.05 ms/sec

The sizing cost for MSGPROC 1 is

MSGPROC 1 ms/sec
= 3.1 ms/sec + 20.95 ms/sec
= 24.05 ms/sec (2.4% CPU busy)

The sizing calculations for MSGPROC 2 are

- NMMS

  Receive ms/sec
  = (receive rate) × (receive cost)
  = 0.5 event/sec × 6.2 ms/event
  = 3.1 ms/sec

  User filter
  = (receive rate) × (filter cost)
  = 0.5 event/sec × 9.0 ms/event
  = 4.5 ms/sec

  Display ms/sec
  = (pass rate) × (display cost)
  = 0.38 event/sec × 41.9 ms/event
  = 15.92 ms/sec

- System

  Terminal I/O ms/sec
  = (pass rate) × (terminal I/O cost)
  = 0.38 event/sec × 10.1 ms/event
  = 3.84 ms/sec

The sizing cost for MSGPROC 2 is

MSGPROC 2 ms/sec
= 3.1 ms/sec + 4.5 ms/sec + 15.92 ms/sec
= 25.52 ms/sec (2.6% CPU busy)

Table 10.
Sizing formulas for components of display processing.

NNM	
Receive ms/sec	= (receive rate) × (receive cost)*
User filter	= (receive rate) × (filter cost)
RMS ms/sec	= (receive rate) × (RMS cost)
Display ms/sec	= (pass rate) × (display cost)
System	
Terminal I/O ms/sec	= (pass rate) × (terminal I/O cost)

*Advanced configuration mode only.

System costs for terminal I/O for the two MSGPROC tasks are

Terminal I/O ms/sec
= 3.84 ms/sec + 5.05 ms/sec
= 8.89 ms/sec

Total sizing cost for display processing in the NMMS process is the sum of costs for display processing in the two MSGPROCs:

NNM display processing ms/sec
= 24.05 ms/sec + 25.52 ms/sec
= 49.57 ms/sec (5.0% CPU busy)

Sizing Costs for Intersystem Routing. In the VLX example, there are no INMC links and no costs for intersystem routing. When INMC links are present, the sizing cost of intersystem routing can be calculated as

Intersystem routing ms/sec
= (forwarding rate) × (INMC cost)

For example, if 0.5 events/second are forwarded from the EMSPROC task to the ISR task, the sizing cost for intersystem routing on a VLX system with the advanced configuration mode would be

Intersystem routing ms/sec
= (receive rate) × (INMC cost)
= 0.5 event/sec × 38.4 ms/event
= 19.2 ms/sec (2% CPU busy)

Table 11.
Sizing formulas for components of global memory management.

GMM (RMS) ms/sec = (receive rate) × (GMM_P [RMS] + GMM_B [RMS])
GMM (NRD) ms/sec = (critical event rate) × (GMM_P [NRD] + GMM_B [NRD])

Table 12.
Summary of NNM sizing costs for the advanced configuration mode on a VLX system.

NNM process	Cost (ms/sec)
NMEM	99.50 (10.0% CPU busy)
NMBK	6.75 (0.6% CPU busy)
NMMS	49.57 (5.0% CPU busy)
NMIS	0.00
GMM	5.20 (0.5% CPU busy)
Total	**161.02 (16.1% CPU busy)**
System function	**Cost (ms/sec)**
Read template	3.75
Write log	6.05
Terminal I/O	8.89
Interrupt handling*	35.94
Total	**54.63 (5.5% CPU busy)**
Total CPU demand, NNM plus system costs	215.65 (21.6% CPU busy)

*Interrupt handling is calculated as a 20% overhead on NNM CPU costs, including costs for other system services. Thus
Interrupt handling = 0.2 × (161.02 + 3.75 + 6.05 + 8.89)
= 35.94 ms/sec.

Sizing Costs for Global Memory Management. The components used in sizing GMM are

- GMM_P (RMS).
- GMM_B (RMS).
- GMM_P (NRD).
- GMM_B (NRD).

Table 11 gives formulas for sizing these components.

The sizing cost of GMM is

GMM (RMS) ms/sec
= (receive rate) × [GMM_P (RMS) + GMM_B (RMS)]
= 0.5 event/sec × (9.2 ms/event + 1.2)
= 5.2 ms/sec (0.5% CPU busy)

GMM (NRD) ms/sec
= (critical event rate) × [GMM_P (NRD) + GMM_B (NRD)]
= 0.0 event/sec × (49.1 ms/event + 2.3 ms/event)
= 0 ms/sec

The total sizing cost for GMM is 5.2 milliseconds/second, or 0.5 percent CPU busy.

Summary of NNM Sizing Costs for the Advanced Configuration Mode on a VLX System. Table 12 summarizes the results of the preceding sizing calculations. It includes the 20 percent overhead cost for system interrupt handling.

As shown in Table 12, for the example given, with no intersystem communication and RMS configured as the EMSPROC task, total CPU utilization for the advanced configuration mode on a VLX system is 21.6 percent.

CPU Costs for Automated Recovery Under RMS

Table 13 shows CPU costs for the automated recovery of failed objects under RMS. The costs reflect the combined CPU demand of NNM and the Tandem Distributed Systems Network Management (DSNM) product; subsystem costs are not included. The cost of automated recovery is the same for both the basic and advanced configuration modes.

Memory Requirements

When NNM is started, extended memory is allocated to all configured processes. The total amount of memory initially claimed can be considerable, often as much as 30 to 35 megabytes.[3] Page faulting is to be expected. However, the appearance that NNM requires inordinate amounts of memory is an artifact of the initialization process. Once initialization is completed, most NNM memory is released and available to other processes. As these processes request memory, the Tandem memory manager swaps out initialized NNM memory and allocates space to them. Test results show that NNM generally uses only about 10 percent of its initially allocated memory, although this may vary depending on the site.

Table 14 shows NNM memory requirements. In the table, the column Working-set memory reflects actual usage, and Total memory is NNM's total memory requirement. Figures in the table are based on the following conditions:

- RMS active in the EMSPROC task.
- One active OCS panel for displaying events.
- One operator logged on.
- One event per second.
- No global memory variables accessed.
- No NRD events received.
- Simple pass-all MSGPROC and LOGPROC NCL procedures.

[3]The amounts are for the advanced configuration mode, configured for one copy of each process, running on a CISC hardware platform. The amount of memory allocated depends on which, and how many, NMM processes have been configured. NNM parameter values for cache and buffering will also affect the amount of memory claimed during initialization.

Table 13.
CPU costs for RMS automated recovery on CLX, VLX, and Cyclone/R systems (CPU time in ms).

Recovery object	CLX	VLX	Cyclone/R
Pathway terminal	3.82	2.72	0.92
Pathway TCP	5.56	3.79	1.25
Pathway TCP and terminal	6.10	4.11	1.44
ATP6100 line, 4 subdevices and TACLs	152.41	54.92	32.06
Expand line	17.34	6.86	3.55

Table 14.
NNM memory requirements.

	CISC (2K pages)		RISC (4K pages)	
NNM component	Working-set memory	Total memory	Working-set memory	Total memory
NCP_P	6	873	7	632
NCP_B	3	873	2	632
GMM_P	9	756	9	451
GMM_B	3	756	2	451
$SECEXIT_P$	8	198	8	110
$SECEXIT_B$	5	198	4	110
NMEM	339	3112	469	2727
NMIS	7	2498	15	2267
NMMS	412	3678	424	3187
NMBK	168	2926	226	2625
NMNC	588	3916	670	3575
Totals: Advanced mode	960 (1.92 MB)	15868 (31.74 MB)	1166 (4.66 MB)	13192 (52.78 MB)
Totals: Basic mode	622 (1.24 MB)	7570 (15.14 MB)	672 (2.69 MB)	5961 (23.84 MB)

Additional memory would be required for global variables, INMC links, extensive use of NCL procedures, and other features. Additional memory is also used to buffer unprocessed events during event floods. This memory is released as event flooding subsides.

Table 15.
Subvolume space requirements for installing NNM on CLX and VLX systems.

Subvolume	No. of files	Total pages	Total file space (MB)	Unused pages	Unused file space (MB)
ZNNM	11	5954	12.2	6	—
ZNNMDATA	12	4266	8.7	225	0.5
ZNNMNDS	468	4114	8.4	1786	3.6
ZNNMPDS	399	1560	3.2	1058	2.2
ZDSMS	74	1940	4.0	153	0.3
ZNARS	17	70	0.1	26	—
ZNNMNDO	1	—	—	—	—
ZNNMNCS	0	—	—	—	—
ZNNMUCO	0	—	—	—	—
$SYSTEM.SYSTEM	3	164	0.3	22	—
Totals	**985**	**18068**	**37.0**	**3276**	**6.7**

Table 16.
Subvolume space requirements for installing NNM on Cyclone/R systems.

Subvolume	No. of files	Total pages	Total file space (MB)	Unused pages	Unused file space (MB)
ZNNM	11	20774	42.5	183	0.4
ZNNMDATA	12	4266	8.7	225	0.5
ZNNMNDS	468	4114	8.4	1789	3.7
ZNNMPDS	399	1560	3.2	1058	2.2
ZDSMS	74	4876	10.0	423	0.9
ZNARS	17	70	0.1	26	—
ZNNMNDO	1	—	—	—	—
ZNNMNCS	0	—	—	—	—
ZNNMUCO	0	—	—	—	—
$SYSTEM.SYSTEM	3	260	0.5	38	0.1
Totals	**985**	**35920**	**73.6**	**3742**	**7.7**

Disk Space Requirements

The installation of NNM and its component subvolumes requires approximately

- 37.0 megabytes of file space on CISC-based Tandem CLX and VLX systems.
- 73.6 megabytes of file space on RISC-based Cyclone/R systems.

Table 15 reproduces a Disk Space Analysis Program (DSAP) report showing subvolume space requirements for NNM installation on Tandem CISC-based systems. Table 16 presents the same information for RISC-based Cyclone/R systems.

In Tables 15 and 16, $SYSTEM.SYSTEM consists of the files NNM, DSNM, and ZDSCONF. When NNM runs under production conditions, five of the subvolumes listed in Tables 15 and 16 require added file space.

The subvolume ZNNMDATA provides files space for log files and any user-defined tokens or message templates. During performance testing, log files were defined for 100 pages of data. Records were an average of 220 bytes long. The amount of space necessary for log files will vary from site to site.

The subvolume ZNNMPDS contains NNM-supplied panel source files. Additional file space must be allocated for any user-created panels.

The subvolume ZNNMNDO contains the compiled, NNM-supplied NCL object file. During performance testing, this file consumed 2000 pages, or 4 megabytes, of file space.

The subvolume ZNNMNCS contains user-written NCL source code. The amount of file space required will vary according to the site. During performance testing, ZNNMNCS added 36 pages for 13 files.

The subvolume ZNNMUCO is for compiled, user-written NCL procedures. During performance testing, 2000 pages of file space were required in ZNNMUCO.

In addition to disk space for the subvolumes in Tables 15 and 16, NNM uses swap files for temporary storage of its extended segments. At the rate of one event per second, with RMS in the EMSPROC task and with DSNM and an EMS Distributor included, NNM requires

- 16.2 megabytes of swap file space for the basic configuration mode.
- 26.7 megabytes of swap file space for the advanced configuration mode.

Table 17 presents a detailed analysis of NNM swap file space requirements.

Table 17.
NNM swap file space requirements (in pages).

	Objects	Extended segments	Total	
NNM basic configuration mode				
NCP	128	1152	1280	
GMM	128	1152	1280	
SECEXIT	128	4	132	
NMNC	64	3784	3848	
Total			**6540**	**(13.4 MB)**
NNM advanced configuration mode				
NCP	128	1152	1280	
GMM	128	1152	1280	
SECEXIT	128	4	132	
NMEM	64	2240	2304	
NMBK	64	2048	2112	
NMIS	64	1728	1792	
NMMS	64	2688	2752	
Total			**11652**	**(23.9 MB)**
DSNM				
CMDSVR	64	20	84	
CIP	64	584	648	
SCPI	64	36	100	
PWI	64	8	72	
CDFI	64	8	72	
Total			**976**	**(2.0 MB)**
EMS Distributor	128	256	384	(0.8 MB)

Conclusion

Before installing NNM, it is important to have site-specific estimates of NNM's resource requirements in CPU utilization, memory, and disk space. This article has presented tables of atomic CPU costs for major NNM functions and demonstrated the use of atomic costs in sizing calculations. It has presented detailed data on NNM memory, swap-file, and disk-space requirements.

References

Stockton, M. 1992. Measuring DSM Event Management Performance. *Tandem Systems Review*. Volume 8, No. 1. Tandem Computers Incorporated. Part no. 65250.

Stockton, M. 1993a. NNM: Event Management Architecture. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Computers Incorporated. Part no. 89807.

Stockton, M. 1993b. NNM: Configuration and Performance Guidelines. *Tandem Systems Review*. Vol. 9, No. 4. Tandem Computers Incorporated. Part no. 89807.

Acknowledgments

Thanks are due to the entire NET/MASTER group. Special thanks go to Rich Difeo, Robin Maybury, Alexander Kasavin, and members of the DSM Applications group for collecting the numbers that are the basis of this article.

Mark Stockton is a principal of the consulting firm Zen Systems. He has worked with Tandem as a consultant for over eight years. Initially, Mark worked with Tandem developers on the requirements, definition, and design of the DSM products. In 1989, he began working with the DSM Applications group as a performance consultant. In conjunction with the DSM Applications group, for the past two years Mark has worked on evaluating the performance of NNM.

Implementing a Systems Management Improvement Program

Increasingly, corporations are requiring their MIS departments to deliver services continuously, streamline their operations, and reduce the cost of ownership of their information systems. To meet these challenges, operations groups must analyze, develop, and implement processes and tools to improve the capabilities and efficiency of systems management.

Launching a systems management improvement program will have a profound effect on the operations environment. One must define the scope and direction of the program based on the business requirements of the application. Often called service-level objectives (SLOs), these requirements specify the availability, performance, security, and integrity required by a business solution to meet end-user objectives.

This article describes a case study that shows how a user can implement a systems management improvement program, achieving results that help to satisfy SLOs. The company embarked on the project because their operations and support groups could not deliver their business services efficiently. They took too long and used too many resources to manage the system. Even with great effort and dedication, they did not have their environment under control.

When they completed the improvement program, they had improved the quality of end-user services by improving their applications' availability. They had reduced costs and improved the productivity and efficiency of the operators and system-support personnel. They had significantly reduced help-desk phone calls by implementing automation to detect and recover from problems before those problems affected end users. Most important, instead of being reactive, they could anticipate problems and changes, which gave them time for education and the evaluation of new technology.

Before describing the case study, this article briefly defines systems management improvement. It also describes a framework that can help users analyze and assess their situation before and after they carry out an improvement program.

The Systems Management Improvement Program

Managing an application that must always be available is far more challenging than managing one that must be available only during business hours Monday through Friday. To maintain availability during limited time periods, one can implement and test changes during off hours. To maintain permanent availability, one must do everything online.

Operations management (OM) professionals can approach online systems management problems by treating the entire operations endeavor as a set of processes that can be controlled, measured, and improved. Loosely defined, a process includes all the tasks, whether performed by software or human interaction, that accomplish a given objective.

One must consider not only the tools and products required to improve systems management, but also the way they will be used in a specific organization. Buying a product will not solve any problems if one does not plan the introduction of the new technology and train personnel effectively. One must consider the relationships among all the required tasks, the tools and methods used, and the skill, training, and motivation of the people involved.

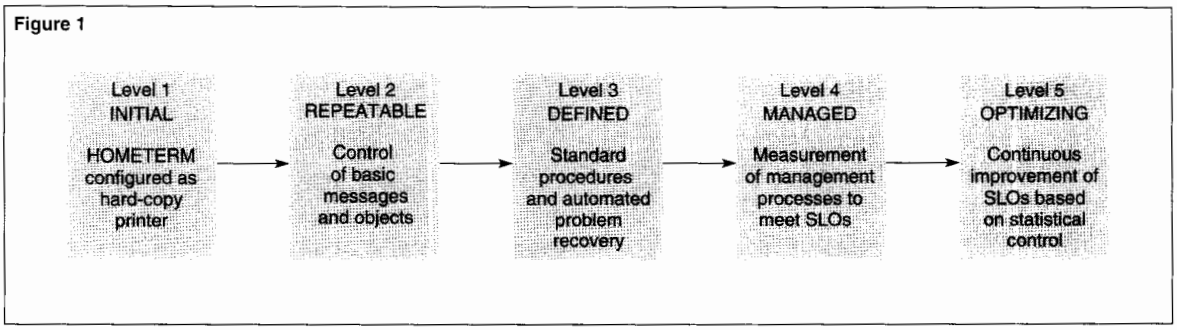
Improvement Steps

To improve their online systems management processes, OM organizations should follow six steps:

1. Assess the current status of their systems management processes.
2. Develop a vision of the processes they want to establish.
3. List the required process-improvement actions in the sequence in which they will be implemented.
4. Schedule the required actions and commit the resources to accomplish them.
5. Execute the plan.
6. Start over at Step 1.

The assessment step is critical. If one introduces changes without having a clear view of the strengths and weaknesses of the current environment, one could create unanticipated problems. For example, installing an automated operations product too early in the improvement program could degrade problem recovery instead of improving it. For more information about the kinds of problems that the introduction of technology can cause, refer to Norman (1993a and 1993b).

Figure 1.
Maturity framework for systems management processes.



The Maturity Framework

OM professionals should have a clear picture of the improvement goals as well as a way to gauge progress. The framework made use of in this article categorizes online systems management processes into five maturity levels. This framework, shown in Figure 1, roughly parallels the one defined by Crosby and Humphrey of the Software Engineering Institute at Carnegie Mellon University, and by Weinberg (Crosby, 1979; Humphrey, 1990; Weinberg, 1992).

The maturity framework can help to provide perspective and guide the direction of an improvement program. By determining the maturity level of their systems management processes, OM professionals and their managers can identify areas where improvement will be most fruitful. The five-level model, described below, will continue to evolve as knowledge of the ideal OM environment increases.

Level 1. An organization starts at maturity-level 1 when it first encounters a new technology and has to manage and control it. Operators perform tasks in an ad hoc fashion and use tools informally to solve problems. Consistent, documented procedures do not exist.

Level 2. At maturity-level 2, OM professionals have some experience with the management and control of the technology. They develop rules of thumb to solve simple problems. Some routine tasks are documented in run books. Operators can perform these tasks consistently and are freed to use their talents to solve complex problems.

Level 3. At maturity-level 3, OM professionals examine their processes and tools in depth and formally document them. Because they understand how problems occur and how to recover from them, OM professionals can safely introduce automated operations software to perform problem management.

Level 4. At maturity-level 4, OM professionals measure the efficiency of their systems management processes to test how well they are meeting their SLOs. They analyze the way they handle problems and assess the groups involved in solving problems. If there are deficiencies in the current procedures, they improve them. For example, they may examine the efficiency of automation and determine how to improve it to better meet their SLO for availability.

Level 5. At maturity-level 5, OM professionals continuously measure, analyze, and improve their systems management processes. They can plan for and incorporate new procedures and technologies because they have established methods for managing and improving processes.

Case Study Profile

In 1991, this North American company was growing rapidly, expanding its operations in Europe and Asia. Its worldwide users needed to have business services available almost continuously. The OM group no longer had periods of down time in which to perform maintenance and installation tasks.

Table 1 shows the characteristics of the company's production environment. The environment included over 10,000 objects (such as CPUs, disks, files, processes, communication lines, subdevices, and terminals).

The complexity of the system was growing rapidly. OM professionals had to ensure that each of the 10,000 objects was installed and configured correctly and that it ran efficiently. The business applications and the system generated more than 15 events (status, warning, and problem messages) per minute. However, most problems were reported by end users over the phone. Even the most experienced operators had difficulty detecting, recognizing, and recovering from problems in this complex environment.

As the quality of end-user services decreased, the OM group recognized that it would take a serious effort to cope with these new challenges. OM professionals expressed their concerns to the MIS managers, who decided in October 1991 to initiate a systems management improvement program.

Table 1.
Case study profile.

HARDWARE CONFIGURATION		
CPUs	**Network**	**Workstations**
One 12-CPU VLX™ system for all user applications	X.25	135 PCs
10 VLX CPUs for production use	SNAX™	750 6530 terminals
2 VLX CPUs for development use		

SOFTWARE CONFIGURATION	
OLTP and batch applications	**Database management**
OLTP applications based on Pathway and TMF™	Enscribe and NonStop™ SQL
2.3 million lines of code	5 gigabytes of data stored in mirrored disks
1,200 requesters	OLTP and batch use the same database
1,500 servers	
Batch processing controlled by NetBatch™	
Applications and system messages directed to HOMETERM hard-copy consoles	

SYSTEM ACTIVITY
240,000 OLTP transactions per day
500 batch jobs per day
65% CPU utilization
0.72 swap rate
Transaction rate growing at 5% per month

Assessment

In November 1991, OM professionals assessed their systems management processes. In particular, they measured outages, observed the working environment, and analyzed the effectiveness of their existing tools and processes.

Outage Measurement

The first problem they faced was the complexity of measuring business services; the applications provided a few hundred different kinds of business transactions to more than 1,000 end users. OM analysts found it nearly impossible to gather data showing how well they were delivering these services. They had information about the availability and performance of the whole system, but not of each business service. Most important, they had no data showing how well the services were delivered to each end user.

The assessment would have been easier if they had already established a way to measure SLOs for their business services. For this organization, finding a way to measure their current situation was itself a first step in improving their systems management processes.

Instead of tracking how well each individual service was delivered, they decided to measure all outages. To create a base measurement, they had the help-desk operators log each outage. The operator entered in the log the time of occurrence, end-user name, business services affected, a problem description, and the time to repair (outage duration).

By analyzing this information, OM professionals determined how many problems occurred per day, which business services were most affected, and how long it took to resolve problems and restore availability to the end user. During peak hours of the day, the help desk received between 20 and 25 phone calls per hour. Each outage took between 5 and 20 minutes to resolve.

The outages occurred at individual terminals or workstations, affecting only one or two end users at a time. Thus, these measurements evaluated availability in terms of each individual end user. (The system as a whole was available throughout the measurement period.) Though MIS managers did not establish an explicit service-level agreement with their end users, these measurements laid the groundwork for a de facto SLO for availability.

Assessment Findings

The assessment team evaluated how the help-desk operators performed their work, the processes and technology the operators used to solve problems, and their physical environment. The team created an operator workload profile that showed the times of day the operators were busiest and the number of interactions required to solve problems.

Too often, operators did not detect problems; end users phoned in to report them. Sometimes operators learned of a critical situation only when scores of messages started printing on hard-copy consoles. The audio alarms generated by these printers made the operations environment even more stressful. There were so many messages that the operators could not sift through them and take effective action. Other aspects of the physical environment included an inadequate telephone system and an insufficient work space, the latter making it hard to look at operator manuals and product documentation. It was also hard to read information on the terminal consoles. OM analysts were amazed that operators could perform even routinely in this unfriendly environment.

All problem recovery was performed manually. Its success depended greatly on the operators' knowledge and skill. Working under stressful conditions to handle a large and growing system, operators (even the most skilled ones) found it nearly impossible to avoid making errors.

All application and system messages were directed to hard-copy consoles configured as the HOMETERM device. In some situations, an application encountering an internal error (such as an arithmetic overflow) would prompt the hard-copy device.

The hard-copy console arrangement provided inadequate support for problem detection and analysis. Operators had trouble correlating the information on many pages of listings and building a high-level view of the current situation. Because they couldn't see what was going on in the system, they had little control over it.

To help operators monitor the system, OM analysts developed customized TACL™ (Tandem™ Advanced Command Language) macros, which monitored conditions such as available disk space and the presence or absence of CPU processes. However, the benefits were often limited because the macros had limited applicability. OM analysts had to recode or change them each time they changed system configurations. Thus, maintaining the macros was a time-consuming task.

In addition, an operator had to execute the macros and analyze the output. Often when a serious problem occurred, the operator was unavailable to perform these tasks, and the macros were not executed.

Based on this assessment, the OM analysts concluded that their systems management processes were at maturity-level 1.

MIS managers understood the risks of allowing the high rate of operations errors to continue. Moreover, they became concerned about the operators' job satisfaction. They supported the improvement program because it would benefit both the OM organization and end-user services.

Developing a Vision

MIS managers next began to develop a vision of the OM processes they wished to have in place. Recognizing that this was an iterative process, they began by outlining general goals.

First, they wanted to improve the quality of end-user services. To improve application availability, they needed to lower the number of application outages and reduce recovery time. To improve system performance, they needed to monitor system objects, prevent potential problems, and improve system measurement techniques.

Second, they wanted to reduce the complexity of systems management tasks. They could accomplish this by improving system visibility and control, by reducing the time it took to detect and correct problems, and by minimizing human intervention.

Third, they wanted to lower the cost of ownership by improving the productivity of their operators and OM analysts. To accomplish this, they would have to replace the primitive technology currently in use with tools that would monitor, diagnose, and correct problems.

After these goals were established, OM analysts outlined more specific objectives. At this stage they considered the products available and budget constraints, but they were still listing ideas without trying to organize or prioritize them. Their ideas were as follows:

- Handle all system and application messages through a standard console-management service.
- Define application instrumentation standards to be followed by all development groups and projects.
- Use a management tool to convert text messages in applications coded before the standards were established.
- Document all major system components, their configurations, and how they deliver services. Define the actions to be taken when problems occur.
- Automate intervention and recovery tasks, currently performed by human operators, for routine (recurring) problems. At a minimum, support these tasks with easy-to-use and effective command and control software.
- Acquire a standard automation product.
- Automate object-state monitoring.
- Automate performance monitoring. Make automatic reporting and trend analysis available on demand.
- Automate and instrument change management.
- Develop a model of the management process.
- Document the OM organization's structure and responsibilities.
- Implement automatic monitoring of all critical resources.

OM analysts also thought of monitoring each end-user service. However, this objective would be hard to achieve because it would require changes in the business application. For example, to measure response time, users would have to change the application to use counters in Tandem's Measure™ system performance measurement product.

Although they couldn't include this objective in their current improvement program, OM analysts did intend eventually to monitor SLOs for each service (such as availability, response time, and throughput). Monitoring would allow deviations in service levels to be reported immediately to the systems management group.

Action List

After envisioning the improvements they wanted to make, OM professionals analyzed the relationships among tasks, decided which tasks were most important, and determined the sequence in which to implement them. This, too, was an iterative process. The result was the following action list:

1. Manage system messages.
2. Manage application messages.
3. Improve system visibility by monitoring critical objects.
4. Introduce automated problem-recovery software.
5. Improve the efficiency of automation and other management processes by implementing OM process statistics.

These actions had to be carried out in order. For example, if OM analysts had introduced automation before they had control over their system and application messages, they could not have realized the full benefit of the automated operator. Similarly, if they had introduced object monitoring before performing actions 1 and 2, human operators would have faced another layer of unmanageable information instead of having enhanced control of the system.

Because system and application messages presented different challenges, OM analysts tackled them separately. Both actions included tasks (such as filtering out unimportant messages and building an online run book) that approximated levels 1 and 2 of the maturity-level framework.

Once they had stabilized their operations environment, OM analysts could carry out the remaining actions. They recognized the importance of going from one maturity level to the next without trying to skip a level. Their improvement program succeeded because they proceeded in small, sequential steps.

Resource Commitment and Project Scheduling

A project of this size and complexity had to be adequately staffed and financed. MIS managers assigned a senior OM analyst to work on the project for nine months. The analyst had supported Tandem systems for more than four years. He dedicated 75 percent of his time to the project and used the remaining 25 percent to perform normal system-support tasks. In addition, a senior operator was asked to assess new tools and processes and help install and configure those tools, adapting them to the specific needs of the operations staff.

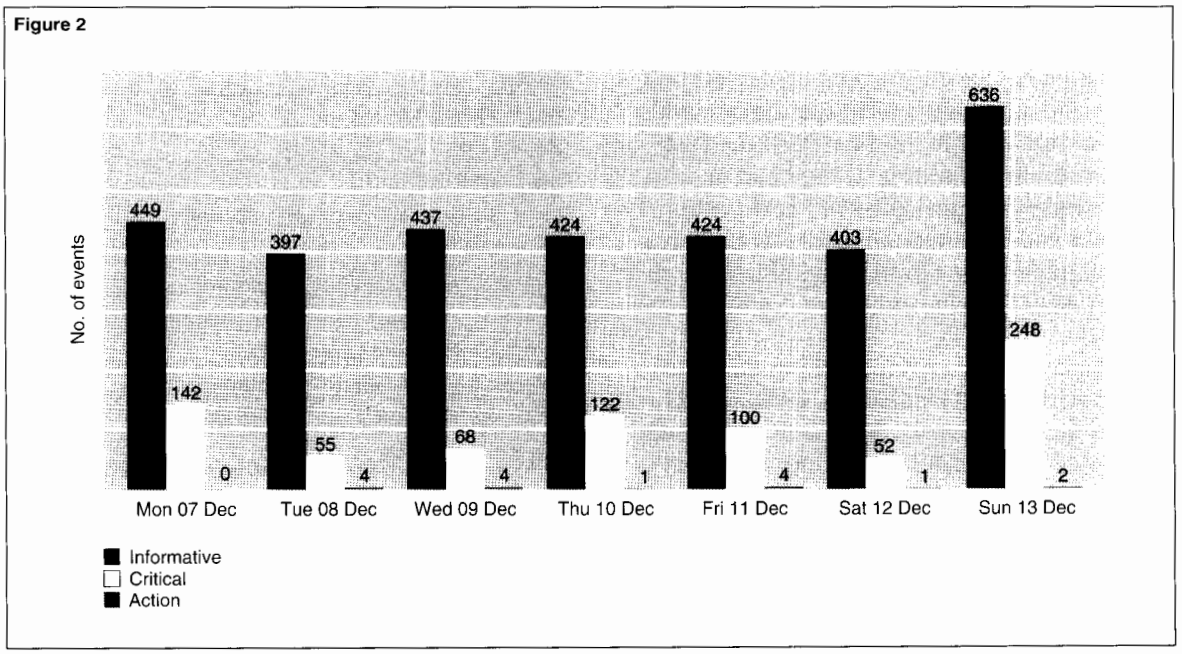
Table 2.
Schedule for the systems management improvement program.

Activity	Time allotted
Step 1: Assessment	2 resource-weeks
Step 2: Vision	2 resource-weeks
Step 3: Action list	1 resource-week
Step 4: Resource commitment and scheduling	—
Step 5: Execution	
Action 1: System message management	4 resource-weeks
Action 2: Application message management	4 resource-weeks
Action 3: Monitoring critical objects	6 resource-weeks
Action 4: Implementation of automation	8 resource-weeks
Action 5: Implementation of process statistics	8 resource-weeks
Step 6: Assess results	1 resource-week

The project succeeded in part because the operators and OM analysts were willing to learn new ways to do their work by using a new set of tools. MIS managers encouraged this attitude by supporting the staff from the outset. The managers said that most problems were not caused by the staff, but by the current systems management processes used to deliver services.

Once the actions and resources were defined, OM analysts could create a project schedule, shown in Table 2.

Figure 2.
Tandem subsystem events generated by the case-study system during one week.



Execution

Managing systems with hard-copy printers was inefficient. Before they could improve the management of system and application messages, OM analysts needed to acquire an operations console system.

They installed the operator console using the default configuration. The operations group took training classes and spent a few days getting used to the new facility.

Action 1: System Message Management

After a few days, operators noticed a typical problem of large systems. The system generated too many messages (many of them only informative events), which were then displayed on the operator console. Operators couldn't concentrate on reading these messages and selecting the important ones.

The Tandem Distributed Systems Management (DSM) architecture and most operator consoles provide a facility, called event filtering, that reduces message noise and highlights messages that require operator attention or intervention.

OM analysts took a systematic approach to building event filters. They began by analyzing all messages generated by their Tandem subsystems. For each subsystem, they selected the important messages, defined their severity, and documented the recovery steps. They produced a document that specified the critical events and described how operators should react to them. This task took about three weeks.

They used the document to build a set of filters managed by Tandem's Event Management Service (EMS), an operating system service and a component of DSM. The EMS filters selected only the events that were relevant to the users' environment. The filters also specified whether the events were critical.

Another benefit of the document was the creation of an online run book that defined the operational policy for each critical event. The run book, implemented with the console facility, was always available. OM analysts could override the definitions of the Tandem default messages and add unique definitions for their application messages.

Figure 2 shows a chart, created with Tandem's EMS Analyzer product, that presents a typical week of events generated by the case-study system. The chart only includes events generated by Tandem subsystems, not the user application. It shows that the system generated many informative events, which were not often used by operators. Also, there were more critical messages than actual problems, because sometimes the same message was duplicated, or the same problem was detected by multiple components, each of which generated its own events.

Action 2: Application Message Management

The next problem the OM analysts faced was to integrate their application messages into the console facility. To begin with, they established standards specifying that applications should use EMS for event generation.

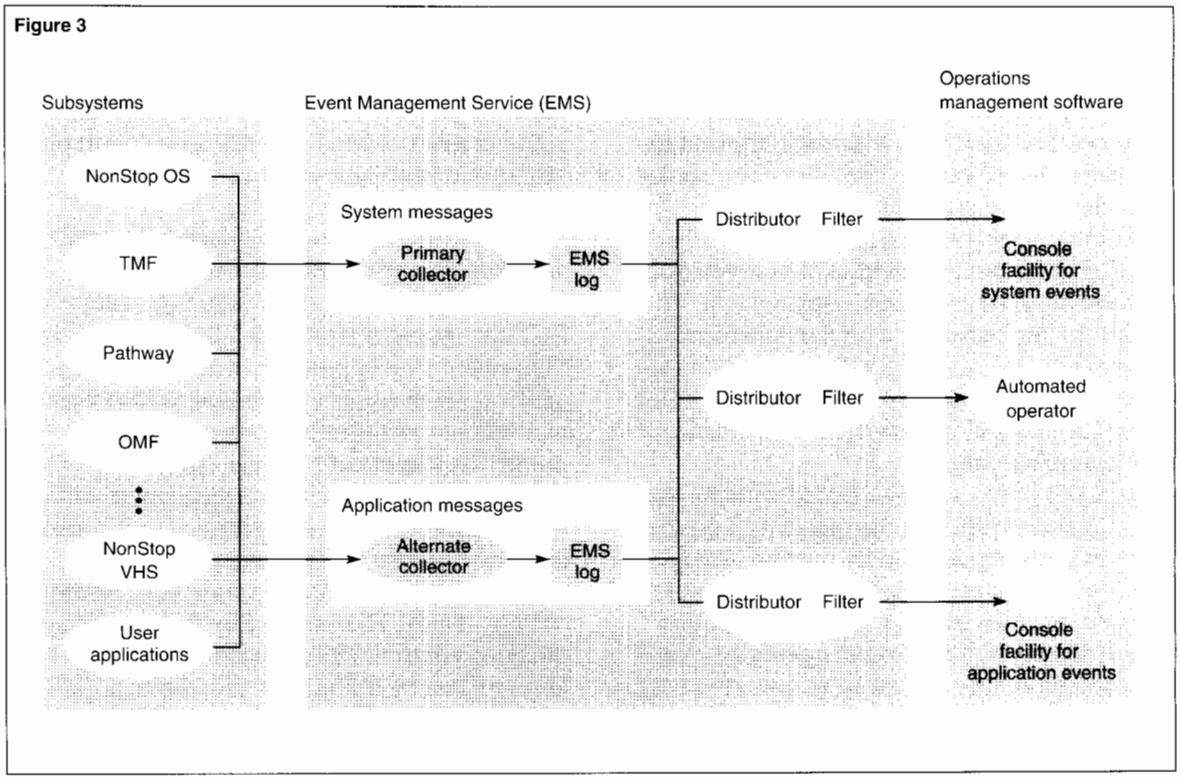
It was relatively easy to specify a policy for event generation and management that applied to developing new applications. For a detailed discussion of how to instrument applications, see the article by Dagenais in the October 1991 issue of the *Tandem Systems Review* (Dagenais, 1991). The *Standard EMS Events* document, available on Tandem's InfoWay™ online support service, provides more information about generating EMS events.

A more difficult task, and business decision, was to improve message management for the applications developed before DSM was introduced. OM analysts had to determine if they could convert or modify 1,500 application programs that sent their event messages in text format to a hard-copy printer.

After evaluating the effort needed to convert these programs, OM analysts decided they had to find a simpler solution. They used the Tandem NonStop Virtual Hometerm Subsystem (VHS) product, which manages application-message conversion to EMS format without requiring the programmer to modify any application code. To an application, NonStop VHS acts as a physical terminal; whenever it receives a text message, it converts the message into an EMS event.

OM analysts decided to implement NonStop VHS in stages. They started by converting applications in their development environment and finished by doing their most critical production applications. They converted all of their application parameters (the IN, OUT, and HOMETERM parameters in their Pathway transaction processing system) to use NonStop VHS as their HOMETERM device.

Figure 3.
The case-study system's OM environment.



Since the distinctions between critical, action, and noncritical messages, as defined by EMS, did not exist when MIS programmers designed the applications, the OM analysts had to find a way to highlight the critical and action messages. They followed the same strategy they had used to analyze Tandem system messages. They looked at three weeks of application messages and identified those that were critical and those that required operator action.

As a result of this analysis, they developed another EMS filter that searched text messages for words such as ERROR, ABENDING, ABORT, and EXCEPTION, and highlighted those messages. OM analysts used the filter to identify two types of messages: critical and action. Critical messages might affect the delivery of a business service; action messages required operator intervention.

Because the number of application messages per day was an order of magnitude greater than the number of system messages (8,000 versus 800), OM analysts decided to create a second console environment dedicated to the applications. Figure 3 shows the system and application event-management architecture they implemented.

Having system messages and application messages displayed on two consoles helped to correlate the causes of problems when they did occur. For example, a communication line going down may generate only one critical message, whereas the application may generate tens of them. This arrangement made it easier to understand the cause-effect relationship of problems.

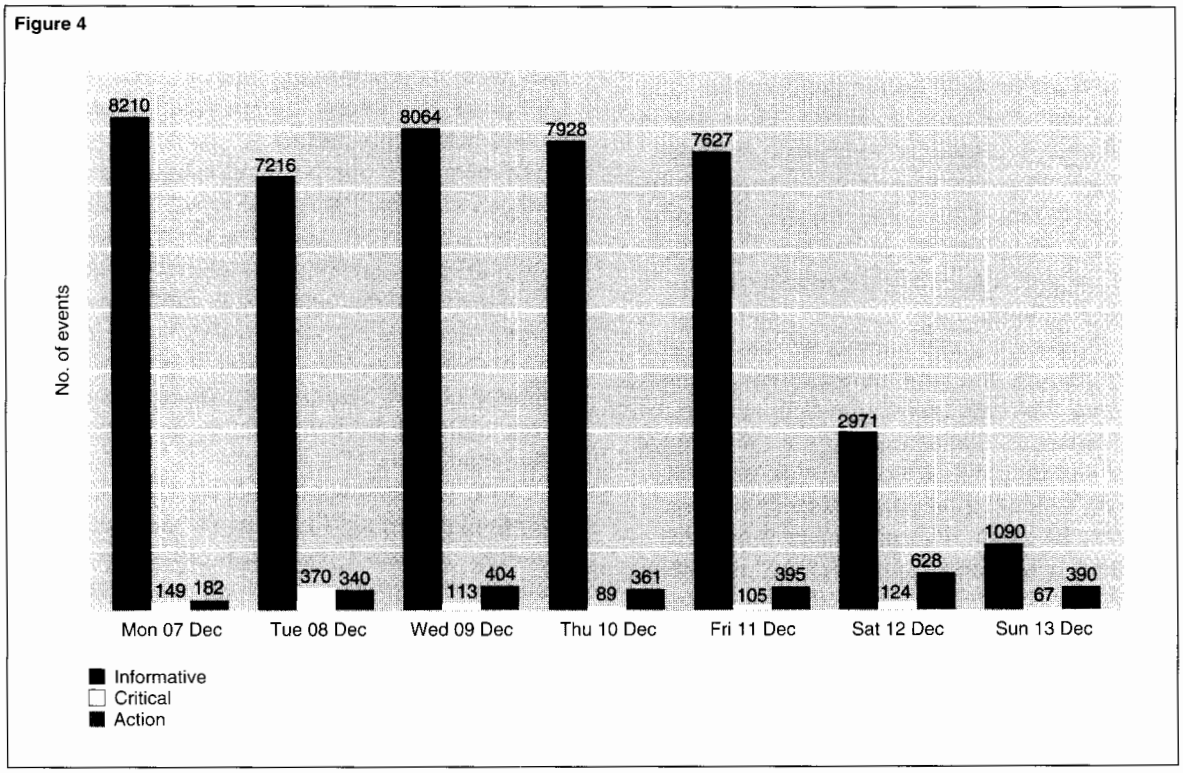


Figure 4.
User-application events generated by the case-study system during one week.

The chart in Figure 4 shows the events generated by user applications on the case-study system during the same week as that shown in Figure 2. These events came from a variety of applications: operations, online, point-of-sale, and batch. Batch applications generated more than 80 percent of the messages.

Using the information shown in Figure 4 and other EMS Analyzer reports, OM analysts identified the CPU processes that generated the most messages. They found that one process generated over 15 percent of all batch messages, and six processes generated 50 percent of all batch messages. They showed this information to development programmers, who made minor changes in the application programs and thus reduced the number of event messages.

The benefit of this action phase was the reduction of information overload. Operators could focus on critical events instead of having to react to every event. When a critical event appeared on the system console, it was highlighted. The operator could get an online description of the problem and recommended procedures explaining how to handle it. The new technology greatly improved problem visibility and also helped the operators to learn the DSM terminology.

The management of system and application messages created a solid foundation on which to build the remaining portions of the improvement program. If OM professionals had not analyzed the current messages and documented a policy for responding to them, it would have been risky to implement automation. The result might have been the automation of a badly defined process.

Action 3: System Visibility and Object Monitoring

The next action item was to improve the monitoring of all critical objects. More than 10,000 objects interacted to provide end-user services. CPUs, disks, printers, communication lines, processes, files, and terminals had to be fully and continuously operational. Operators could not possibly verify the health of this system manually.

In some cases, when an object reached a critical condition, an event message was generated. In others, no event message was generated, and it was up to the operator to monitor the object.

For example, when a disk or file becomes full, the Tandem disk process (DP2) will not generate an event. This can create a serious problem. It was imperative that this and other conditions be detected before they had a negative effect on the applications.

Instead of having operators wait for one of these objects to fail, OM analysts developed TACL monitoring macros. Executed once an hour, the macros informed operators of potential problems such as a stopped process, a disk or file becoming full, or a transaction running too long.

The macros, however, were error prone. Sometimes, when there was a serious problem, they were forgotten. They seldom provided information in a format that was readily usable by automation products.

To meet their object-monitoring needs, OM analysts selected Tandem's Object Monitoring Facility (OMF) software for several reasons. First, OMF constantly monitors objects at intervals defined by the user and as short as one minute. Second, it generates EMS-compatible events that can be filtered and displayed on the operator console and used by automated operations software to recover from problems or other conditions.

Third, it provides a high-level view of the system that operators can easily interpret. OMF can represent many thousands of objects and their states on one screen. With a quick look at this screen, operators get an immediate impression of the health of the system they have to manage.

OM analysts implemented OMF in stages, which had many advantages. After gaining experience with one or two objects, they could plan for the next objects more efficiently. The sequence of implementation was CPUs, disks, processes, spooler objects, and Tandem's TMF (Transaction Monitoring Facility). For each object type, they created an inventory that defined which objects were critical and what action to take when an object such as a printer stopped working. The inventory also indicated what to do when an object exceeded a predefined threshold (as when a disk became more than 80 percent full or a process changed priority).

Action 4: Introducing Automated Problem-Recovery Software

The preceding actions allowed operators to display significant events on integrated consoles and detect critical conditions before they occurred. Now OM analysts were ready to install a standard automated-operator product. They started by using the default rules set to perform problem recovery for the objects supported by those rules.

The default rules provided problem recovery for subsystems such as Pathway, Expand™ data networking software, and SNAX software. OM analysts then modified the basic rules to take into consideration the specifics of the installation. This approach gave the OM analysts a clear understanding of the automation process and provided insight into how the automated operator worked.

Automation often works in the background. OM analysts found that the lack of visibility of the recovery process posed a potential problem. In certain cases, they tried to bring down and restart a terminal or a communication line without success, only to find out later that the automated operator was working well and recovering from the failure efficiently.

To address this problem, OM analysts decided to give visibility to the recovery rules. Each time a rule executed, the automated operator would generate an event to inform operators of the outcome of the recovery procedure. The implementation of these recovery events helped OM analysts to understand the effects of automation and led to the creation of statistics on the efficiency of automation.

Another benefit of this effort was that OM analysts gained experience in customizing the basic rule set. The article by Collins in the October 1991 issue of the *Tandem Systems Review* explains in detail how to design and build rules for automated operations (Collins, 1991).

The basic automated operator did not actively monitor objects. It triggered a rule only when a subsystem generated an event. Moreover, its use was limited to a few subsystems that could generate EMS events and provide a programmatic interface.

OM analysts used OMF to extend the applicability of the automated operator. For each type of object supported by OMF, they wrote a customized recovery rule. In some cases, they used the automated operator as a passthrough to a TACL server, which contained the code for recovering an object and performed most of the recovery. For example, when a critical process failed, OMF detected it and generated an EMS event. The automated operator received the event and executed a customized PROCESS recovery rule, which sent the event-related information to a TACL server. The restart code executed by the TACL server then specified ASSIGN, DEFINE, and PARAM attributes before restarting the process.

This approach succeeded. It now takes less than five minutes to add a process to be monitored in OMF and write the specific restart code in TACL.

Action 5: OM Process Statistics and Efficiency of Automation

After implementing such significant changes, OM analysts wanted to measure the results. Useful statistics would greatly enhance the analysts' control over their systems management processes. Specifically, they wanted to review and optimize the automated recovery rules.

In actions 1 and 2, OM professionals used EMS Analyzer to analyze events generated by Tandem subsystems and user applications. In Action 5, they used EMS Analyzer to gather management information and analyze the net effect of automation on the system.

The first special reports the OM analysts created concerned their online applications. The reports showed problems with the way the application was handling terminal errors. In addition, configuration parameters were incorrect, and there were other application programming problems (such as *reply size invalid* errors). A team that included representatives from operations, system support, and application development reviewed the reports. They quickly identified and resolved many problems. This analysis was a simple process that for the first time gave visibility to transient and often undetected problems. It provided the operations group with management information that previously had been difficult to obtain. The information also allowed development programmers to improve the quality of the application by decreasing the number of failures.

The review team used a second set of reports to analyze critical events on data communication lines. Here, too, they identified recurring problems. By comparing events on the X.25 lines provided by two different telephone companies, they saw that one vendor clearly provided more reliable services than the other. When MIS managers learned of these differences, they decided to use only the more reliable telephone vendor, thus improving their own services to end users.

As a result of their analysis of the communication subsystems, OM analysts could improve the efficiency of operational command and control. Formerly, when multiple terminals on a telecommunication line went down, operators had to restart each terminal independently. Now, using Tandem's Subsystem Programmatic Interface (SPI), OM analysts developed programs in TACL and C to inquire about and restart multiple devices on telecommunication lines. They used Tandem's DNS™ (Distributed Name Service) software to manage these configurations. With this solution, the operator could, for example, execute a simple command to restart over 50 terminals; the program would also inform the operator of the result of the action.

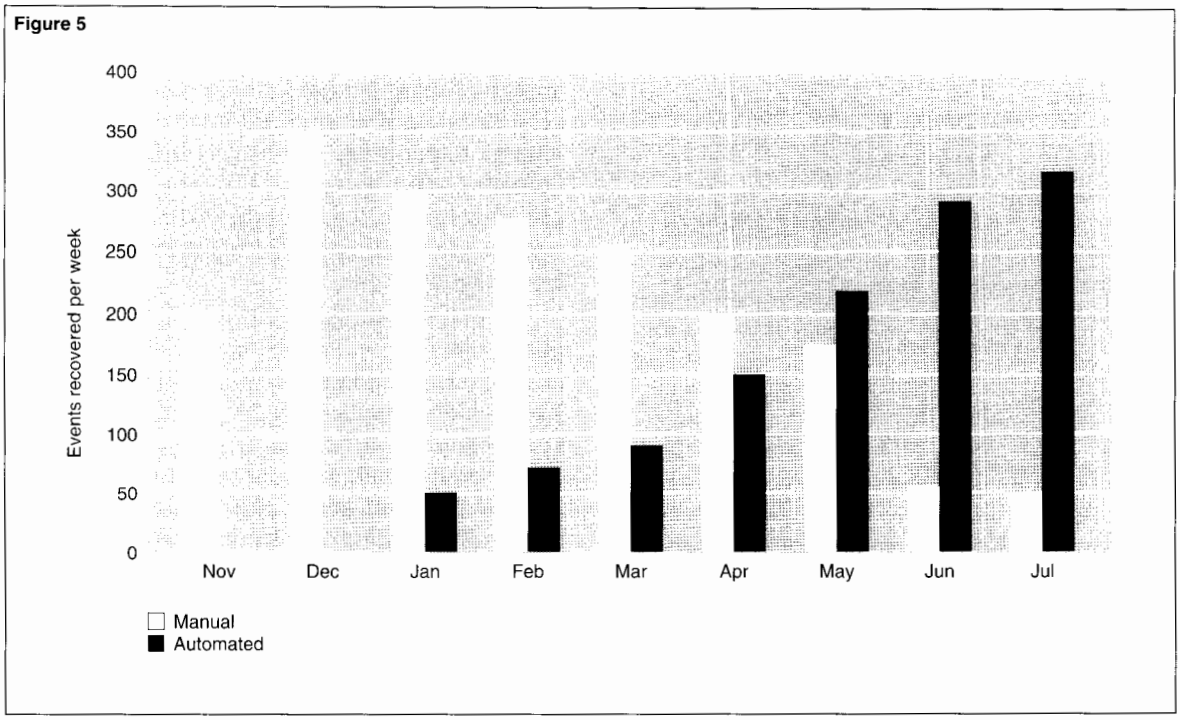


Figure 5.
A trend analysis showing the number of problem events recovered per week by human and automated operators during the improvement program.

Most important, OM analysts used statistics to track the efficiency of automation. During the first few months after the automated operator was installed, it recovered between 50 and 80 incidents per week without operator intervention. After OM analysts used OMF to develop and optimize new rules, automated recoveries grew to 300 per week. The objects being recovered came from the Pathway, X.25, SNAX, and Spooler subsystems.

The high number of recoveries proved that automation worked and that the improvement program had accomplished more than MIS managers had expected. Today, the automated operator automatically recovers between 200 and 500 incidents each week.

Figure 5 compares the number of problem events recovered manually with the number recovered by the automated operator during the improvement program. Each bar shows the average number of events recovered per week for the month indicated.

Manual event recoveries increased in December, after OM analysts installed the console facility. Because of the enhanced system visibility provided by the console facility, operators could detect and fix previously unnoticed problems. After OM analysts installed the automated operator, automated recoveries began to replace manual recoveries. The increasing efficiency of the automated operator gave human operators more time to manage other tasks.

Table 3.
Tandem products and tools that can help users implement a systems management improvement program.

Product or tool	Solution
NonStop NET/MASTER OCS	Centralized control of messages
EMS filters	Identification of important events
EMS analyzer	Analysis of important events
NonStop VHS	Conversion of application text messages to EMS events
OMF	Monitoring critical objects
NonStop NET/MASTER RMS	Automated recovery of objects
TACL macros	Simplified command and control
DSNM	Simplified command interface
DNS	Consolidation of object control
Measure	System performance measurement
ViewSys™	System performance measurement
NetBatch-Plus	Batch job management

Assessing the Improvement Program

After completing their improvement program, OM analysts evaluated their systems management processes and concluded that they were now at maturity-level 3. Moreover, because they had instituted some process-measurement procedures, they were moving toward maturity-level 4. Table 3 lists Tandem products and tools that can help users implement an improvement program.

The OM analysts recognized, however, that both technology and the business challenges they faced would continue to change rapidly. They intended to meet these challenges by proposing further improvements. They now had the tools and processes in place to be able to measure the situation before and after the proposed changes.

Further Reducing Help-Desk Phone Calls

After having automated many processes, OM analysts found that the number of calls to the help desk had been reduced from about 25 per hour to 10 per hour. Wanting to reduce this number further, they used the same methodology they had used during the initial assessment step, but focused on analyzing the causes of recurring problems.

For one month, operators documented each phone call received by the help desk. OM analysts then prepared summary information, from which they identified a particular problem. When an application terminal would go into a special state, it could not be detected as such by the application. The terminal was physically working, but from the application's point of view, it was in an exception mode.

OM analysts showed the problem to a support analyst, who developed a monitoring application that would evaluate the states of the terminals. If the monitoring application found a terminal in an unacceptable state, it would restart the terminal.

After implementing the new terminal-monitoring application, OM analysts measured help-desk phone calls for another month and presented the results to MIS managers. Calls were reduced by over 50 percent (to fewer than five per hour), showing that the new application was fixing the problem before end users became aware of it. Once again, the results achieved by the improvement program exceeded the expectations of MIS managers.

Assessing the Productivity of OM Professionals

Because the improvement program eliminated many low-level, repetitive tasks, operators became more productive. They had also learned new skills during the program. As a result, MIS managers could reduce the operations staff. They transferred one operator to the system-support group and another to the application quality-assurance group. These transfers lowered the cost of OM personnel and were positive career steps for the two former operators.

In addition, the improvement program freed up time for the system-support group, allowing them to learn C and write management programs such as the terminal-monitoring application described above. They also had time to start a client/server development project, which spearheaded client/server application design for the whole MIS department. Further, application developers became sensitized to OM issues during the program, which meant that new applications would be easier to manage, providing additional productivity in the future.

Conclusion

The case study described in this article shows that a systems management improvement program can provide benefits that exceed the expectations of MIS managers. OM organizations need to focus not simply on management products, but on all the processes they use to implement those products and provide services to end users.

A successful improvement program must begin with careful assessment and planning. Most important, OM organizations need to proceed in stages. For example, if they intend to introduce automation, they must filter and control system and application messages before installing automated operations software.

Once an OM organization completes an improvement program, it will have built the foundation for future improvements. It can apply the improvement process to a wide range of systems management problems. An improvement program can enhance OM productivity, raise the quality of services delivered to end users, and position the OM organization to take advantage of new technology and satisfy changing business requirements.

References

Collins, J. 1991. Writing Rules for Automated Operations. *Tandem Systems Review*. Vol. 7, No. 2. Tandem Computers Incorporated. Part no. 65248.

Crosby, P. 1979. *Quality Is Free*. McGraw-Hill.

Dagenais, J. 1991. Instrumenting Applications for Effective Event Management. *Tandem Systems Review*. Vol. 7, No. 2. Tandem Computers Incorporated. Part no. 65248.

Gilb, T. 1988. *Principles of Software Engineering Management*. Addison-Wesley.

Humphrey, W. 1990. *Managing the Software Process*. Addison-Wesley.

Norman, D. 1993a. *Turn Signals Are the Facial Expressions of Automobiles*. Addison-Wesley.

Norman, D. 1993b. *Things that make us SMART*. Addison-Wesley.

Weinberg, G. 1992. *Quality Software Management*. Volume 1. Systems Thinking. Dorset House Publishing.

Acknowledgments

I would like to thank the following people for making comments that greatly improved the content and presentation of this article: Jim Collins, Ted Schachter, Ries Wytenburg, Chantal Tremblay, Francois Carriere, Peter Alexiou, Bruce Brooks, Mike Choi, and Steven Kahn.

Jean Dagenais manages the Centre de Technologie Tandem de Montreal, where he supervises research and development in the area of operations management. He joined Tandem in Montreal as an account analyst in 1984. Since 1989, he has managed the development of four of Tandem's systems management products.

TandemSystemsReview*Index*

The *Tandem Journal* became the *Tandem Systems Review* in February 1985. Four issues of the *Tandem Journal* were published:

Volume 1, No. 1	Fall 1983
Volume 2, No. 1	Winter 1984
Volume 2, No. 2	Spring 1984
Volume 2, No. 3	Summer 1984

As of this issue, 23 issues of the *Tandem Systems Review* have been published:

Volume 1, No. 1	Feb. 1985	Volume 6, No. 1	March 1990
Volume 1, No. 2	June 1985	Volume 6, No. 2	Oct. 1990
Volume 2, No. 1	Feb. 1986	Volume 7, No. 1	April 1991
Volume 2, No. 2	June 1986	Volume 7, No. 2	Oct. 1991
Volume 2, No. 3	Dec. 1986	Volume 8, No. 1	Spring 1992
Volume 3, No. 1	March 1987	Volume 8, No. 2	Summer 1992
Volume 3, No. 2	Aug. 1987	Volume 8, No. 3	Fall 1992
Volume 4, No. 1	Feb. 1988	Volume 9, No. 1	Winter 1993
Volume 4, No. 2	July 1988	Volume 9, No. 2	Spring 1993
Volume 4, No. 3	Oct. 1988	Volume 9, No. 3	Summer 1993
Volume 5, No. 1	April 1989	Volume 9, No. 4	Fall 1993
Volume 5, No. 2	Sept. 1989		

The articles published in all 27 issues are arranged by subject below. (*Tandem Journal* is abbreviated as TJ and *Tandem Systems Review* as TSR.) A second index, arranged by product, is also provided.

Index by Subject

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
Designing and Implementing a Graphical User Interface	S. Wolfe	TSR	9,3	Summer 1993	89806
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
PATHFINDER–An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES *(cont.)*					
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
State-of-the-Art C Compiler	E. Kit	TSR	2,2	June 1986	83937
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CLIENT/SERVER					
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Designing and Implementing a Graphical User Interface	S. Wolfe	TSR	9,3	Summer 1993	89806
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
DATA COMMUNICATIONS					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DATA MANAGEMENT					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Batch Processing in Online Enterprise Computing	T. Keefauver	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32968
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Online Information Processing	J. Viescas	TSR	9,1	Winter 1993	89804
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
Optimizing Batch Performance	T. Keefauver	TSR	5,2	Sept. 1989	28152
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
OPERATING SYSTEMS					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Approach	A. Borr	TSR	1,2	June 1985	83935
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83935
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
PERFORMANCE AND CAPACITY PLANNING					
A Performance Retrospective	P. Oleinick, P. Shah	TSR	2,3	Dec. 1986	83938
Buffering for Better Application Performance	R. Mattran	TSR	2,1	Feb. 1986	83936
Capacity Planning Concepts	R. Evans	TSR	2,3	Dec. 1986	83938
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Credit-authorization Benchmark for High Performance and Linear Growth	T. Chmiel, T. Houy	TSR	2,1	Feb. 1986	83936
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
Estimating Host Response Time in a Tandem System	H. Horwitz	TSR	4,3	Oct. 1988	15748
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938
Getting Optimum Performance from Tandem Tape Systems	A. Khatri	TSR	2,3	Dec. 1986	83938
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
Implementing a Systems Management Improvement Program	J. Dagenais	TSR	9,4	Fall 1993	89807
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
Optimizing Sequential Processing on the Tandem System	R. Welsh	TJ	2,3	Summer 1984	83933
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PERFORMANCE AND CAPACITY PLANNING *(cont.)*					
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Performance Considerations for Application Processes	R. Glasstone	TSR	2,3	Dec. 1986	83938
Performance Measurements of an ATM Network Application	N. Cabell, D. Mackie	TSR	2,3	Dec. 1986	83938
Predicting Response Time in On-line Transaction Processing Systems	A. Khatri	TSR	2,2	June 1986	83937
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
The Performance Characteristics of Tandem NonStop Systems	J. Day	TJ	1,1	Fall 1983	83930
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11978
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
PERIPHERALS					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Data-Encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Data-Window Phase-Margin Analysis	A. Painter, H. Pham, H. Thomas	TSR	2,2	June 1986	83937
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
Peripheral Device Interfaces	J. Blakkan	TSR	3,2	Aug. 1987	83940
Plated Media Technology Used in the XL8 Storage Facility	D.S. Ng	TSR	2,2	June 1986	83937
Streaming Tape Drives	J. Blakkan	TSR	3,2	Aug. 1987	83940
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	3,2	Aug. 1987	83940
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PROCESSORS					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250
The High-Performance NonStop TXP Processor Transaction Processing	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
SECURITY					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SYSTEM CONNECTIVITY					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
Terminal Connection Alternatives for Tandem Systems	J. Simonds	TSR	5,1	April 1989	18662
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
SYSTEM MANAGEMENT					
Configuring Tandem Disk Subsystems	S. Sitler	TSR	2,3	Dec. 1986	83938
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Implementing a Systems Management Improvement Program	J. Dagenais	TSR	9,4	Fall 1993	89807
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Management Architecture	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Processing Costs and Sizing Calculations	M. Stockton	TSR	9,4	Fall 1993	89807
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
SYSTEM MANAGEMENT *(cont.)*					
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
UTILITIES					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Index by Product

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
3207 TAPE CONTROLLER					
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
5120 TAPE SUBSYSTEM					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
5200 OPTICAL STORAGE					
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	4,1	Feb. 1988	11078
6100 COMMUNICATIONS SUBSYSTEM					
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
6530 TERMINAL					
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
6600 AND TCC6820 COMMUNICATIONS CONTROLLERS					
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
Ada					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
BASIC					
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
C					
State-of-the-art C Compiler	E. Kit	TSR	2,2	June 1986	83937
CIS					
Customer Information Service	J. Massucco	TSR	3,1	March 1987	83939
CLX					
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
COBOL85					
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
COMINT (CI)					
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CYCLONE					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
DAL SERVER					
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DP1 AND DP2					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
DSM					
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
DYNAMITE					
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
ENABLE					
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
ENCOMPASS					
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
ENCORE					
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
ENSCRIBE					
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
EXPAND					
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
FASTSORT					
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
FOX					
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
FUP					
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
GUARDIAN 90					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
B00 Software Manuals	S. Olds	TSR	1,2	June 1985	83935
C00 Software Manuals	E. Levi	TSR	4,1	Feb. 1988	11078
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Multiprocessor Approach	A. Borr	TSR	1,2	June 1985	83935
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83934
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
INTEGRITY S2					
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
MEASURE					
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
MULTILAN					
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
NETBATCH-PLUS					
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NONSTOP NET/MASTER					
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Management Architecture	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Processing Costs and Sizing Calculations	M. Stockton	TSR	9,4	Fall 1993	89807
NONSTOP SQL					
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32986
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
OSI					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
PATHFINDER					
PATHFINDER–An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930
PATHWAY					
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
POET					
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
PS MAIL					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
RDF					
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848
RESPOND					
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
RSC					
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
SAFEGUARD					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SNAX					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
SPOOLER					
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11078
TACL					
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
TAL					
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
TCM					
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
TLAM					
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
TMDS					
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
TMF					
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934
TNS/R					
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
TRANSFER					
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TXP					
The High-Performance NonStop TXP Processor	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
V8					
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935
VIEWSYS					
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
VLX					
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
XL8					
Data-encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Plated Media Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937

TandemSystemsReview*OrderForm*

Use this form to order new subscriptions, change subscription information, and order back issues.

☐ I am a Tandem customer. My Tandem sales representative is _____.

☐ I am not a Tandem customer and am enclosing a check or money order for the requests indicated on this form. (Subscriptions are $75 per year and each back issue is $20. Make checks payable to Tandem Computers Incorporated.)

Subscription Information

☐ New subscription

☐ Update to subscription information
Subscription number: _____

Your subscription number is in the upper right corner of the mailing label.

COMPANY

NAME

JOB TITLE

DIVISION

ADDRESS

COUNTRY

TELEPHONE NUMBER (include all codes for U.S. dialing)

Title or position:

☐ President/CEO
☐ Director/VP information services
☐ MIS/DP manager
☐ Software development manager
☐ Programmer/analyst
☐ System operator
☐ End user
☐ Other: _____

Your association with Tandem:

☐ Tandem customer
☐ Third-party vendor
☐ Consultant
☐ Other: _____

Back Issue Requests

Number of copies

Tandem Systems Review

_____ Vol. 1, No. 1, Feb. 1985
_____ Vol. 1, No. 2, June 1985
_____ Vol. 2, No. 1, Feb. 1986
_____ Vol. 2, No. 2, June 1986
_____ Vol. 2, No. 3, Dec. 1986
_____ Vol. 3, No. 1, March 1987
_____ Vol. 3, No. 2, Aug. 1987
_____ Vol. 4, No. 1, Feb. 1988
_____ Vol. 4, No. 2, July 1988
_____ Vol. 4, No. 3, Oct. 1988
_____ Vol. 5, No. 1, April 1989
_____ Vol. 5, No. 2, Sept. 1989
_____ Vol. 6, No. 1, March 1990
_____ Vol. 6, No. 2, Oct. 1990
_____ Vol. 7, No. 1, April 1991
_____ Vol. 7, No. 2, Oct. 1991
_____ Vol. 8, No. 1, Spring 1992
_____ Vol. 8, No. 2, Summer 1992
_____ Vol. 8, No. 3, Fall 1992
_____ Vol. 9, No. 1, Winter 1993
_____ Vol. 9, No. 2, Spring 1993
_____ Vol. 9, No. 3, Summer 1993
_____ Vol. 9, No. 4, Fall 1993

Tandem Journal

_____ Vol. 1, No. 1, Fall 1983
_____ Vol. 2, No. 1, Winter 1984
_____ Vol. 2, No. 2, Spring 1984
_____ Vol. 2, No. 3, Summer 1984

For questions or ordering information, call 800-473-5868 in the U.S. and Canada or +1-408-285-0665 in other countries.

Send this form to:

Tandem Computers Incorporated
Tandem Systems Review, Loc 208-65
10400 Ridgeview Court
Cupertino, CA 95014-0723
FAX: +1-408-285-0840

Tandem employees must order their subscriptions and back issues through Courier.

Menu sequence: Marketing Information → Literature Orders → Technical Marketing Pubs (TSR)

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD

TandemSystemsReview*ReaderSurvey*

The purpose of this questionnaire is to help the *Tandem Systems Review* staff select topics for publication. Postage is prepaid when mailed in the United States. Readers outside the U.S. should send their replies to their nearest Tandem sales office.

1. How useful is each article in this issue?

Product Update

01 ☐ Indispensable 02 ☐ Very 03 ☐ Somewhat 04 ☐ Not at all

NonStop NET/MASTER: Event Management Architecture

05 ☐ Indispensable 06 ☐ Very 07 ☐ Somewhat 08 ☐ Not at all

NonStop NET/MASTER: Configuration and Performance Guidelines

09 ☐ Indispensable 10 ☐ Very 11 ☐ Somewhat 12 ☐ Not at all

NonStop NET/MASTER: Event Processing Costs and Sizing Calculations

13 ☐ Indispensable 14 ☐ Very 15 ☐ Somewhat 16 ☐ Not at all

Implementing a Systems Management Improvement Program

17 ☐ Indispensable 18 ☐ Very 19 ☐ Somewhat 20 ☐ Not at all

2. I specifically would like to see more articles on (select one):

21 ☐ Overview discussions of new products and enhancements
22 ☐ Performance and tuning information
23 ☐ High-level overviews on Tandem's approach to solutions
24 ☐ Application design and customer profiles
25 ☐ Technical discussions of product internals
26 ☐ Strategic information and statements of direction
27 ☐ Other_____

3. Your title or position:

28 ☐ President, VP, Director 29 ☐ Systems analyst 30 ☐ System operator
31 ☐ MIS manager 32 ☐ Software developer 33 ☐ End user
34 ☐ Other_____

4. Your association with Tandem:

35 ☐ Tandem customer 36 ☐ Tandem employee 37 ☐ Third-party vendor 38 ☐ Consultant
39 ☐ Other_____

5. Comments

_____

_____

_____

_____

NAME

_____

COMPANY NAME

_____

ADDRESS

_____

_____

10/93

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD