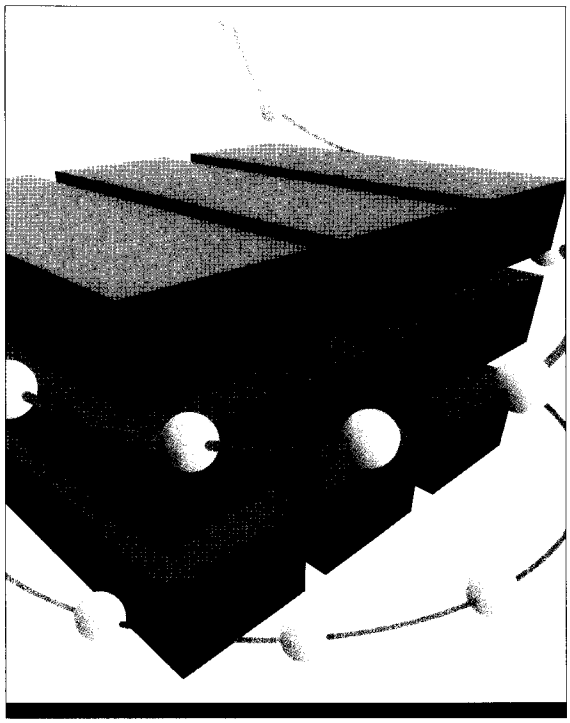
T A N D E M

SYSTEMS REVIEW



Client/Server Overview

Implementing Client/Server Using RSC

Designing Client/Server Applications for OLTP

Technical Information and Education

Product Update

T A N D E M

SYSTEMS REVIEW

VOLUME 8, NUMBER 3

FALL 1992

TANDEM

Editor's Note

Client/server computing represents a substantial departure from traditional computing strategies. By dividing applications between a client residing on a workstation and a server residing on a host, the client/server relationship takes advantage of both workstation and mainframe technologies. Because they greatly expand the range of applications and often decrease implementation time, client/server applications and technologies address the rapidly changing needs of both users and business.

Significant challenges face information systems groups as processing power is shifted to the desktop. Many more computer systems must be supported, and the complex issues of distributing applications and databases must be resolved. Despite these challenges, client/server is becoming the predominant application architecture of the 1990s.

This issue of the *Tandem Systems Review* includes three articles on client/server computing that will aid system designers and developers to implement client/server applications in Tandem environments. These articles describe how cooperative applications can be built today using connectivity and application development software provided by Tandem and third-party vendors.

Please take a few moments to fill in the "Reader Survey" questionnaire at the end of this book. We would like to receive your comments so that we can better meet your technical information needs. *—SWT*

EDITOR
Susan W. Thompson

WRITERS: Randall Frost, Steven Kahn, Richard Koman, Mark Peters

ASSISTANT EDITOR: Hannah Silver

PRODUCTION MANAGER
Anne Lewis

ILLUSTRATION AND LAYOUT
Christine Kawashima

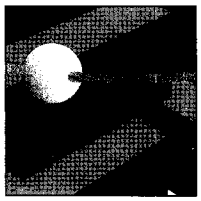
COVER ART: Steve Elwood

SUBSCRIPTIONS: Elaine Vaza-Kaczynski

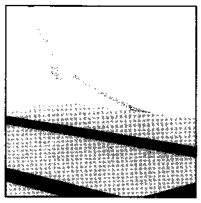
ADVISORY BOARD
Mark Anderton, Jim Collins, Terrye Kocher, Randy Mattran, Mike Noonan

Tandem Systems Review is published quarterly by Tandem Computers Incorporated. All correspondence and subscriptions should be addressed to *Tandem Systems Review*, 18922 Forge Drive, Loc 216-05, Cupertino, CA 95014.

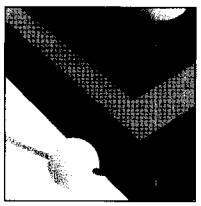
Subscriptions: $75.00 per year; single copies are $20.00. Detailed subscription information is provided on the subscription order form at the end of this book.



CLIENT/SERVER



DEPARTMENTS



Product

Systems Products

NonStop CO-Cyclone/R and NonStop CO-CLX800 Systems

June 1992

The NonStop CO-Cyclone/R and NonStop CO-CLX800 systems are fault-tolerant, high-performance computers geared to meet the needs of the telecommunications industry. The CO-Cyclone/R and CO-CLX800 systems provide all of the features of Tandem's Cyclone/R and CLX800 systems, respectively. In addition, the CO-Cyclone/R and CO-CLX800 are designed to meet the rigorous standards for equipment installed in telephone central office environments, as specified in Bellcore's Network Equipment Building System (NEBS) requirement.

Guardian 90 Based Software

MacOffender Software

August 1992

Tandem's MacOffender software makes it possible to monitor the performance of a NonStop system or network of NonStop systems from a single Macintosh computer. Through various charts and graphs, MacOffender displays in real time the resource usage statistics for the processors, processes, and disks in the system. It thus enables users to identify system performance problems quickly.

MacOffender does not require special training or the use of complex commands. For detailed information about a system resource, the user simply double-clicks the Macintosh mouse pointer on the corresponding field. In addition, MacOffender supports Apple's System 7 balloon help, which explains the system resource graphs and type of performance data that the software provides.

Tandem Reload Analyzer

August 1992

Tandem Reload Analyzer software enables users to quickly identify and prioritize for reorganization the fragmented tables or files within a database. The software examines the key-sequenced tables created by Tandem's NonStop SQL relational database management system or the key-sequenced files created by Tandem's Enscribe record management system. It then assesses the extent of fragmentation to determine the optimal time for reorganizing the database.

Tandem Data Build

August 1992

Tandem Data Build software provides a fast and efficient way to convert a wide variety of databases and various types of data to Tandem database structures. The flexibility of Tandem Data Build enables users to convert virtually any data (selected fields or whole records) that has an accurate source-record description.

The Tandem Data Build software generates a COBOL85 program that converts the data, creates a customized NonStop SQL or Enscribe database, and loads the converted data. In addition, Tandem Data Build maintains an audit trail, making it possible to verify the accuracy of each step in the conversion process. For users who may require assistance with their conversion, Tandem Data Build is available with a choice of two levels of on-site service, a training workshop or a conversion consulting service.

Guardian TN3270 Server

August 1992

The Guardian TN3270 server is a Transmission Control Protocol/ Internet Protocol (TCP/IP) service that enables users to access 3270 applications on Tandem NonStop systems. The product provides TCP/IP access to Pathway, TACL, TEDIT, PS Mail 3270, and other Tandem 3270 applications from terminals, hosts, and a wide variety of workstations. The Guardian TN3270 server is based on the TN3270 public domain protocol and can be used with many popular client software packages.

The Guardian TN3270 server is fully compatible with Tandem's Distributed Systems Management (DSM) services. It supports the Subsystem Programmatic Interface (SPI) for command and control, and logs all events to Tandem's Event Management Service (EMS).

Tandem Capacity Model 2.0

July 1992

The Tandem Capacity Model (TCM) is a host- and workstation-based tool for system modeling. It provides capacity planners with a reliable way to determine future resource needs and response times.

TCM 2.0 offers a number of new features. They include modeling of batch transactions; detailed workload information; priority queueing algorithms for estimating transaction response times; support of the new RISC-based systems; calculation of response times based on user-defined percentiles; future throughput estimates based on capacity history; merging of transactions from one performance model (PM) to another; and substantial improvements in the performance of MeasTCM.

Integrity Systems

Integrity CO-1300 Systems

June 1992

Tandem's Integrity CO-1300 systems are fault-tolerant, high-performance computers for critical telecommunications applications. The CO-1300 systems are specifically designed to meet the rigorous standards for equipment installed in telephone central office environments. The Integrity CO-1300 systems' rugged packaging is compatible with Bellcore's Network Equipment Building System (NEBS) requirement. The system meets stringent safety and electrical requirements, has a three-stage alarm interface, and is powered by redundant -48 or -60 volt DC feeds.

Integrity CO-1300 systems use the industry's most reliable implementation of UNIX System V, Release 4: Tandem's NonStop-UX operating system. In addition, these systems use reduced instruction set computing (RISC) microprocessors and thus offer the outstanding performance and other advantages provided by RISC technology.

Integrity CM-1300 Systems

September 1992

Tandem's Integrity CM-1300 systems are high-performance computers designed specifically to meet the needs of users who require fault-tolerant operation for critical business applications. Integrity CM-1300 systems use the industry's most reliable implementation of UNIX System V, Release 4: Tandem's NonStop-UX operating system. In addition, these systems use reduced instruction set computing (RISC) microprocessors and thus offer the outstanding performance and other advantages provided by RISC technology.

NonStop-UX Based Software

MHX400 Message Handling System, Release A10

June 1992

Release A10 of the MHX400 Message Handling System provides user agent (UA) functionality through the Tandem X.400 user agent and corrects reported problems on the MHX400 product. The Tandem X.400 user agent allows mail users on Tandem Integrity system local terminals to generate connections to public and private electronic mail worldwide.

With message transfer agent (MTA) and the X.400 UA, users can exchange messages with other mail users on local or remote X.400 mail systems through IEEE 802.3 Ethernet or X.25 networks. Use of the standard X.400 data interchange format and X.400 gateways also allows messages to be exchanged with proprietary electronic mail systems. The X.400 UA for the NonStop-UX system and the MTA are fully compatible and run as concurrent processes on the Integrity systems.

NonStop-UX Operating System, Release 2.0

September 1992

Release 2.0 of Tandem's NonStop-UX operating system provides advanced data-integrity features, an exceptionally robust and reliable kernel, and highly sophisticated diagnostic and maintenance facilities. The system is fully compatible with UNIX System V, Release 4.

Integrity System Management Suite

September 1992

Integrity System Management Suite (ISMS) is a suite of tools that simplifies the daily management of a single Integrity system or a network of Integrity systems. ISMS enables system operators and administrators to use an advanced graphical user interface (GUI) and object-oriented technology to manage Integrity systems from X-based terminals or X-compatible workstations.

SX25 Communications Software

September 1992

SX25 communications software is a new version of X.25 that is available with the NonStop-UX, Release 2.0, operating system. The SX25 software is functionally equivalent to the previous version of X.25, but incorporates new features and a different streams-style application interface that makes the product much easier to use. Among the new features of SX25 are link access procedure (LAP) high-level data-link control (HDLC), ISO-required 1984 facilities, larger packet sizes, and V.25 dial-up support.

Communications and Networking Products

Tandem Access/One

June 1992

The Tandem Access/One product line has been enhanced to include a two-slot enclosure, NetDirector software, and a fault-tolerant power supply (FTPS) for the 11-slot enclosure. The two-slot enclosure enables users to extend enterprise-wide networks to departments and remote locations. NetDirector is a multitasking network management platform that enables users to monitor, configure, inventory, and control their Ungermann-Bass networks easily and efficiently. With the FTPS for the 11-slot enclosure, users can protect their mission-critical networks from a loss of power. Users can replace their existing power supply module with the FTPS; no additional slots are required.

The Access/One smart hub provides a single architecture that enables users to connect dissimilar devices to Tandem NonStop and Integrity S2 systems in local area networks (LANs), wide area networks (WANs), and metropolitan area networks (MANs). Using twisted-pair telephone wiring to connect to devices, the Access/One hub minimizes the cost and time it takes to add and remove devices on a network. Access/One plug-in modules make it possible to add devices quickly as the need arises. With Access/One network management software, users can control network operations from a single console.

Workstation and Terminal Products

Nanao High-Resolution Super VGA Monitors

July 1992

Tandem now offers two high-resolution Super VGA monitors, the 17-inch Nanao F550i and the 14-inch Nanao 9065S. These monitors provide very high resolution as well as complete image control. In addition, these high-quality units conform to the strict MPR-II Swedish-government guidelines for electrical and magnetic emissions.

AST 14-inch Super VGA Monitor

July 1992

The AST 14-inch Super VGA monitor is a high-quality, low-cost monitor. It offers 800 x 600 noninterlaced resolution and a virtually limitless color selection.

Orchid Fahrenheit 1280 Graphics Card

July 1992

The Orchid Fahrenheit 1280 graphics card provides display resolutions up to 1280 x 1024 with 16 colors, and a 72-Hz refresh rate. It also incorporates an onboard graphics processor driven by the S3 Inc. graphical user interface (GUI) accelerator chip. The graphics processor makes opening, closing, and resizing windows virtually instantaneous. Scrolling and moving windows and graphics images are also much faster than with standard Super VGA.

PSX AP486/33E Workstations

July 1992

The PSX AP486/33E workstations are high-performance desktop computers with the Extended Industry Standard Architecture (EISA) I/O bus standard. These workstations employ Intel 33-MHz 486DX technology and feature an upgradable processor card, expandable memory and mass storage, and onboard video support. All AP-series workstations come with MS-DOS 5.0, QBASIC, utilities, a keyboard with 16 function keys, and user documentation.

The EISA I/O bus design provides advanced I/O capabilities for intelligent peripheral devices. The EISA 32-bit data path is double that of an Industry Standard Architecture (ISA) interface and thus helps to reduce I/O bottlenecks. Because it is an extension of ISA, EISA maintains backward compatibility with ISA peripherals.

PSX CP486/33 and CP386/33 Workstations

May 1992

The CP486/33 and CP386/33 workstations offer faster processing speeds than the CP486SX/20 and CP386SX/20 models, respectively, while providing all of the standard CP-series features and functions. These include upgradable processors; a built-in 16-bit Super VGA video adapter; a mouse port, two serial ports, and one parallel port; a 3.5-inch, 1.44-megabyte floppy drive; four half-height drive bays; and five expansion slots. All CP-series workstations also come with MS-DOS 5.0, QBASIC, utilities, a keyboard with 16 function keys, and user documentation.

PSX EP486/33 and EP386SX/25 Workstations

May 1992

The EP486/33 and EP386SX/25 workstations offer faster processing speeds than the EP486/25 and EP386SX/20 models, respectively, while providing all of the standard EP-series features and functions. These include a built-in 16-bit Super VGA video adapter; a mouse port, two serial ports, and one parallel port; four half-height drive bays; five expansion slots; a floppy/IDE disk controller; and room for up to 16 megabytes of memory. All EP-series workstations are also provided with MS-DOS 5.0, a keyboard with 16 function keys, and user documentation.

SCSI Disk and Tape Drives for CP and EP Workstations

May 1992

The SCSI disk drives and components for the CP-series and EP-series workstations can increase the disk storage capacity of these units to 800 megabytes. The SCSI host adapter required by the disk drives and an SCSI tape drive for the CP and EP workstations are also available.

Printer Products

Tandem 551x Line Printers

June 1992

The Tandem family of 551x heavy-duty line printers now includes five new models. These are the 5515-1, 5515-2, 5516-2, 5518-1, and 5518-2. The 5515-1 is similar to the previous 5515, except that an RS-232C/current-loop, Tandem Asynchronous Protocol (TAP) interface, and a 25-foot cable are standard on the 5515-1. TAP is a Tandem proprietary serial interface protocol that provides for improved data integrity and fault tolerance in the printers. The 5515-1 has a print speed of up to 420 lines per minute (lpm) and a rated duty cycle of up to 25,000 pages per month.

The 5515-2 is an enhanced floor model that includes a quietized cabinet for very low noise levels. An improved passive paper stacker and the RS-232C/current loop, TAP interface, and 25-foot cable are all standard. The 5515-2 has a print speed of up to 420 lpm and a recommended duty cycle of 11,000 to 63,000 pages per month. The 5516-2 has the same configuration as the 5515-2, except for its maximum rated print speed of 840 lpm and recommended duty cycle range of 30,000 to 230,000 pages per month.

The 5518-1 includes a new power paper stacker, with closable doors for quiet operation, and a Dataproducts parallel interface. It has a rated print speed of up to 1600 lpm and a recommended duty cycle range of 85,000 to 650,000 pages per month. The 5518-2 is similar to the 5518-1, except that it has a passive paper stacker instead of the power paper stacker.

An Overview of Client/Server Computing on Tandem Systems

Client/server computing is a strategy that takes advantage of the processing resources available in desktop workstations. The central concept is that an application is split between a client residing on a workstation and a server residing on the host. The client makes a request of the server and receives a reply from the server.

Client/server represents a substantial departure from traditional computing strategies; it is a recognition that computing resources are no longer centralized but are distributed throughout an organization. It relies on open interfaces to connect workstations and host systems, distributing application work across both large and small computers.

By greatly expanding the range of applications that can be built, and in some cases decreasing the time it takes to implement them, client/server applications and technologies address the rapidly changing needs of both users and corporations. Client/server shifts computing duties from the host and presents graphical user interfaces (GUIs) to users. It also provides machine independence and lets users analyze host data by using off-the-shelf software programs.

Significant challenges face information systems groups implementing distributed applications. There are new tools to use, new skills to learn, and many computer systems to support. Managing production-level client/server applications is more complex than managing terminal-based systems. Perhaps the greatest challenge is the change in thinking required to make the transition from a central-control model for traditional applications to a distributed-control model for client/server applications.

Despite these challenges, client/server is becoming the predominant application architecture. It is estimated that by 1996, half of all new application implementations will be client/server (Information Week, 1991).

Tandem™ computer systems, based on the Guardian™ 90 operating system, are predisposed to the demands of client/server applications, both query-oriented applications and the more challenging online transaction processing (OLTP) applications. Tandem's requester-server architecture and the NonStop™ SQL relational database management system facilitate these transactions. Similar software facilities are now available on Tandem's NonStop-UX™ operating systems. (NonStop-UX is Tandem's version of UNIX.)

In addition, Tandem's strategy of supporting open standards maximizes system interoperability. By supporting database standards, networking protocol standards, and application programmatic interface (API) standards, Tandem allows workstations of all types to participate in client/server applications. Where standards are still emerging or nonexistent, Tandem has provided APIs and client connectivity software to allow users to create client/server applications.

This article presents three primary architectural models and two variations for implementing client/server applications in Tandem host environments. Both the Guardian 90 and the NonStop-UX operating systems are considered. The article provides a working definition of client/server and describes the technological and market forces behind the movement to client/server architectures. Development considerations for client/server applications are also discussed. The article assumes that the reader is familiar with desktop workstations, the Tandem Pathway transaction processing system, and the NonStop SQL relational database management system. This article is not a compendium of available products for creating client/server applications.

Definitions

Client/server standards are still evolving as software and hardware vendors advance the technology in an atmosphere of collaboration and competition. This situation has left the definition of client/server imprecise. For clarity, this article assumes the following definitions, derived from the Gartner Group (1991).

Presentation is the application user interface. Functions handled by the presentation portion of an application include the display of forms, windows, menus, dialog boxes, graphics, and the handling of basic field editing and data validation.

Application logic is the "personality" of an application. It constitutes all programming logic that describes and performs the business functions. It issues database calls and, in some instances, issues calls to a user-interface subsystem.

Data management executes database calls and performs transaction management.

Requester-server is a relationship between two independent programs that interoperate to carry out a single unit of application work. The requester program directs and invokes the server program. The requester and server processes may reside on the same processor or on two different processors.

Cooperative processing, a form of requester-server, is a system design technique and programming construct in which the three functions of an application are split among two or more processors. These three functions, presentation, application logic, and data management, are defined above.

Client/server is an instance of cooperative processing in which the end user interacts with the computing environment through a programmable workstation. The workstation executes some portion of the application logic beyond terminal emulation. In general, but not always, the client process resides on a workstation and the server process resides on a second workstation, minicomputer, or mainframe system.

Workstation, for the purposes of this article, refers to any desktop computer such as a PC, Apple Macintosh, or UNIX workstation.

Host refers to any Tandem mainframe, minicomputer, or workstation server.

Graphical user interface (GUI) refers to an interface that relies on graphic metaphors such as push buttons, windows, and menus to represent applications, elements, and file management schemes. A GUI characteristically relies on a pointing device to initiate operations.

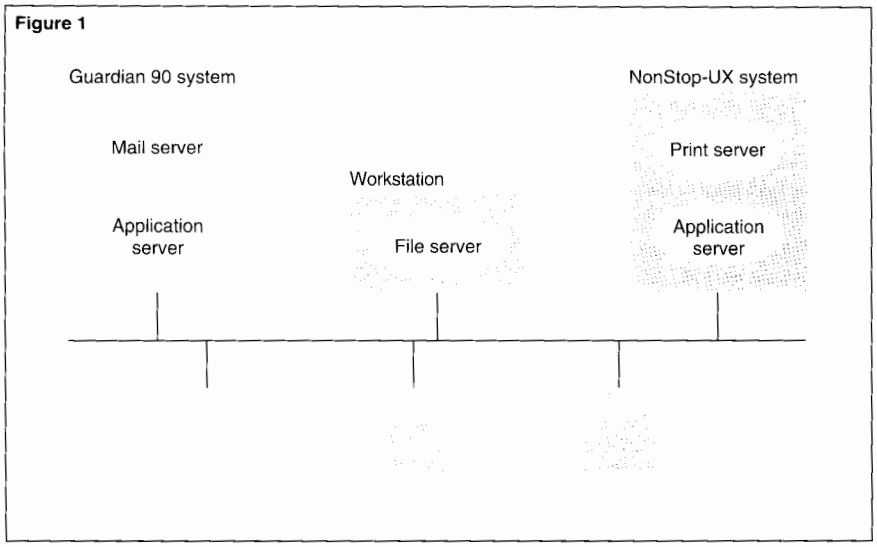


Figure 1.
With client/server, a networked user can access many different servers located on different hardware platforms.

The Shifting Computing Power Structure

Client/server computing evolved from computing architectures that originated in the 1970s. The need to allow user programs access to common services across the system gave rise to the server construct; mail servers and print spoolers are early examples.

The new components of client/server applications are workstations, local area networks (LANs), and open systems interfaces. In the ten years since the introduction of the IBM PC, workstations have proliferated in the workplace. There are an estimated 120 million desktop computers worldwide. In that time, standard configurations for desktop systems have increased exponentially. Desktop storage technology has evolved from 360-kilobyte floppy disks to 500-megabyte hard disks, standard random-access memory configurations have increased from 64 kilobytes to 32 megabytes, and CPU performance has improved by a factor of 20. Character-based interfaces are giving way to GUIs which are available on all workstations.

LAN installations have proliferated, providing the infrastructure needed for client/server applications. Over 60 percent of all workstations are LAN-connected. LANs have encouraged the interconnection of various hardware systems, stimulating demand for open systems interfaces, particularly in networking protocols and host APIs.

The balance of processing power is shifting toward the desktop. Client/server computing acknowledges this shift and is based on a user-centered paradigm. In theory, client/server applications see the system from the user's point of view. The user operates an application interface to access services from a pool of computing resources but does not see which system or systems are accessed in the process of performing a business task. (See Figure 1.)

Client/server architectures view the entire enterprise-wide computing structure as a single system. The theory implies that all users are connected to the same LAN and all systems are online and accessible, so that each user sees the same data representation of the current business situation. While not every client/server implementation will meet this ideal, the vision of a unified enterprise computing system, with all users having flexible access to data and services, illustrates the direction in which client/server concepts and technology are taking mainstream computing.

Motivations for Client/Server

The movement to client/server architectures is driven by two major forces: the desire to exploit workstation and LAN technology and the need to create information systems that can adapt quickly to new business requirements. The key characteristics of client/server applications allow organizations to meet both of these requirements.

Access

End users are generally limited to applications created for them by information systems (IS) professionals. This results in application backlog caused by the long time it takes to define, implement, and deploy applications for end users. With client/server architectures, IS professionals have several new, generally simpler and more flexible ways to address end-user needs.

Isolate Business Functions from the User Interface. Major business functions tend to change less frequently than the organizations that perform them. For example, users of home banking and automated teller machines are performing functions formerly done by bank employees. Client/server provides a model for taking advantage of this fact. Developers can implement business functions as servers, which can be used in different combinations by different client interfaces. IS groups will initially spend more time implementing servers, but after a critical mass of functions is available as host services, developers can address end-user needs more completely and quickly, creating new applications, products, and services as needed.

Integrate Heterogeneous Hosts at the Desktop. Large organizations often run mainframes and minicomputers from different vendors within a single division or department. As organizations now try to streamline business processes by giving workers access to all the information they need to complete a task, they must provide the ability for users to read and update information from heterogenous systems. Using a client/server architecture, a workstation can be the focal point for this integration. The user operates a single user-interface that opens communication sessions to each system.

Create Small Applications More Quickly. For every application scheduled for development, there are many more that are considered infeasible because the user base is too small or because the requirements change too frequently. If the users who need these applications manage to get data access at all, it is by requesting ad hoc queries of their IS department, which takes valuable time away from development personnel. The data is generally returned as lists on computer forms. This data is hard to analyze and often out of date.

Some forms of client/server allow for the creation of applications with little or no programming. This generally involves using workstation tools to dip into existing databases without writing host logic. Point-and-click query tools or high-level graphical development tools like Claris's HyperCard allow for the rapid creation of small applications. These can often be created in one or two days and may have a life span of only a few months or more. The effect is that more people are online, viewing the same data representation of the state of the business and increasing their ability to make decisions in harmony with business reality.

User Interface

Client/server brings to host services the benefits of GUIs, which are now available on every desktop computer. The most common ones (Macintosh, Windows, Presentation Manager, Motif, and OpenLook) have similar menu and window structures. Two characteristics of GUIs contribute to their superior ease of use. First, GUIs are based on visual principles. The user operates an on-screen pointer with a mouse, choosing commands by pointing at them rather than by typing memorized commands. Second, the operation of windows, menus, buttons, and fields is consistent between all programs; users can apply current skills to new applications, making programs easier to learn. For example, a user can apply GUI skills learned while using a word processor to an OLTP client/server application.

GUIs also make users more productive and more satisfied with their work. A recent study found that people running GUI-based programs completed 35 percent more tasks in a given time period than those operating an equivalent character-based program (Temple, Barker & Sloane, 1990). The GUI users also made fewer errors and were less fatigued and more satisfied than the group using a character-based interface. The study also concluded that GUI users were more likely than the other group to explore and teach themselves application capabilities.

Distribution of Processing

The modular nature of client/server lets designers shift off the host those portions of the application workload best handled by a workstation, while leaving on the Tandem system application logic and data that benefit from fault tolerance, transaction protection, and central storage. This strategy reduces host application workload, and more fully uses desktop computing power. In Pathway applications, for example, between 20 and 60 percent of application code is in the requester. In client/server applications, the function of the requester is moved to the workstation, relieving the Tandem host of nearly all of this processing.

Machine Independence

Client/server applications rely on a well-defined programmatic interface between the workstation and host systems. When the APIs used are standard or widely accepted, they are implemented on a variety of workstations and hosts. This fact makes it possible to move client applications to different workstations without changing the host application or to move server functions to different hosts without changing the workstation application. For example, the Oracle SQL*Net API is available on both Guardian 90 and NonStop-UX systems. As long as the client application uses facilities found on both systems, an application accessing data from a NonStop-UX system could be moved to a Guardian 90 system without any changes to the workstation application.

Client/Server Application Models

In client/server, application functions are distributed between the workstation and the Tandem system. Client/server applications can be characterized by the way processing is distributed. Three primary models describe distribution of processing.

- *Client-transaction server* is the model of choice for high-volume OLTP because it is based either on Tandem's Pathway OLTP environment or on compiled database programs under NonStop-UX.
- *Client-database server* is appropriate for end-user query and reporting access and light transaction processing because each database query is handled dynamically, offering flexibility but reducing throughput.
- *Client-presentation server* is best for applications that need an improved user interface but do not need to shift application logic to the workstation.

Each model is discussed separately to help readers analyze methods for distributing processing and to categorize products that help users implement client/server applications. To illustrate the distribution of processing in client/server applications, this article divides a typical business application into presentation, application logic, and data management.

The models for client/server are not hard and fast, however; a single application does not have to be implemented as all client-transaction server or all client-database server. In practice, more than one model is commonly used within the context of a complete application.

Client/server brings the benefits of GUIs to host services.

Client-transaction server is best applied to the relatively few applications in which transaction volume is great and the requirements change infrequently, such as order-entry. Client-database server requires far less effort to design and implement small applications; however, these applications generally do not perform as well as client-transaction server, so the client-database server model is applied to the more numerous, light-duty applications.

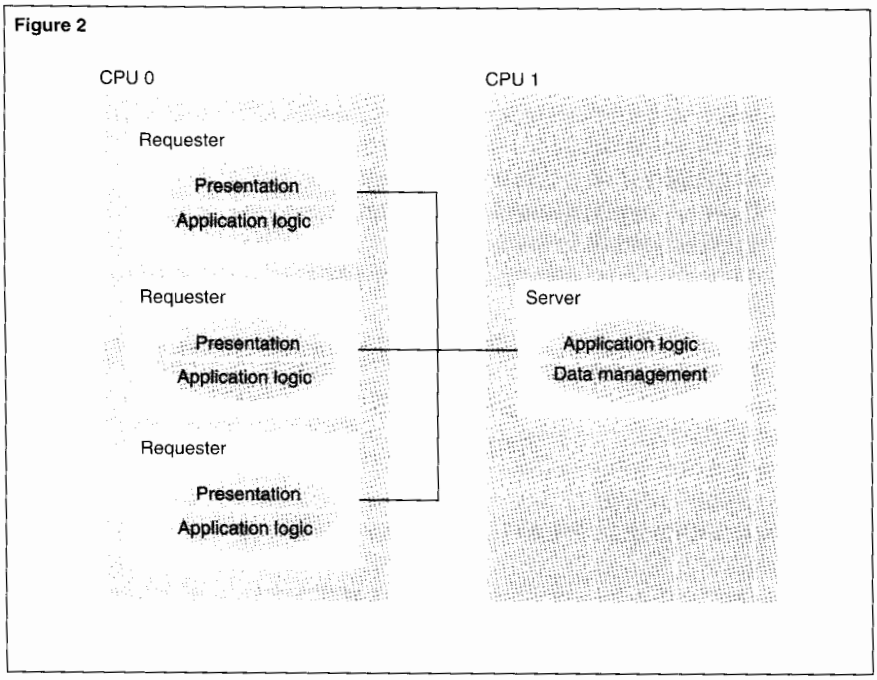


Figure 2.
Pathway requesters can access servers located on different CPUs of the Tandem system.

Client-Transaction Server

Tandem's Pathway environment, introduced 12 years ago, was the first software architecture to rely on requester-server for the purpose of OLTP. It is still the ideal way to deploy and manage transaction servers on Guardian 90 systems. The client-transaction model makes full use of Tandem's architectural strengths and can support high-volume OLTP transaction loads.

Pathway requesters contain the presentation services and application logic, while the servers contain some logic that defines business transactions accessing the database management system (DBMS). The requester communicates with the server using an interprocess message that indicates which transaction to perform. It also contains required parameters. The server executes the transactions and sends the results back to the requester for display. Pathway optimizes OLTP performance by running the requester and server processes in different CPUs and by allowing multiple requesters to access a single server. (See Figure 2.)

Client-transaction server follows a similar scheme, splitting the application in the middle. It puts presentation services and part of the application logic on the workstation and puts the rest of the application logic and database management on the Tandem host. (See Figure 3.) The server application logic defines one or more services, which represent one or more database transactions. The client sends a request to the server indicating which stored service to invoke.

Stored Procedures. Database server products often extend SQL to include transaction server features. A facility, usually called *stored procedures*, allows the application programmer to save blocks of statements representing transactions and simple logic in the database. When a particular transaction is needed, the DBMS calls it by name and executes it. When a database server is executing stored procedures, it is a form of client-transaction server.

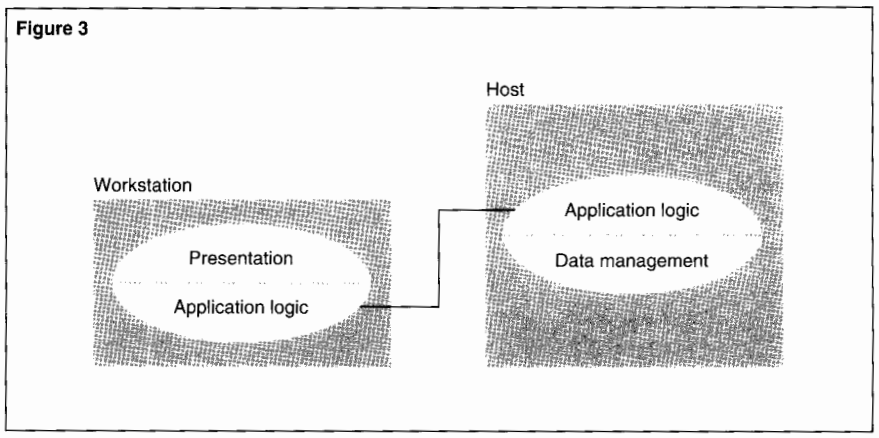
One can use existing Pathway applications in a client/server environment because the same servers can be used to support workstations as well as terminals. Similarly, one may think of Pathway servers as network-accessible business services that can be used in different combinations by different applications. For example, a bank with an existing Pathway-based teller application would already have servers for opening an account, transferring funds, and depositing and withdrawing funds from an account. The same servers could be used by a workstation-based client to implement a graphical application (in a kiosk or other convenient location) that bank customers use directly.

A characteristic of this model is that users are limited to the set of stored transactions in the server. This constraint has the benefit of offering only valid services to client applications. It also ensures database integrity because transactions are predefined, and presumably pretested, before making them available for clients.

Client-transaction server also isolates the client application from the database schema because the request messages invoke services, which in turn access the database. A change could be made to the database, yet the service name and message format would remain unchanged. Network traffic is reduced because a single client request can invoke a series of database manipulations.

Development is more challenging with client-transaction server, however, because programmers, often different people, usually write client and server processes independently, on different computer systems and with different programming tools. All the savvy of an IS group will be brought to bear on these applications, but the result will be powerful applications that meet the demands of high-volume OLTP.

Tools for Client-Transaction Server. Several Tandem products strengthen Tandem systems' ability to deliver OLTP in a client/server environment. The Remote Server Call (RSC) product enables workstations to send and receive messages from Pathway servers over asynchronous communication lines or over a LAN. Pathway Open Environment Toolkit (POET) is a GUI development environment built on top of RSC. Tandem Dynamic Data Exchange (DDE) Gateway provides a client services layer on top of RSC that allows any Windows application that complies with Microsoft's DDE specification to access Pathway servers. For more information, see the accompanying articles in this issue on RSC (Iem and Kocher) and POET (Culman).



On NonStop-UX, Tandem has announced support for the Tuxedo transaction processing monitor, which supplies some of the functionality of Pathway and TMF™ (Transaction Monitoring Facility) such as multithreading, server process management, and transaction protection.

Figure 3.
In client-transaction server, the workstation client invokes stored transactions in the host server.

Figure 4

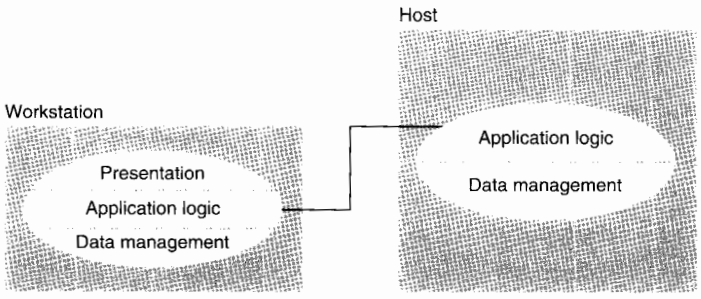


Figure 4.
With local data access, some data management functions are placed on the workstation, while critical data remains on the host.

Variation: Local Data Access

The client-transaction server model described above moves application logic down to the desktop but does not move data files. This variation puts some data management functions on the workstation, further relieving host system workload. (See Figure 4.)

The best candidates for local data access are static, read-only data, such as zip code tables or product codes, or live noncritical data that is required only by the user, such as a salesperson's lead tracking table. Users can merge this data with customer and product data from the server for reporting or on-screen display. The primary consideration is that data distributed to the desktop is generally not as well secured or protected from disk failures. If the data is well chosen, significant reductions can be achieved in network traffic and host processing.

Client-Database Server

In client-database server, the Tandem host acts as a database engine, receiving database queries from a client application, processing the query, and returning the results. (See Figure 5.) Typically, these queries are SQL statements. The industry movement to SQL as a standard enabled the rise of this type of client/server, but the model does not require SQL.

This approach offers a new ease of access to Tandem online databases. Programmers can write application interfaces using client tools only, such as workstation compilers, or higher-level tools such as Claris's HyperCard for Apple Macintosh computers or Asymetrix's ToolBook for the Microsoft Windows environment. End users can access databases using point-and-click query tools that require no programming and can integrate the data they receive into their spreadsheets, word processors, and other desktop programs.

Host programming is unnecessary with this model. The tools for access are often so intuitive that users who may have been requesting a custom application from their data processing department may be fully satisfied with a client-database server application that requires little or no setup or configuration help.

To gain this level of access, however, the host dynamically processes the SQL query, which must be compiled before executing, slowing performance. Further, there are generally several processes started per user connection, which also affect performance and expandability. This typically limits the use of the database server approach to applications in which the total number of users is small and the database access is limited.

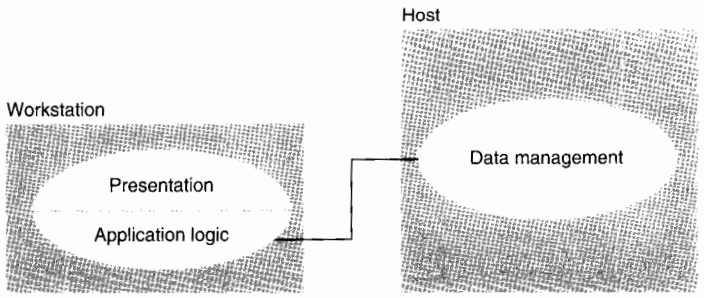
For example, this model is appropriate for simple database update applications, decision support, and reporting applications. In addition, there is no way to enforce database integrity except by limiting access to read-only (as with query-only tools and report writers) or ensuring in the client program that all submitted database statements leave the database in a consistent state.

Because of the openness of the database server approach, it is important to carefully administer its application and use in production environments. Client-database server is not for the majority of users; it is best applied to provide data access to user groups that do not have the size or rigid requirements to warrant development of a client-transaction server application.

Although SQL is a standard, vendors have implemented their own extensions to the standard, including protocols for network access. Many off-the-shelf applications such as query tools, spreadsheets, fourth-generation languages, and compilers have been designed to communicate using these extensions to SQL. For example, on Guardian 90, Tandem's SQL Server Gateway, Oracle, Focus, and Data Access Language (DAL) products are available. On NonStop-UX systems, Informix, Oracle, Ingres, Progress, Unify, Empress, Unidata, and others are available.

On Guardian 90, these products have been implemented as gateways to NonStop SQL; they provide the network API and translate vendor-specific SQL queries to NonStop SQL. While NonStop SQL keeps close to the ANSI standard, there are some extensions and omissions. As a result, some vendor-specific features may not be available through the gateway. On NonStop-UX systems, each vendor's entire database product has been ported, ensuring complete compatibility.

Figure 5



Client-Presentation Server

Client-presentation server is analogous to traditional terminal-based applications because the program running on the Tandem host drives presentation services located on the workstation. It is best used for applications in which a GUI is required, but the IS department does not want application logic distributed to the workstation. Another use for client-presentation servers is the presentation of applications in which screen layouts change frequently.

Figure 5.
In client-database server, the host acts as a remote data management service for a workstation-based application.

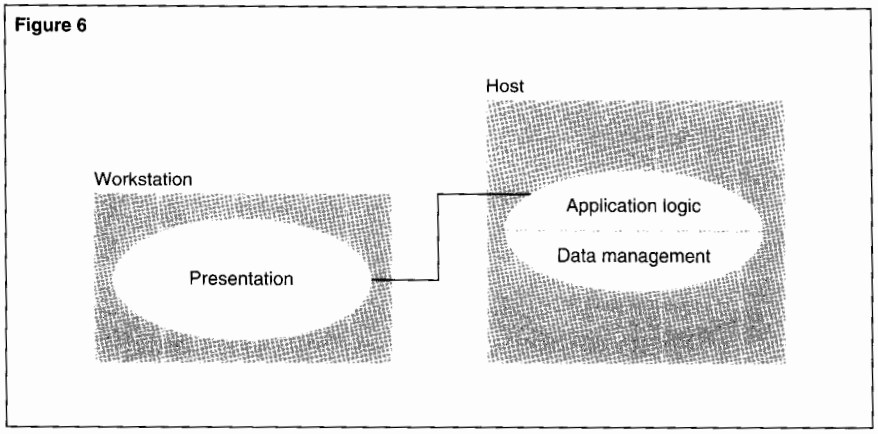


Figure 6.

With client-presentation server, the workstation draws the user interface based on instructions it receives from the host.

In this case the construct is reversed. The server is the workstation-based presentation service that responds to messages from the client running on the Tandem host. (See Figure 6.) Client messages describe what to display on the workstation screen. Consider, for example, a NonStop-UX system displaying a graphical Motif application on X-windows terminals. The window manager running on the X-windows terminal is the server responding to requests from a remote client.[1]

[1] Driving X Windows protocols directly from Guardian 90 systems would not be recommended; because of the transaction orientation of Guardian 90, the high rate of user interrupts to the I/O processor degrades transaction performance.

Depending on the product or products used to create a client-presentation server application, the type of messages sent by the host will vary. With X-windows, the messages may describe the interface at a low level, such as window frames, lines, and text. Any Language Any Computer (ALAC) is a protocol that describes objects at a higher level, such as menus, fields, and movable and resizable windows.

Both protocols can run on all three workstation types, and the network can combine different types of workstations. With ALAC, the application can even store window definitions at the workstation and invoke them by name, rather than issuing a full description of their layout. Software of this type reduces the number and size of network messages and is therefore more suitable for the Guardian 90 operating system, which is optimized for message-based OLTP.

Client-presentation server offers the use of a highly graphic interface that can work in conjunction with workstation productivity applications. Programmers need only write application code for the Tandem system; any changes made to the host application appear on the workstation, without updating workstation software. All control is centralized, as with traditional host-based applications. The display performance of a remotely controlled interface, however, can often be more sluggish than other types of client/server. Another disadvantage is that client-presentation server does not fully exploit the workstation CPU; it merely uses the workstation as a display service.

Because it does not truly embrace the spirit of the client/server distributed model, client-presentation server is an ideal way for IS departments to create applications with modern interfaces, yet avoid many of the new challenges that come from distributed processing.

Variation: Frontware

Frontware is software that runs on a workstation and provides tools for mapping a GUI to an existing terminal-based application. (See Figure 7.) The user interacts with a graphic display, while the frontware sends the user's input to a hidden terminal-emulation session. The effect is that the host system "sees" a terminal, and the user sees a graphical interface. MitemView for Macintosh and Easel for PCs are frontware products that work with Tandem applications.

With frontware there is no need to modify the existing application. Frontware is most often used to add the benefits of a GUI to a host application that, because of its monolithic structure, cannot be modified easily to use a more direct client/server approach. Pathway applications are not monolithic, so frontware is not an effective solution for those applications.

The best use for frontware in the Tandem environment is to create graphical interfaces to system utilities and TACL™ (Tandem Advanced Command Language) sessions, or to create a single unified interface for several host applications. To create a single interface for several terminal applications, frontware emulates several terminal sessions from the various host systems simultaneously, and maps the fields, data, and commands onto the graphic screen. When the user makes changes to the data on the screen, the data is sent back to the appropriate terminal session. The result is inexpensive desktop hardware and software supporting the integration of complex host applications.

With frontware no work is shifted from the current host application, so performance is no better than and sometimes worse than if the application were running on a terminal. Weak performance is somewhat offset by improved user productivity, since users can understand and enter data more easily and accurately. The GUI, however, is largely limited to following the screen layouts and screen navigation of the host application. Additionally, any changes made to the host application must also be made in the frontware client.

Frontware is a stopgap technology allowing the integration of workstations into host environments without changing the host environment. It is not a strategy for creating production-level client/server applications; however, it is appropriate when time and cost do not justify implementing one of the other forms of client/server.

Figure 7

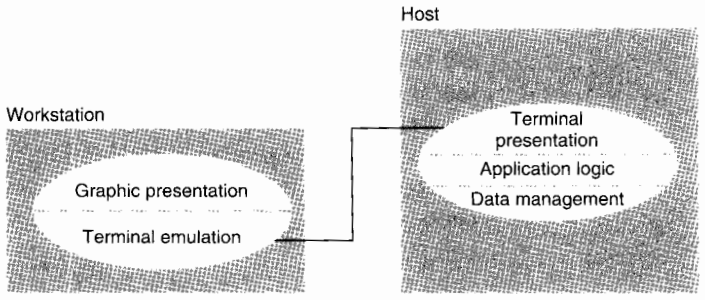


Figure 7.
With frontware, the workstation runs a standard terminal emulation module as well as presentation services.

Development Considerations

Client/server applications are functionally the same as traditional business applications in that they automate business processes. However, they differ from traditional applications because processing is distributed, which increases the number and types of system components used to build the application. Major new components, like workstation operating systems, GUIs, and LANs, must now be integrated. IS groups need to consider several development issues to build applications successfully in this new environment.

Requirements Analysis

The requirements analysis for a client/server application should cover both business and technical requirements. Business processes, target users, and target systems (both workstation and host) need to be detailed. This is a traditional requirements analysis, except that client/server technology offers the possibility of implementing a system-wide application that serves the workflow for the entire enterprise.

Most organizations own a number of departmental workstations on LANs and central host systems. Often different departments have different desktop systems, LAN hardware, and minicomputer systems. An inventory of these systems is important input into the requirements analysis, particularly if the organization intends to use client/server technology to unify the corporate computing structure.

Prototyping

Creating a prototype application is an effective way of refining business requirements and user-interface design requirements. It allows target users to see if developers have assessed their needs properly. A prototype can be created with the same tools used for the implementation phase or with more rapid development tools, which allow for a quicker development and test cycle. Ongoing refinement of the user interface is the best way to ensure that businesses reap the full benefit of moving to GUIs. If the prototype and implementation phases use the same tools, the prototype can actually be a head start for the implementation phase.

User Interface Design

Even with a graphical windowing system, one can design a poor human interface. In fact, the dramatic increase in design possibilities for a GUI over a terminal user interface increases the level of skill required to create an acceptable interface. When scheduling application development, allow a generous amount of time for interface design. The interface design process should be iterative and include user testing on the prototype. A user test may be as simple as taking notes while observing a user interacting with the application or as rigorous as recording times to complete specific tasks and errors for each user.

Transforming a terminal-based application to client/server is a relatively straightforward procedure. One simply puts fields and field-label arrangements in a window and converts function keys to graphical push buttons or pulldown menu items. In this case, the choice of fonts, colors, screen layout, and names of menu items are the critical issues.

A more substantive use of the workstation's capabilities is to employ direct manipulation and metaphors. Suppose, for example, a user wants to make an airline reservation. Instead of filling in the cities of origin and destination on a form, the GUI user might see a map, drag a small picture of a plane from the city of origin, and drop it on the city of destination. The direct manipulation is the act of moving the plane, and the metaphor is the map of the area.

The design and testing of GUIs is challenging work. The best user-interface design most often comes from human factors engineers schooled in user-interface design and testing methods.

Tool Selection

In traditional Tandem terminal-based development, all of the application code resides on the host processor, and Tandem supplies nearly all of the system software and many application tools. In client/server applications, approximately half of all application processing resides outside the Tandem host boundary, and outside vendors supply much of the client software available to create applications.

Before choosing client/server tools, developers should know, at the very least, which hardware, system software, and communication methods, for both desktop and host, the application requires. They also need to consider the type of user interface they want to create. Some tools allow the creation of simple fields and buttons on form-oriented screens. Others allow the creation of more sophisticated multimedia interfaces, using graphic images, sound, and animation. Some workstation development tools allow users to develop C applications that can be ported easily between Macintosh, Windows, Presentation Manager, Motif, and OpenLook. The tool also must support the API chosen to connect the client interface and the Tandem environment.

Client/server tools are in the early stages of their evolution. Currently, there are few computer-aided software engineering (CASE) tools that explicitly generate client/server applications. Most of these tools are separate workstation-based development environments that simply plug in to a particular API on the Tandem system. These frequently are not designed for work group development environments, with source check-in and check-out. The current technology environment for creating client/server applications requires careful design and implementation planning and consideration of the processes for managing source code versions.

The design and testing of GUIs is challenging work.

Tandem provides APIs and communications protocols that support client/server on most workstations and operating systems, using standard networking protocols. RSC is available on DOS, Windows, OS/2, and UNIX, using standard networking and asynchronous protocols.

Tandem supplies tools on the Guardian 90 and NonStop-UX operating systems. POET is Tandem's client-transaction server development environment for Guardian 90. It simplifies the integration of RSC with workstation-based tools such as the Caseworks product CASE:W. On NonStop-UX, Tandem supplies and supports the full product sets of database vendors, including tools for creating client/server applications.

Programming Personnel

By including workstations and networks, client/server computing represents a new expanded systems environment. IS professionals must have the information to understand the new environment and the skills to write applications for it. New expertise in LANs, workstation operating systems, GUI programming, and user-interface design is called for. Effective retraining of personnel is the single most critical item in creating successful client/server applications. Most structured analysis and design methodologies are still valid in this environment, but the implementation, including decisions on distribution of processing, programming in C or new fourth-generation languages, may require new skills.

Security

In Pathway terminal-based systems, a dedicated wire connects each user to the host. Typically, the terminal connection is preconfigured to run one production application, and the password protection is handled by the application.

In Tandem client/server environments, users may connect to any number of printers, file servers, and Tandem database or transaction servers. Each LAN service available has a security model protecting access to it. For example, the DAL database server and the Tandem SQL Server Gateway rely on Guardian 90 security to provide protection. On the other hand, the Oracle gateway product has its own security system that conforms to Oracle standards. These service-based security checks are in addition to network operating system security.

A further consideration is that all users are connected to the same wire in LAN-based client/server environments. Packets of information running across a LAN are usually not encrypted, so any workstation on the LAN can examine packets and display their contents. Additionally, after password logon, any program that is aware of the API being used and the format required can send messages to host servers.

Defensive programming techniques are required to address these additional security problems. One solution is for client/server programs to agree on a unique token that is passed back and forth before each message.

Client Application Distribution

In every client/server implementation, some application software must be installed on the workstation. A daunting new challenge of client/server is maintaining version synchronization of client and server programs, as well as updating the client when versions change.

This issue has been handled in many ways. A simple solution is to have the workstations launch the client application from a file server, which is managed as a central resource by the IS department and always has the latest client software.

A file server, however, is not always a possibility, particularly with X.25 or asynchronous connections. Another solution is to have the client application perform on startup a version check with the server application. If the client version is old, a file transfer starts, putting the new client software onto the workstation.

System Operations

In client/server, all of the host systems, network equipment, and workstations comprise a single computing environment. Management of a production client/server environment is necessarily more complex than a Guardian 90 or NonStop-UX system driving terminals over asynchronous lines. Tighter integration of network management and systems management is required. The industry movement is toward central management of client/server environments, based on a single system control center instead of a LAN and workstation expert at each department.

System administrators should consider the following key issues:

- Tracking network errors from hosts, networks, and workstations and having a clear picture of where they are and what they mean to application performance.
- Fixing network device and workstation problems remotely.
- Administering workstation configurations, especially if users also need to install and run their own software in addition to their IS application.

Conclusion

Client/server architectures make better use of the substantial computing power sitting on end users' desks than do traditional, terminal-based applications. Local data management and GUIs, some even using multimedia features, point to application possibilities that are just beginning to be explored. IS groups must be willing to think imaginatively and to use new tools in order to realize this potential.

Cooperative applications based on Tandem systems can be built today using connectivity and application development software provided by Tandem and third-party providers. Tandem's traditional strengths of fault tolerance, data integrity, and high transaction throughput offer an ideal system environment on which to implement client/server applications.

References

Client/Server and Cooperative Processing: A Guide for the Perplexed. 1991. *Software Management Strategies.* Gartner Group Inc.

Client/Server: What, When, and How. December 1991. *Information Week.*

Dekkers L. Davidson. 1990. *The Benefits of the Graphical User Interface.* Temple, Barker & Sloane.

Acknowledgments

I would like to thank the reviewers of this article for their many suggestions and technical insights. Special thanks are due to David Cooper for his help on security issues.

Howard Cooperstein is a senior systems analyst in Tandem's Application Technology Center working in the areas of workstation integration and graphical user interface design. Howard joined Tandem in 1986 as a software developer.

Implementing Client/Server Applications Using Remote Server Call

For the past twelve years, Tandem™ systems have provided an environment for high-volume online transaction processing (OLTP) applications. Today many businesses are interested in taking advantage of the processing power of desktop workstations through client/server computing. Tandem users can implement OLTP client/server applications using workstations, Remote Server Call (RSC) software, and Guardian™ 90 systems.

As with traditional Tandem systems, client/server on Tandem is built on the Guardian 90 message-based operating system. Although not required, the Pathway transaction processing system, with its requester-server model, and TMF™ (Transaction Monitoring Facility), used for data integrity, provide an enhanced environment for client/server.

The client/server model of choice for OLTP is client-transaction server, which is based on a requester-server model. In this model, some business data processing and presentation services are placed on the workstation, taking advantage of the workstation's capabilities, while those business functions best treated as a shared resource are located on a local area network (LAN) or host server. (Cooperstein, in this issue of the *Tandem Systems Review*, presents more detail on this and other client/server models.)

This model implies the need for a mechanism to enable the client and server to communicate with each other. In a Tandem environment, the Guardian Message System manages the communication of messages between the requester and server. In a Tandem client/server environment, with client programs on workstations sending messages to Guardian 90 servers, that communication is handled by RSC.

Remote Server Call represents a transactional remote procedure call mechanism. It provides a workstation- and transport-independent application programmatic interface (API) that allows client programs to send requests to, and receive replies from, Guardian 90 server applications within client-defined units of work, or transactions. (For a selected list of RSC API calls, see Table 1.)

This article will help users implement client/server applications using RSC. Beginning with an overview of RSC, the article guides users through the implementation process: planning, developing, and managing RSC applications. Specific topics include client tools, installation and configuration, client use of RSC, problem management, testing, RSC management, and implementing services on the client.

The application discussed in this article is assumed to be one with high-performance OLTP requirements. Readers are assumed to be familiar with Guardian 90 architecture and Pathway concepts.

Functions and Components of RSC

In a client/server application, functions are divided between the client and the server. Technically speaking, the client and server are part of a single application. For purposes of this article, however, the client side of the application is referred to simply as the client.

The client programmer has several responsibilites in creating the client. These include both general and RSC-related functions. Under the heading of general responsibilities come these functions:

- Presentation services.
- Data collection.
- Editing and data validation.
- Local and LAN database access.
- Appropriate application logic.
- Security.

Table 1.
Selected RSC API calls.

RSC API calls	Function
TdsConnect	Connect to TDP on Guardian 90 system
TdsBeginSession	Begin an RSC session
TdsBeginTransaction	Begin TMF transaction
TdsWriteRead	Send message to server (and get reply, in waited mode)
TdsIoCheck	Complete nowaited TdsWriteRead
TdsEndTransaction	End a TMF transaction
TdsEndSession	End an RSC session
TdsDisconnect	Disconnect from the TDP

The following functions are related to RSC:

- Transaction boundary begin, end, and abort.
- Messages sent to and received from servers.
- Interprocess message construction and data conversion.
- Session establishment and management.
- Error recovery.
- Security.

The server is responsible for database services such as database integrity, other services such as centralized error reporting and security, and appropriate application logic. The server may also need to manage unsolicited messages, which may be sent by the server to a client.

Table 2.
API implementation on supported workstations.

Workstation	RSC implementation
DOS	Device drivers, C library
Microsoft Windows	Device drivers, Dynamic Link Library (DLL)
OS/2	Background processes, DLL
UNIX	Background processes, C library

RSC consists of components on both the workstation and Guardian 90. On the workstation, the RSC API is implemented differently on various workstations. (See Table 2.) While the implementation is different, the API is the same for all RSC-supported workstations. On the Guardian 90 side, RSC consists of the Transaction Delivery Process (TDP), a command interpreter called TDSCOM, and, for security, a user-provided access control server. Figure 1 shows how RSC is implemented on a workstation and the Guardian 90 host.

RSC Client Components

For the client to send messages to and receive messages from the application servers, it must first establish a network connection using the appropriate RSC calls (TdsConnect, TdsBeginSession). These calls identify the host system and optional Pathway system controlling the servers. When the client is ready to send a message to a server, it must perform the necessary data conversions. The client must also package the data into the proper message format for the server. Once these tasks are completed, the client may then use the appropriate RSC call to send a message to the server (TdsWriteRead). When the server replies, the client must perform any necessary data conversions before using the data.

RSC Host Components

On Guardian 90 systems, RSC, through the TDP component on the host, provides routing to Pathway server classes (through Pathsend) for context-free servers; Pathway Intelligent Device Support (IDS) requesters for enhanced routing or security capabilities; and other Guardian 90 named processes for context-sensitive servers and other purposes. The TDP handles communication protocols, thereby isolating the servers from the network. The servers are not aware they are communicating with a workstation client; for them, the interface is the standard Guardian 90 process-to-process ($RECEIVE) interface.

The result is that programmers do not have to write new servers for client/server applications. Existing Pathway servers may be used by workstation-based clients, and workstation clients and terminal-based requesters may share the same servers. In addition, one TDP may provide routing for clients running on supported workstations, even if they are running different communication protocols.

Unsolicited Messages

Occasionally it is desirable for a server to send information to a client. For example, an electronic mail system running on Guardian 90 could notify a particular client that it has received a mail message. In another example, a centralized Guardian 90 service could send stock price updates to all workstations currently running a stock price application.

In these situations a Guardian 90 process or Pathway server sends an unsolicited message to a connected client. If the application on the workstation is to use unsolicited messages, the workstation client must frequently check for unsolicited messages from the host system, using the RSC call, TdsUmsCheck.

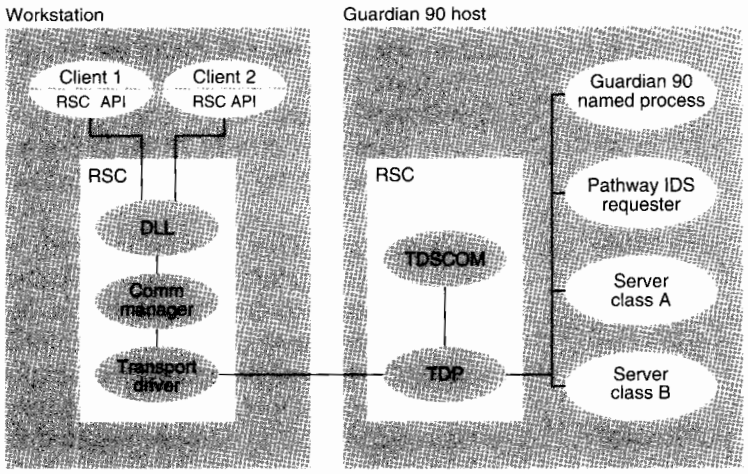
Upon receiving an unsolicited WRITEREAD message, the workstation should respond to the sender; if the sender used a WRITE message, the client does not need to respond. This facility can greatly enhance a client/server application when the server or database has changed and this information needs to be broadcast to client applications.

Networks and Protocols

Client/server communication involves many kinds of networks and protocols. A single application might run on some workstations connected to a Guardian 90 system through a Transmission Control Protocol/Internet Protocol (TCP/IP) LAN, while other workstations might be connected through dial-up communications or an X.25 network. Some workstations may be connected in several ways concurrently. In addition, the network may change. For these reasons, the application should be isolated from the network topology and its corresponding protocols.

RSC creates this isolation. It provides a single API to the application regardless of the network protocol. Using RSC, client programmers have the architecture to use multiple protocol connections transparently. This is useful for fault tolerance if, for instance, a particular network service is unavailable. Currently, RSC supports asynchronous, NETBIOS, TCP/IP, and X.25 protocols.

Figure 1



Planning for RSC

Preparing for client/server using RSC involves deciding on an appropriate application and overall architecture. Programmers should also work with end users to prototype a graphical user interface (GUI). Only then can programmers get down to specific predevelopment tasks. These tasks include selecting the tools to create the client, installing and configuring RSC, and validating that all of the components are working.

Figure 1.
Components of RSC in a Microsoft Windows environment.

Client Tool Integration

The first step in implementing client/server with RSC is choosing the tools to create the client. The tools used to create the Guardian 90 servers are not of interest here since RSC on the Guardian 90 system uses the standard message system interface ($RECEIVE) to communicate with Pathway servers and other Guardian 90 processes.

Hundreds of client tools are currently available to help programmers develop the client; however, not all of these tools support the necessary programming capabilities to interface with RSC. There are three important requirements for a client tool. It must be able to define a structure in order to describe the interprocess message format, call a C library (or a Dynamic Link Library in the case of Windows and Presentation Manager), and handle and manipulate non-ASCII data.

The use of COBOL servers on Guardian 90 may impose some additional restrictions. For instance, the interprocess message defined by a COBOL server may contain data types, such as display-numeric, which may be difficult for a client tool to handle. The Guardian 90 COBOL compiler may also implicitly include fillers in an interprocess message in order to maintain alignment rules. Tools that cannot handle such structures are more difficult to use with RSC. Finally, COBOL represents strings with trailing spaces; other languages and tools often do not, so a conversion from one format to another may be necessary. Even if a particular tool cannot use RSC directly, it is usually possible to address such issues with a layer between the client application developed with the tool and RSC.

Installation and Configuration

After developers select a client tool, the next step is installing and configuring RSC. Installation consists of copying the appropriate software to the workstation and to the Guardian 90 system. Installation on the workstation also involves selecting the desired network device drivers and making adjustments to startup files, such as CONFIG.SYS in the case of DOS.

RSC configuration parameters exist both on the workstation and the Guardian 90 host. On the workstation, the configuration includes an .INI file that holds configuration options. Besides network configuration parameters, the .INI file contains the workstation name, the name or address of the TDP to use, the name of the Pathway system to use, and TMF options. These values are used as defaults when a client calls the TdsConnect, TdsBeginSession, TdsWrite, and TdsWriteRead procedures.

RSC also provides a mechanism for a client program to alter these default values while the client is executing. For production applications, this feature is normally required to isolate them from configuration changes on live server systems.

This is accomplished by calling an RSC procedure (TdsCreateOptions) to establish a private configuration area (called an options structure) for an individual client. Other RSC procedures (TdsLoadOptions, TdsSetOption) are then used to load the options structure.

On the host, the developer can configure the TDP using the TDSCOM command interpreter either interactively or with an input file. Configuring the TDP consists of defining which network the TDP should use and which workstations are allowed to communicate with the TDP. Once these objects have been defined, they may be activated. It is not necessary to stop the TDP process in order to make changes to any of the configuration information.

Validating the Installation

After installing RSC, it is a good idea to verify all the components are configured correctly. RSC includes a sample application consisting of an *echo* server for the Guardian 90 system and a simple client. The server echoes back any message it receives from the client. This process validates that the network and RSC are installed and configured correctly.

Using RSC in Client Programs

Several of the RSC-related functions in the client are similar to those in Pathway requesters. These duties include establishment and maintenance of sessions, message construction and data conversion, transaction support, sending and receiving messages, error recovery, and security. Of these, only establishment and maintenance of sessions will be unfamiliar to programmers of Pathway requesters.

Many of these duties are common for all RSC clients. For instance, of the responsibilities listed above, all clients may need the same kind of session establishment and management, error recovery, and data conversion routines. Since the code to handle these tasks is necessary in most clients, a common practice in this environment is to write a client services layer, which handles these common functions and services.

Connections and Sessions

When a client requests connections and sessions, RSC allocates communication resources for the client to use in subsequent RSC calls. In addition, RSC checks the validity of the workstation and, optionally, the user.

The RSC procedure TdsConnect establishes a connection with a TDP on a Guardian 90 system. Entries in the options structure determine which TDP and Tandem system will be used. This connection constitutes the equivalent of a physical connection to one TDP. A single client may connect to more than one TDP, and multiple client applications on the same workstation may connect to the same TDP. When a client requests a connection, the TDP checks that the client's workstation name (configured in the .INI file) matches one of the terminal names specified in the configuration of the TDP.

Only a few RSC-related functions in the client are unfamiliar to Pathway programmers.

After connecting, the client must begin a session before it can send messages to a server on the Guardian 90 system. This is accomplished by using the RSC call TdsBeginSession. Three session types are supported by RSC: interprocess, which communicates with Pathway servers through Pathsend and with Guardian 90 processes; Pathway IDS, which uses a user-provided IDS requester to communicate with Pathway servers; and Pathway, which uses a standard IDS requester[1]. For sessions dealing with Pathway, the TdsBeginSession call establishes the PATHMON name controlling the default Pathway system.

[1] The Pathway session type is provided for backward compatibility reasons only and will not be discussed further in this article.

Each RSC connection may support multiple RSC sessions for all protocols. In an asynchronous environment, this allows multiple clients on one workstation to access Guardian 90 servers over the same line. (Without RSC, it would take extensive coding to provide this capability.) Regardless of communications protocol, multiple sessions also allow a client to have multiple TMF transactions in use at the same time since each session may have one TMF transaction in progress at any time.

The use of the TdsBeginSession call also allows the client to send a user identification and password to the TDP, which then forwards them to an optional user-written access control server (ACS). It is up to this server to authenticate the user and tell the TDP if the user should be allowed to continue. The client should use the TdsEndSession and TdsDisconnect calls to terminate sessions and connections.

Message Construction and Data Conversion

Before using RSC procedures to send a message to a Guardian 90 server, the client program must construct a buffer in the interprocess message format expected by the Guardian 90 server. In addition, the client program must perform appropriate data conversion.

To convert data between different types of hardware, RSC provides two API calls to convert two common data types. The TdsSwapInt and TdsSwapLong procedures swap the position of bytes in an integer and long integer, respectively. These procedures are used when the ordering of bytes in an integer or long integer is reversed between the workstation and the Guardian 90 host (which is the case for all IBM PC-compatible workstations). To convert any other data, the application must provide its own routines.

The client program will receive a reply from a Guardian 90 server in a format defined by the server. It is therefore the client's responsibility to parse the buffer and perform appropriate data conversion on the data. TdsSwapInt and TdsSwapLong may be used in this situation as well.

Transaction Support

For transaction definition, programmers can choose to either begin and end transactions automatically, whenever a TdsWrite or TdsWriteRead is used, or explicitly, through RSC calls.

If the scope of the transaction is a single-send operation to one server, the automatic begin-end facility reduces network traffic, thereby improving performance. For those situations where a multiple-send operations must be protected by the same transaction, RSC supports calls that explicitly begin (TdsBeginTransaction) and end (TdsEndTransaction) a transaction. The scope of the transaction includes only that work done on the Guardian 90 system; it does not extend to local data.

Regardless of the options selected, the TDP will not abort a transaction unless the session with the client fails. There is no automatic abort facility; it is the responsibility of the application (client or server) to abort. For a client to abort a transaction, the RSC call is TdsAbortTransaction. The server would use a Guardian 90 procedure call.

Sending and Receiving Messages

The client sends and receives messages through RSC. When RSC connections and sessions have been established, the TdsWriteRead procedure can be used, in waited or nowaited mode, to send a message to a Tandem Pathway server or Guardian 90 process.

In waited mode, the client is blocked until the client receives a reply from the Tandem system. In nowaited mode, the client uses the TdsIoCheck procedure to complete the TdsWriteRead call and receive a reply message. The client must call TdsIoCheck periodically to poll for the TdsWriteRead completion. (Unlike the Guardian 90 procedure AWAITIO, TdsIoCheck will not wait for the completion.) The TdsWrite RSC call can be used when no reply data is expected.

Error Recovery

The client initiates connections and sessions with Guardian 90 servers; it is therefore the responsibility of the client to reestablish the connections, sessions, and application context if an error or failure occurs. The client must detect error conditions and respond according to the type and severity of the error. Using RSC, the client application programmer may call TdsErrorInfo to determine this information. After evaluating the error, the application programmer may programatically remedy the situation and bring the client and server into a known state.

The client sends and receives messages through RSC.

Security

Since the client and server communicate with each other through messages, security can be an important consideration. Whenever a client and server do not execute on the same system, there is the possibility for a security breach. There are several levels of security to be considered in this environment: physical security of the workstation, transport (network) security, and application security.

Physical Security of the Workstation. In a traditional terminal-based application, security was typically implemented in the host and little attention was paid to the terminals. Terminals have no intrinsic computing value, and therefore pose no security risk without a connection to the host. However, workstations do have intrinsic computing value, and may in fact store valuable corporate information. Therefore, controlling physical access to a workstation may be an important part of an overall security plan.

Controlling the physical access to a workstation may be accomplished in several ways. The method chosen depends on the degree of security required. Options may range from locked disks and screen savers with password protection to diskless workstations and locked rooms.

Transport Security. Transport security depends on the communications protocol and the medium being used. For instance, asynchronous communication is a point-to-point protocol, and the medium is usually point-to-point as well. In such an environment, data transferred between a workstation and a host is isolated from other users. However, it is now possible to have an asynchronous connection that uses LAN wiring. This kind of shared medium environment allows more security breaches than does a point-to-point environment.

It has been said that users should not put anything on a LAN that they would not want posted on the side of a barn. Intrinsically, LAN protocols allow every workstation to see everything sent over the LAN. In addition, some debugging and monitoring devices can display all of the information passing through the LAN. Data encrypting may be used to protect data in this environment. This solution is outside the scope of RSC but may be provided by hardware devices or by the application.

Application Security. As described earlier, RSC provides for authentication at connection and session establishment. It may also be desirable for the application to include an authorization mechanism in every message. RSC, through the TDP, provides the flexibility for a client to send to any Pathway server or Guardian 90 named process; however, this may not be appropriate for all applications.

Using Guardian 90 security and, optionally, Safeguard™, the Tandem system protection product, the TDP may be restricted from accessing Pathway servers and Guardian 90 processes. This restriction is based on the Guardian 90 user identification inherited by the TDP when it starts. If any client needs to access a process or server, however, Guardian 90 and Safeguard security are no longer useful. The TDP must be allowed access to these processes; therefore, it becomes the application's responsibility to provide an authorization scheme.

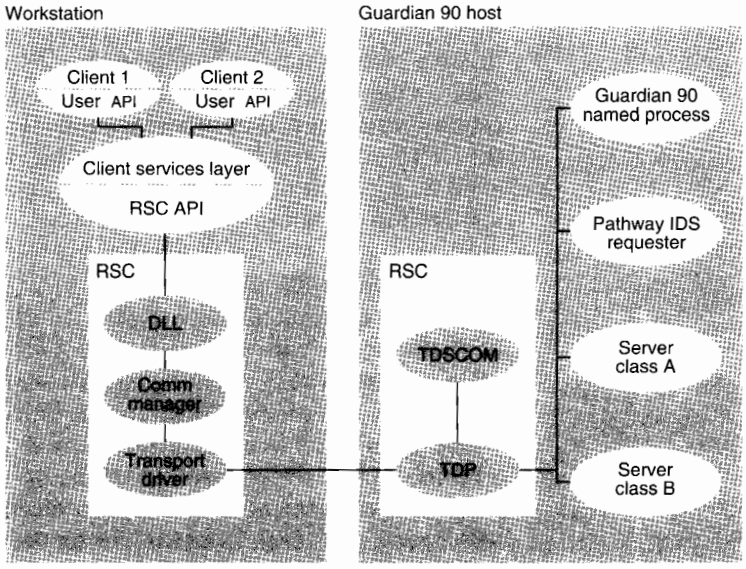
There are several appropriate authorization schemes. One simple scheme includes an encrypted user identification (or an equivalent token) in each message. A server could use this token to verify that the client is authorized to use its services. The article by Culman in this issue of the *Tandem Systems Review* addresses this question in more detail.

Implementing Services in the Client Environment

Client duties can be handled in a user services layer that deals with common functions and services. This architecture has the advantage of isolating the clients from code that may change. It may also make the client developer's interface with the network and host servers even easier, since each client programmer will not be required to know the details of RSC. It should improve quality by reducing the amount of new code that has to be tested and debugged. Another benefit is that this strategy uses workstation resources more effectively by creating a single shared object for all clients to use.

In addition, if the client development tool or language of choice does not conveniently interface to RSC, this kind of services layer may be used as a bridge between the two products. The support layer can ease the migration to an industry-standard API such as Distributed Computing Environment/Remote Procedure Call (DCE/RPC), a proposed industry standard from the Open Software Foundation.

Figure 2



The investment in a client services layer should pay for itself as more client applications find their way to the workstation. Several possible interfaces to such a support layer include Dynamic Link Library (DLL), Dynamic Data Exchange (DDE), Object Linking and Embedding, and a process for UNIX and OS/2. (Figure 2 illustrates a client services environment under Microsoft Windows.)

Tandem provides two products that are examples of such an approach: the Tandem Dynamic Data Exchange (DDE) Gateway and Pathway Open Environment Toolkit (POET). These products are both built upon RSC.

Figure 2.

In this example, the client services layer is implemented as a DLL in a Microsoft Windows environment. The client API is defined by the programmer based on application requirements.

System Management

Managing distributed systems is a challenging task for an organization to undertake. In this environment, applications are split up and usually reside in two or more locations. In the OLTP environment, synchronization and availability of computing resources is imperative to the operation of the application.

Problem Management

Due to the distributed and complex nature of client/server, with workstations, LANs, wide area networks (WANs), and host servers interacting, it is often difficult to determine which component or components caused a failure. It is therefore useful to think of problem management as four steps: detection (discover a problem), isolation (locate the source of the problem), recording (identify the nature of the problem), and recovery (determine strategies to fix the problem).

Detection. In many cases, detecting a problem is fairly straightforward. Most hardware and software components report unexpected or unusual conditions. The client application programmer should use RSC error facilities to detect RSC-related errors. Other errors may be detected by an operations staff that monitors the host system from a centralized console.

Isolation. Problem isolation is more complicated. Even when a software component reports an error, it may not be clear whether the fault lies with the hardware or the software. In the typical client/server environment, with many layers of software and many different hardware components, identifying exactly where a problem occurred is a challenging task. For instance, if the connection between the workstation and the host is lost, any number of components could be responsible. Possible sources of the problem include a LAN card malfunction, wiring discontinuity, LAN router failure, operator error, or application error.

The operations staff is typically responsible for problem isolation when the application cannot resolve a problem. The staff should establish a predefined process to methodically isolate problems. The following sample process illustrates how an error might be pinpointed, potentially saving the operations staff valuable time: First, a utility program determines if the LAN is operational. If the LAN passes that test, another utility program sends a message to the workstation hardware to determine if the workstation is communicating with the LAN. If this works, an application on the host could send an unsolicited message to the workstation (assuming the application supports unsolicited messages).

Depending on the type of network being used, there may be tools and facilities to help isolate the problem. Some LAN vendors, for instance, supply network management facilities that provide tracing capabilities and statistic collection (including number of retries and number of protocol errors).

When using RSC, extensive error information is available to the client programmer. An RSC call, TdsErrorInfo, can be used to classify an error according to its type, its severity (fatal, able to be retried, or configuration), and the subsystem reporting the error.

RSC reports connection errors, session errors, I/O errors, and application program errors. RSC reports on the following subsystems: RSC API, Pathway, Pathsend, Guardian 90 file system, NETBIOS, asynchronous communication, TCP/IP, and unsolicited messages. In addition, the TDP has a facility to trace specific client messages by using the TDSCOM interface.

Recording. Once a failure is identified, it must then be logged for future analysis. This information is useful in determining the state of the application, and analysis may detect possible conditions and trends. The RSC TDP process automatically logs all critical errors to the Guardian 90 $0 process.

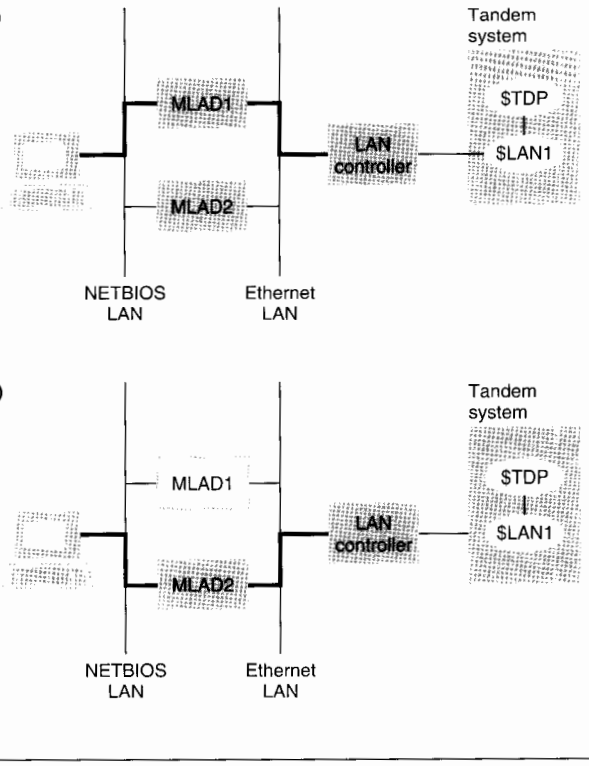
It is helpful to centralize the reporting of errors, for instance on a Guardian 90 system. For the workstation to log errors on the host, the LAN or WAN must be available. If the network is not available, the client should record pertinent errors locally and provide for a mechanism to send them to the host when the network becomes available.

In general, the client can send an error message to a central Guardian 90 location using the same RSC procedures as for any Guardian 90 server. Many Guardian 90 applications implement a centralized error server to simplify error handling in all other components of the application. It is a natural extension for a client to use the same mechanism.

Recovery. Recovery from a failure requires that a problem has been detected, isolated, recorded, and that a rule or action is defined and executed to bring the application to a known state. Successful recovery depends on establishing how failures should be handled. This includes deciding which system components are responsible for which failures. In most cases, the client application will initiate and resynchronize communications with the host servers.

In some cases, the network provides tools and facilities to aid in the recovery of an application. For instance, TCP/IP is a reliable protocol and will attempt to resend any packets that fail. Information pertaining to retries is subject to the configuration parameters associated with the communication subsystem.

Figure 3



RSC indicates if an error condition is fatal, able to be retried, or a configuration error. From the client programmer's perspective, errors that can be retried are the most pertinent. If the error is a communications failure, RSC provides facilities to reconnect to the same or a different TDP, through an alternate communication path. The client and TDP configurations must be set up to allow switching to an alternate path, and the client application must programmatically reestablish the connection and any outstanding sessions. (See Figure 3.)

Figure 3.
(a) Typical NETBIOS LAN environment, where both MLAD1 and MLAD2 are functioning. The path between the workstation client and the TDP is via MLAD1. (b) If MLAD1 fails, the client programmatically switches to MLAD2 and continues processing.

In addition to reestablishing connections and sessions, the client must also determine the outcome of any TMF transactions in progress at the time communications failed. Even though the TDP will automatically abort any uncommitted transaction, it is still possible that the TDP committed a transaction but was unable to reply to the client when the communication failure occurred. If the suspect transaction can be retried, the client simply reestablishes the connection and session and reissues the transaction. The possibility that the suspect transaction cannot be retried dictates that programmers include in the application a scheme to determine if the transaction was committed or not.

RSC Management

RSC is managed by TDSCOM, a program that runs at the Guardian 90 level. In addition to configuring objects, the TDSCOM process provides commands to start, stop, and get the current status of objects. These objects are TERM, NETNAME, TCPIPPORT, and ASYNCPORT. For operational purposes, TERM is configured to represent a workstation. Thus system managers can manage all RSC sessions on a given workstation by referencing the TERM object name. Named communication resources are configured on the Guardian 90 system to be referenced by the workstation component of RSC. For example, a NETNAME and a TCPIPPORT represent a shareable LAN resource, and ASYNCPORT represents an asynchronous connection, such as a dial-in port. Multiple copies of the TDP process can run on a single Tandem system in multiple processors independently. Each TDP controls and reports on its own objects.

Testing

RSC provides two facilities to aid in testing. The first helps to verify that the network is functioning properly and that RSC is configured correctly. The second aids a programmer in debugging an application. In this case, the TDP is capable of recording the events it processes for a given object. Event logging may range from all events to any combination of connection, session, and message. This allows programmers to see what steps the TDP is taking on their behalf.

Conclusion

Remote Server Call is a collection of high-level API calls that aid a client programmer in building an application that uses a workstation and a Tandem Guardian 90 system. RSC provides a facility for PC and UNIX clients to send messages to and receive messages from Guardian 90 servers, while maintaining transaction protection, data integrity, and OLTP performance. Configuration and management services are also provided, allowing for the control of hundreds of workstations.

The inherent complexity of the client platform leads to many issues associated with developing client/server applications. Client programmers are responsible for many tasks that are common to all client/server OLTP applications. These services are best placed in a client services layer that is used by all client programmers. This approach decreases the amount of code that must be tested and ensures the timely delivery of new functions for applications.

Client/server computing is a potentially rewarding way to satisfy end-user requests for corporate data. Using RSC, a client services layer, and appropriate client development tools, the application programmer can focus on fulfilling application requirements, improving user interfaces, providing value-added features and enhancements, and creating more satisfied users.

Michael Iem joined Tandem in 1987 and has presented client/server to Tandem users at Tandem's Application Technology Center. Currently, he delivers client/server consulting services. Mike has a B.S. degree in Industrial Engineering from Purdue University.

Terrye Kocher joined Tandem in 1982 as an account analyst. Since moving to the Tandem Customer Support and Services organization in 1988, she has done consulting on NonStop SQL, application development environment, and client/server computing.

Designing Client/Server Applications for OLTP on Guardian 90 Systems

The Pathway transaction processing system has enabled high-volume online transaction processing (OLTP) on Tandem™ systems. As high-volume OLTP applications migrate from conventional terminal-host to client/server architectures on Guardian™ 90 operating systems, programmers will require new guidelines for application design if end users are to realize such benefits of client/server processing as better user interfaces. Toward this end, Tandem has released an application development tool, Pathway Open Environment Toolkit (POET), that assists programmers in generating client programs that communicate with Pathway servers on Guardian 90 systems. The POET product, which requires Tandem's Remote Server Call (RSC) software to gain access to Pathway servers, currently supports construction of Windows 3.x clients.

This article considers client/server design, citing examples from the experience of developers and early users of POET. It describes the major design considerations associated with developing client/server applications for OLTP on Guardian 90 systems, and advises programmers on the transition from the conventional user-interface features of terminal-host designs to graphical user interface (GUI) features. The article also makes specific design recommendations for implementing servers and clients.

The reader should be familiar with the Guardian 90 operating system, the Pathway transaction processing system, and TMF™ (Transaction Monitoring Facility), and should have read the accompanying articles on client/server computing in this issue of the *Tandem Systems Review* (Cooperstein 1992; Iem and Kocher 1992).

Major Design Issues

When applying client/server technology to high-volume OLTP applications on Guardian 90 systems, there are many design issues to consider. The programmer must address these issues early in the design cycle because they have a profound effect on application structure and on how lower-level design decisions are made.

Client-Transaction Server Model

The first article in this issue of the *Tandem Systems Review* describes three client/server models that operate in the Guardian 90 environment (Cooperstein, 1992). Of these, one is required for high-volume OLTP: the client-transaction server model. In this model, application function is split between the client and server. The server application logic defines one or more services that the client may invoke. Only with this server configuration, called a Pathway server class, can server logic easily gain access to the following Guardian 90 fundamentals:

- Fault tolerance.
- High performance.
- Linear expandability.
- Data integrity.
- Manageability.

Fatter Clients and Leaner Servers

When designing terminal-host Pathway systems, programmers traditionally placed much of the application logic, possibly including some presentation logic, in the server. This philosophy was necessary to achieve good performance because interpreted Pathway SCREEN COBOL (SCOBOL) requesters executed more slowly than did compiled servers. However, this design principle no longer applies when considering client-transaction server computing. Because client computing resources are relatively inexpensive and much more responsive to the end user, it makes sense to locate all application presentation logic in the client. Also, depending on the application, it may be appropriate to place some data manipulation functions in the client. Thus, servers become leaner because only the pure transaction-processing logic resides in them, and clients become fatter. The result is a more balanced, or peer-to-peer, application architecture.

Placement of Data

Data has widely varying frequencies of change in typical OLTP applications. Some data, tax tables in a payroll system, for example, changes very infrequently. Other data, such as department information in a payroll system, changes occasionally but usually not on a daily basis. And other data, such as orders in an order entry system, changes on a minute-by-minute basis. Unlike the traditional OLTP system where all data resides on the host, client/server computing permits distribution of certain data interactions to the client. By bringing appropriate data closer to the user, this distribution makes the application more responsive to the user and reduces server processing cycles.

The decision about where data interaction should take place rests largely on the static quality of the data. One should either download static data, such as tax tables, from the server at initial execution of the client or associate the data with the client program code as a configuration file. Somewhat more dynamic data should be downloaded to the client at initial client execution, then occasionally checked for validity. One should always access dynamic data directly from the server. Ultimately, update access should always result in critical data being stored on the host server; it is only there that data integrity can be guaranteed. Since the majority of the I/O requests in a typical OLTP system are for read access, implementing these recommendations will significantly improve the performance and usability of a client/server system.

High-Value Data Messages

Not all OLTP applications can utilize high-speed local area networks (LANs); therefore, efficient use of a low-speed communication network may be an important design goal. However, even if the application is to use a LAN, reducing the size and frequency of data messages between the client and server will result in better utilization of the LAN and lower server overhead.

An important step toward achieving efficient use of the communication system is to design *high-value* data messages. This means that the application design should minimize both the size and the frequency of data messages sent between the client and server. One may achieve size reduction by filtering data on both sides so that only the data necessary to complete the transaction is sent. Reductions in frequency can be achieved by using the maximum data transfer size when sending multiple rows of data and by implementing the data access recommendations given above.

Graphical User Interfaces

Because GUIs form a superset of terminal user interfaces (TUIs), it is possible to faithfully replicate TUIs in a client/server system. Although it may seem attractive in some ways, replicating TUIs results in an application that gains few of the benefits of client/server computing (Gartner, 1992). Only when the full power of GUIs is exploited can the benefits of client/server computing be realized by the end user.

Tiered Architectures

In traditional OLTP systems, the architecture was typically two-tiered: terminals were attached to mainframe hosts. While it was not unusual to have concentrators and minicomputers placed between the terminal and host, the logical design of the applications still appeared two-tiered to the programmer.

With client/server computing, it is possible to have three or more processing tiers. (See Figure 1.) Three-tiered architectures may be attractive when work-group or cell computing also requires access to corporate systems. The tiered structure simplifies software distribution, connects heterogeneous devices, and facilitates work group interactivity without affecting the host. However, as the number of tiers increases, the complexity of the design and programming tasks also tends to increase.

Local Processing for Batch-Like Transactions

In some OLTP applications, it is necessary to build up a large amount of data, referred to as *context*, before it becomes possible to process a particular transaction. To a certain extent, these types of transactions resemble small batch-processing jobs. An example would be an order with many line items where the user might wish to access the host to build the order, but not submit it for further processing until after receiving the approval of another user.

In terminal OLTP systems, one either held the context for this type of transaction in the database in some special state or accumulated it in a temporary workspace and then applied it to the database. With client/server computing, batch-like transactions can accumulate context on the client before they are submitted to the server; the latter takes place when the user is ready to commit the transactions to the database. This architecture increases responsiveness to the user and reduces server processing cycles.

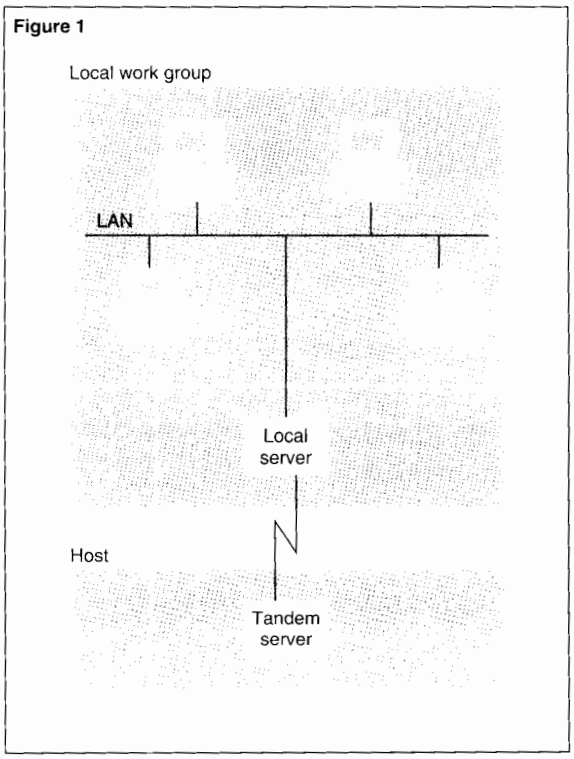


Figure 1.
Three-tiered client/server computing.

The name service in POET is an example of this client/server design principle. It is a classical database maintenance program that reads a database into the client and performs maintenance locally. After editing the database, the user commits it back to the server. Processing cycles are thus optimized on both the client and server. Of course, with this approach, one must protect against concurrent update of the database by separate clients.

Figure 2

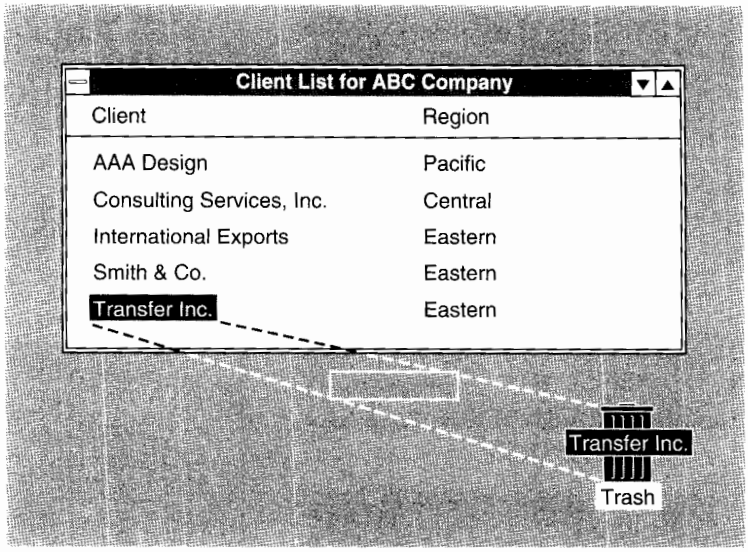


Figure 2.
Drag-and-drop operations.

Proper Client Development Tools

The client development tool can have a significant impact on the productivity of the development team and the ability to implement specific types of clients. High-level client development tools (for example, Microsoft's VisualBasic, PowerSoft's PowerBuilder, and Borland's ObjectVision) may significantly increase programmer productivity. They may, however, consume more resources or provide less flexibility in creating user interfaces than low-level client development tools such as C, C++, and CASE:W. The latter offer performance benefits and provide full access to the power of the GUI, but are not as productive for the programmer to use. Careful evaluation of these trade-offs must be made before choosing a tool.

POET addresses development tools at both levels. It can interface with any client development tool that supports Dynamic Link Library (DLL) calls. Among these are all the tools mentioned above.

New User-Interface Models

Probably the most challenging design issue facing programmers moving to client/server computing is the implementation of GUIs. Among the chief benefits of GUIs are consistent user interfaces, error forgiveness, and dynamic status feedback; in addition, GUIs are user-centered.

The programmer who is unfamiliar with GUIs should consider contracting for human-factors engineering services. GUIs offer so many options for presenting data that the choices may overwhelm most application designers. Fortunately, there are a number of user-interface standards, such as Common User Access (CUA) from IBM, that can help the programmer achieve good GUI design. However, these standards do not address all of the user-interface design issues associated with moving to client/server computing from traditional OLTP. The programmer must find ways to transform many traditional TUI functions into GUI implementations.

Drag-and-Drop Operations for Updating Databases

In traditional OLTP systems, pressing one or more function keys updates the database. Although one can replace function keys with GUI *push buttons*, a more intuitive approach is to implement *drag-and-drop* operations between objects. Examples include deletion of records by dragging the records to a trash-can icon (see Figure 2) or maintenance of a many-to-many database relationship by dragging one object to a list associated with another object.

Push Buttons for Function Key Actions

Function keys initiate host transactions in many terminal OLTP systems. In GUI environments, push buttons may replace function keys. Push buttons indicate action in a conceptually simpler fashion, and they also may be self-documenting; that is, the button's name may be the same as its function. A caveat is in order, however, when considering the replacement of function keys with push buttons: too many push buttons may confuse the end user.

Multiple Windows Replacing Single Terminal Screens

In a terminal OLTP system, one can typically initiate many user functions without leaving one screen. While this is an efficient programming approach, it results in screens that are neither simple to operate nor easy to understand. Frequently, fields on the screen do not correspond to the type of transaction the user wants to perform.

With GUIs, however, one may replace these function-heavy screens with several simple windows. When the user requests a function from the menu, the window that appears exactly matches the requirements of the function. (See Figure 3.) This methodology greatly reduces the number of push buttons necessary to initiate commands in a window. Typically, a simple window will require only OK and Cancel push buttons.

Radio Buttons for Mutually Exclusive Field Choices

Often, in terminal OLTP systems, the user encounters screen fields that will accept several values, each of which may be valid. The user then has to select from a small list of codes displayed next to the field or navigate to another screen and then select a value. With GUIs, one might have a group of mutually exclusive *radio buttons* instead. Here, the user would view each of the possible choices accompanied by descriptive text. (See Figure 4.)

Figure 3

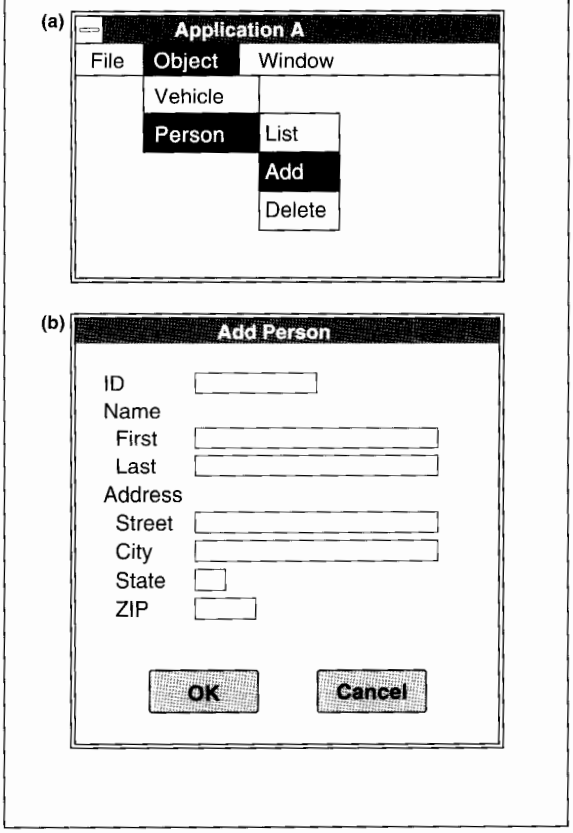


Figure 3.
Selecting Person and then Add in the Application window (a) brings up the Add Person window (b).

Figure 4

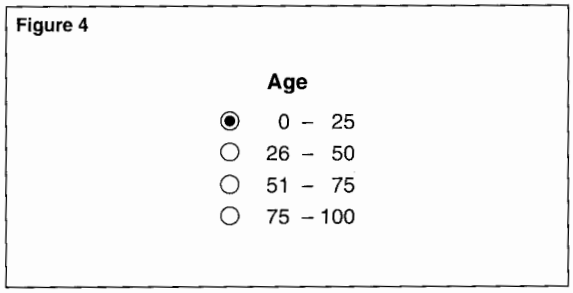


Figure 4.
Mutually exclusive radio buttons.

Figure 5.
Check boxes have two clearly distinguishable states: for example, ON and OFF.

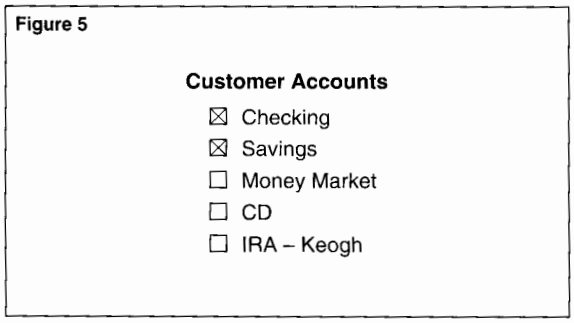


Figure 6.
List boxes contain lists of objects or settings that the user can select.

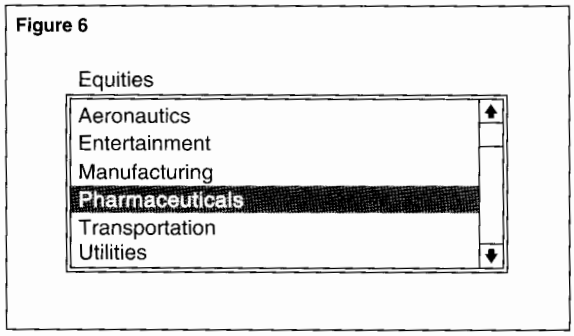
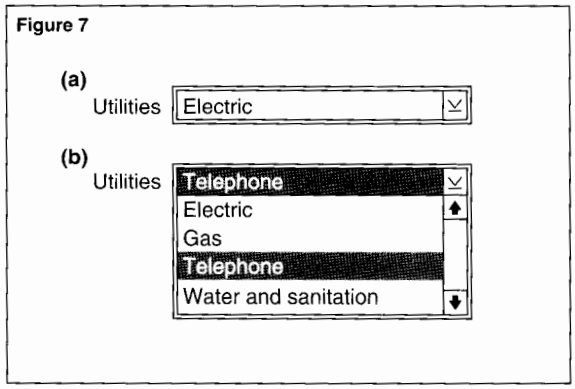


Figure 7.
A drop-down list displays only one item (a) until the user acts to display other objects or choices (b).



Check Boxes for Boolean Data Fields

Frequently, in terminal OLTP systems, many screen fields present Boolean choices that require the user to enter Y or N, or to select a field by typing an X. With GUIs, it is more appropriate to use *check boxes*. In the latter case, the user selects one or more Boolean options by simply clicking on the options. (See Figure 5.)

List Boxes for Browsing Through Field Values

In well-designed TUI systems, users can list valid values for an entry field and then select a value from the list. This is usually done by navigating to a screen where there is a list of values, then pressing a select-and-return function key. While this method adds to the friendliness of the application's interface, it is often slow and it tends to increase the overall processing load on the host.

With GUIs, one can make this operation much faster and more intuitive through the use of *list boxes*. The list box is part of the window, and a field is selected without removing the rest of the window. One can use several list box variations for this purpose. In CUA, the standard list box has a fixed size. The user may scroll through the list and select a choice. (See Figure 6.) Although effective, this object does consume a fair amount of pixel space in the window.

A more compact list box in CUA is the *drop-down list box*. The current selection in the list displays as a normal edit field. Before selecting from a list, the user presses a button that causes the list of selections to descend. (See Figure 7.)

To facilitate making a choice from a predefined list or typing a value into a field, CUA provides an object called the *combination box*. (See Figure 8.) The combination box functions much like the drop-down list, but allows the user to type characters into the selection field.

Notebooks for Multiple-Page Screens

Because the standard 80-character by 24-line terminal screen provides little space for data fields and text, it is common for the logical application object to contain more attributes than the display medium can handle at one time. TUI designers create multiple-page screens to handle this problem. Although the basic GUI model does offer some relief in this area (for example, smaller fonts and larger display space), the display medium is still limited and the attributes of a large application object may not all fit in one window. With CUA, the user views an object that resembles a bound notebook. (See Figure 9.) Clicking on the tabbed divider pages makes available all the attributes of a large application object. Push buttons outside the notebook initiate actions on the entire contents.

Server Design Recommendations

The idea of dividing application logic into two separate processes is well known and understood by Tandem users familiar with Guardian 90 architecture. Requester-server processing, basic to Guardian 90 architecture, is conceptually very similar to client/server computing. Thus, designing servers for client/server applications resembles, although is not exactly the same as, designing servers for requester-server applications.

In the Pathway requester-server design model, the sole function of the server is usually to process transaction requests from one or more SCOBOL requesters. Thus, one tailors servers to meet the specific needs of requesters. With the advent of client/server computing, and the future clearly heading toward true distributed processing, servers must now become generic service providers to many potential consumers. One must design servers that are free of any bias toward a particular device or computer. Nevertheless, servers must present a protocol that is flexible enough to meet the needs of all consumers. The impact on server design is not trivial: data messages must be flexible in size, services must defend themselves against unauthorized access, servers must check data message version numbers, and servers may have transaction boundaries.

Large Data Messages With Variable Length

As already mentioned, the terminal OLTP screen is not able to display more than 80 x 24 or 1,920 characters. Also, performance of the Pathway terminal control process (TCP) suffers when a requester stores too much terminal context, or data. Thus, the size of data messages between requesters and servers is usually limited to several thousand characters. The table list screen is an example of this design; the screen characteristically displays only 8 to 12 rows from the database at a time, resulting in size limitations on the data message from the server.

With client/server computing, this type of restriction can be lifted because clients have large local-data storage capacities. A server that passes hundreds of rows from the database to the client for temporary storage or manipulation might be desirable from a design standpoint. In this case, creating servers that transfer as much data as the communication line can handle at one time would be sensible. It might also be reasonable to allow variable-length messages because some requests might be smaller than the maximum message size. Exploiting large message sizes greatly increases the speed and efficiency of all parts of the system. In the case of POET and RSC, the maximum data message size is 32,000 bytes.

Figure 8

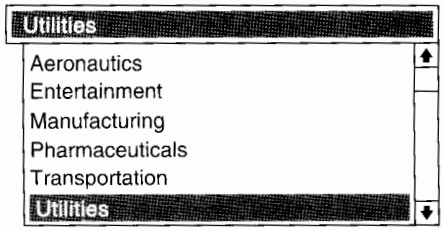


Figure 8.
A combination box contains a list of objects or settings that the user can scroll through and select to complete the entry field.

Figure 9

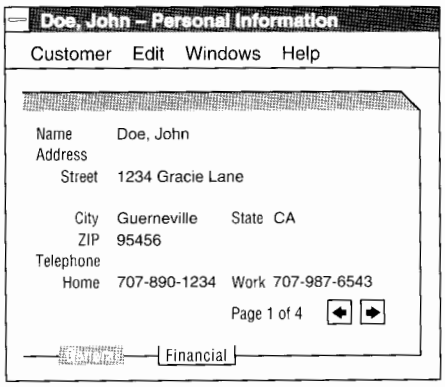


Figure 9.
The notebook box allows the user to turn the pages of the notebook and to move from one section to another.

Server Security

In most terminal-based Pathway systems, the application's first line of defense is the requester. Except for the single security server that maintains the user access profiles, security is not an issue for normal transaction servers. Rather, it is the responsibility of the requester to filter unauthorized access and provide basic data validation. With requester functions now moving into unsecured workstation clients, the application's first line of defense becomes the server. Although most client users are incapable of programming a workstation to send unauthorized messages to a server, the possibility is a source of concern. Given the rise in computer crime over the past few years, undefended servers are sure to become an attractive target.

Client/server computing requires a more generic but flexible server design.

There are a number of ways to defend servers. One can configure RSC to require user authentication through a security server. Although this method limits system access, it does not limit specific application-server access. Thus, a user who has logged on through RSC for access to a particular system could in principle gain access to any other server in the system.

Another way to defend servers is to have the server authenticate each transaction; this could be done by requiring the user ID and encrypted password as part of the data message. Servers could call a security server to validate each new user, then place the user ID in a memory-cached table for validation of subsequent messages. In this way, processing overhead for authentication would be kept to a minimum.

Checking Data-Message Compatibility

In Pathway systems, a central operations group usually manages the requester and server components of an application. Therefore, the chances of a version mismatch are fairly low. However, the probability of a mismatch in client/server computing is significantly higher. Mismatch detection is necessary even with a host-controlled, workstation-software-distribution system like ANNATEK's Network Navigator because the end user has ultimate control over the software.

The preferred method of ensuring version compatibility between clients and servers is to send a client version number along with the data message to the server. The server then decides whether it can process messages for that version of the client. If it can, it returns a reply that matches what the client is expecting. If it cannot, it returns an error message and rejects the request. This method permits the simultaneous existence of multiple versions of a client in a network. It also allows for flexibility in the distribution of new applications because not all clients are required to change to the new version at the same time.

Transaction Boundaries in the Server

In Pathway applications, initiation and control of logical transaction boundaries traditionally occur in the requester. Because processing of a business transaction may require the requester to access multiple servers, this model is appropriate. Usually, a single programmer builds the requester-server pair to process a business transaction, so spreading transaction boundaries across multiple servers is common practice. Due to the enhanced complexity of client vis-a-vis requester programming, many information systems organizations will probably choose separate client and server programmers. Thus, programming of a business transaction may require the coordinated efforts of two or more programmers.

With client/server computing, one could consider placing transaction boundaries in the server. This would reduce the need for programming coordination and simplify client programming. However, a caveat is in order when considering this approach. Servers with TMF transaction boundaries cannot participate in other TMF transactions (nested transactions); thus, server reuse is limited. Moreover, these servers will not be able to engage in the heterogeneous network transactions that will be possible in the distributed computing environments of the future.

Optimized Data Message Content

Efficient transport of data messages between requesters and servers is seldom a high priority in traditional terminal-based Pathway systems. There, the data typically moves from process to process, either through memory or through the interprocessor bus, both of which are very high bandwidth connections. With client/server computing, the connection between processes may have a much lower bandwidth. One could have, for example, a 2400-baud asynchronous modem connection. In these systems, optimization of the amount of data sent between clients and servers is critical. This usually requires more flexibility in the size of reply messages that the server can return to the client.

Client Design Recommendations

Client design is one of the greatest challenges facing Tandem programmers who are moving to client/server computing. One can leverage very little of the Pathway requester programming model into building clients. The challenge includes designing applications for new hardware platforms, new programming environments, and new user interface models; there is also a need for a new design model.

Data Conversion

Data conversion is rarely an issue in terminal-based Pathway systems. Because all data resides on the host and most programming is in SCOBOL and COBOL, data incompatibility does not concern the programmer. With client/server computing, the issue becomes more complex. Not only do data incompatibilities due to hardware arise, but the likelihood of language data incompatibilities becomes greater. Processing capacity is least expensive at the client workstation, so the latter is the recommended place to perform data conversion. Hardware-level data conversion, usually in the form of byte swapping, will almost always be required. Because most client programming is in C and most server programming is in COBOL, extensive language-level conversion is usually necessary.

The POET product includes a large library of data-conversion routines that operate between Guardian 90 systems and PCs, and between the C and COBOL languages. POET also generates a default C function to convert an entire data message based on the message's Data Definition Language (DDL) description.

Input Editing and Validation

The accepted design principle in traditional Pathway systems is to force resource-intensive input editing and validation into the server. Client/server computing violates this principle; clients should now do as much as is practical to validate a business transaction before sending it to a server. This may involve importing portions of the database into the client for validation.

Programming Model

Requester programming in standard Pathway systems involves a fairly straightforward programming model. Neither concurrency nor multiprogramming is of concern to the programmer. With client/server computing, the client programming model is very different. Client programming now rests upon an object-oriented, event-driven model where both hardware and software initiate messages. These messages must be processed in a fairly short time if the desktop is to remain responsive to the user. Program operations now consist of small procedures that do not assume unlimited access to the CPU. These procedures execute in response to an event, then quickly return control to the processing cycle, usually by initiating another event.

In client/server programming, client programmers should continue to follow the event-driven programming model. The client should not wait for a response when making a request to the server. Rather, it should make the request and then allow the event generated by the server to continue processing. The remaining portions of the application and the desktop thus continue to process events, while remaining responsive to the end user. Adhering to this programming model becomes especially important in the case of designing Microsoft Windows clients, which represent non-preemptive, multitasking environments. There, if the client programmer does not adopt an event-driven style, the entire desktop will freeze up while waiting for the server to respond.

Client design poses one of the greatest challenges for Tandem programmers.

The POET product provides an event-driven application programming interface for communicating with Pathway servers. All POET server calls release control after sending a request and return events upon receiving the server's response. In the event-driven programming model, it is possible for the client user to initiate the same transactions multiple times. To prevent this from happening, POET provides functions that disable the active window in the client while a transaction is being processed. Meanwhile, the rest of the desktop is allowed to operate.

Modular Implementation

Terminal-based Pathway systems clearly permit modular requester construction. Each requester program is associated with a screen, and all of the requester programs are gathered into a library for execution. Modularity makes it possible to use large teams of requester programmers. With client/server systems, implementation of client modularity is less easy. Depending on the client tool, clients may comprise single monolithic programs, independent objects bound together into a single executable file or run module, or freestanding executable files. Given that a programming team will build most OLTP client/server applications, clients should consist of independent objects, either bound together or running as freestanding executable files. Thus, the client-development tool chosen should support this style.

The choice between single and multiple executable files rests in part on the following considerations. First, if the application employs low-speed communication lines, multiple executable files may be the preferred choice. This permits client software to be updated more quickly. However, a workstation software-distribution system is essential if this option is selected because there will be many client objects to manage. Second, if multiple executable files are used at run time, it may be wise to implement an intra-application version-checking procedure. This becomes especially critical when workstation software distribution is not automated. Finally, if the application uses high-speed communication lines and workstation software distribution will not be automated, the recommendation is to implement a single executable file at run time.

The value of the POET and CASE:W products together provide a client application development environment that meets all the modularity requirements of a large client programming team. Objects can be built independently and tested, then packaged together either as single or multiple executable files.

Help Facilities

The programmer traditionally designs a unique end-user help facility for each Pathway system. The Pathmaker™ online application generator provides a comprehensive help facility, but many Pathway systems were built before Pathmaker was available. With client/server computing, most client platforms provide a powerful help facility that the client programmer can extend. It is recommended that the programmer use the platform help facility rather than building one. This not only increases the programmer's productivity, but produces clients with consistently designed interfaces.

Preloading Objects

Because terminals in traditional Pathway systems do not store data locally, preloading objects at the user interface is not possible. With client/server computing, this is not only possible but desirable. For example, the programmer can dynamically load drop-down list boxes that present static data values to the user at start-up of the client from the server. This will eliminate significant network traffic by replacing host data access with local data access.

Instrumentation for Manageability and Testability

Well-designed terminal Pathway systems characteristically capture all error conditions and gracefully recover from them. This is valuable not only at run time, but also in testing. Client/server systems must now implement the same design principles. The need for instrumentation actually increases with client/server computing because more errors are possible due to the greater number of software products in use. The need for instrumentation also increases because two, separate, noncompatible computers are employed. At the very least the instrumentation must handle all error conditions and provide some ability to view data messages.

POET's instrumentation capabilities exceed those of RSC alone. Besides processing and displaying all RSC errors, POET can also view messages moving between clients and servers. This capability exists both in development and at run time.

Conclusion

Client/server computing represents the next major evolution in application technology. Although the transition to client/server computing for OLTP on Guardian 90 systems poses many challenges, employing Tandem's POET application development tool and implementing the design recommendations made in this article will significantly reduce the magnitude of those challenges.

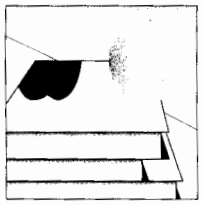
References

Cooperstein, H. 1992. An Overview of Client/Server Computing on Tandem Systems. *Tandem Systems Review*. Vol. 8, No. 3. Tandem Computers Incorporated. Part no. 89803.

Gartner Group, Inc. 1992. *Client/Server Architecture: A Management Perspective*. C/S: R-900-100.

Iem, M. and Kocher, T. 1992. Implementing Client/Server Using RSC. *Tandem Systems Review*. Vol. 8, No. 3. Tandem Computers Incorporated. Part no. 89803.

Bill Culman joined Tandem in 1984. After spending several years in the field supporting customers, he became the development manager for the Pathmaker and POET products. Before coming to Tandem, Bill worked for another large computer manufacturer.



Tandem Education

The following paragraphs provide highlights of the latest education courses offered by Tandem. To sign up for a class or to order an independent study program (ISP), users should call 1-800-621-9198. Full descriptions of all available courses and ISPs appear in the *Tandem Education Course Catalog* and on InfoWay.

C Programming on the Guardian Platform

This three-day course presents a solid introduction to the Tandem Guardian 90 implementation of ANSI-standard C programming. The course takes an intensive look at basic C syntax and I/O routines. A knowledge of C is a prerequisite.

Client/Server Concepts and Solutions

This four-day course covers the application of client/server principles in relation to Tandem systems. Students learn how to build simple application programs that give users the convenience of a point-and-click graphical interface, while retaining the Guardian 90 operating system's power as a high-performance, reliable, distributed transaction processing system.

Foundation of Windows Programming

In this five-day course, students gain a thorough understanding of Windows programming techniques. Numerous labs, extensive sample code, and interaction with the instructor give students practical, hands-on experience in the subject.

Integrity Architecture and NonStop UNIX Administration

This eight-day course focuses on the Tandem enhancements that differentiate the Integrity systems from other available UNIX systems. NonStop-UX (B00 release) is Tandem's implementation of the UNIX System V Release 4 operating system.

The Technical Information and Education department is an annotated list of new Tandem education courses, consulting and information services, and books Tandem is offering its users.

Interfacing C Programs to Guardian Processes

In this course, students spend two days getting a solid foundation in the interface of C and TAL programs under the Guardian 90 operating system. The course provides the tools needed to call C programs from TAL, as well as to call TAL programs from C.

Introduction to C Programming

This five-day course teaches students to use the C programming language effectively. Students learn to design large programs using C modularity features and to recognize tools that aid in developing, debugging, and maintaining multimodule C programs.

NonStop NET/MASTER Basic Operations

In this three-day course, students acquire the skills needed to perform daily NonStop NET/MASTER operations. The course includes hands-on exercises using the NonStop NET/MASTER Management Services application in an operational environment.

NonStop NET/MASTER Installation and System Management

In this two-day course, students learn how to install and manage a NonStop NET/MASTER system. The course teaches the skills needed to install, start up, and manage the NonStop NET/MASTER system.

NonStop NET/MASTER Network Control Language Programming

This five-day programming course provides a working knowledge of the NonStop NET/MASTER Network Control Language (NCL). The course emphasizes the skills needed to write NCL procedures.

NonStop UNIX Basics

In this five-day course, students acquire the skills needed to define the UNIX hierarchy; log in to a UNIX system and use commands; state the purpose of key system directories and files; and perform basic user administration functions. The course also provides an overview of NonStop-UX features and operations.

Remote Server Call

This two-day course offers a thorough introduction to Tandem's Remote Server Call (RSC) product and gives students a general understanding of cooperative processing between Tandem computers and PCs. The course details how RSC integrates PCs and other intelligent devices with Tandem's Guardian 90 operating system and Pathway system environments.

SeeView Programming

This independent study program introduces concepts and techniques used in script development. Users learn basic SeeView programming concepts. In addition, programmers develop a sample Guardian 90 utility interface system using proper screen design techniques.

UNIX System V Release 4 Administration

This five-day course focuses on the major functions of UNIX administration including one-time-only operations and day-to-day routines. Differences between Release 3 and Release 4 of UNIX System V are also covered.

TandemSystemsReview*Index*

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

Volume 1, Number 1	Fall 1983	Part no. 83930
Volume 2, Number 1	Winter 1984	Part no. 83931
Volume 2, Number 2	Spring 1984	Part no. 83932
Volume 2, Number 3	Summer 1984	Part no. 83933

As of this issue, 19 issues of the *Tandem Systems Review* have been published:

Volume 1, Number 1	February 1985	Part no. 83934
Volume 1, Number 2	June 1985	Part no. 83935
Volume 2, Number 1	February 1986	Part no. 83936
Volume 2, Number 2	June 1986	Part no. 83937
Volume 2, Number 3	December 1986	Part no. 83938
Volume 3, Number 1	March 1987	Part no. 83939
Volume 3, Number 2	August 1987	Part no. 83940
Volume 4, Number 1	February 1988	Part no. 11078
Volume 4, Number 2	July 1988	Part no. 13693
Volume 4, Number 3	October 1988	Part no. 15748
Volume 5, Number 1	April 1989	Part no. 18662
Volume 5, Number 2	September 1989	Part no. 28152
Volume 6, Number 1	March 1990	Part no. 32986
Volume 6, Number 2	October 1990	Part no. 46987
Volume 7, Number 1	April 1991	Part no. 46988
Volume 7, Number 2	October 1991	Part no. 65248
Volume 8, Number 1	Spring 1992	Part no. 65250
Volume 8, Number 2	Summer 1992	Part no. 69848
Volume 8, Number 3	Fall 1992	Part no. 89803

The articles published in all 23 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 *(cont.)*					
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	
83936Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
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
CUSTOMER SUPPORT					
Customer Information Service	J. Massucco	TSR	3,1	March 1987	83939
Remote Support Strategy	J. Eddy	TSR	3,1	March 1987	83939
Tandem's Software Support Plan	R. Baker, D. McEvoy	TSR	3,1	March 1987	83939
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
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 or 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
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
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DATA MANAGEMENT *(cont.)*					
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
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	1369353
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
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
MANUALS/COURSES					
B00 Software Manuals	S. Olds	TSR	1,2	June 1985	83935
C00 Software Manuals	E. Levi	TSR	4,1	Feb. 1988	11078
New Software Courses	M. Janow	TSR	1,2	June 1985	83935
New Software Courses	J. Limper	TSR	4,1	Feb. 1988	11078
Subscription Policy for Software Manuals	T. McSweeney	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,2	June 1986	83937
OPERATING SYSTEMS					
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
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 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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
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
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
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 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
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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PERFORMANCE AND CAPACITY PLANNING *(cont.)*					
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
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
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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
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
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
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
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848
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	839393

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					
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
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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DSM *(cont.)*					
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
FASTSORT					
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938
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					
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
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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
GUARDIAN 90 *(cont.)*					
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
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 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

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
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	89803
PS MAIL					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939
RDF					
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848
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
TRANSFER					
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
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
MISCELLANEOUS[1]					
A Performance Retrospective	P. Oleinick	TSR	2,3	Dec. 1986	83938
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
Batch Processing in Online Enterprise Computing	T. Keefauver	TSR	6,2	Oct. 1990	46987
Buffering for Better Application Performance	R. Mattran	TSR	2,1	Feb. 1986	83936
Capacity Planning Concepts	R. Evans	TSR	2,3	Dec. 1986	83938
Configuring Tandem Disk Subsystems	S. Sitler	TSR	2,3	Dec. 1986	83938
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Credit-authorization Benchmark for High Performance and Linear Growth	T. Chmiel, T. Houy	TSR	2,1	Feb. 1986	83936
Data-window Phase-margin Analysis	A. Painter, H. Pham, H. Thomas	TSR	2,2	June 1986	83937
Estimating Host Response Time in a Tandem System	H. Horwitz	TSR	4,3	Oct. 1988	15748
Getting Optimum Performance from Tandem Tape Systems	A. Khatri	TSR	2,3	Dec. 1986	83938
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
New Software Courses	M. Janow	TSR	1,2	June 1985	83935
New Software Courses	J. Limper	TSR	4,1	Feb. 1988	11078
Optimizing Batch Performance	T. Keefauver	TSR	5,2	Sept. 1989	28152
Optimizing Sequential Processing on the Tandem System	R. Welsh	TJ	2,3	Summer 1984	83933
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
Peripheral Device Interfaces	J. Blakkan	TSR	3,2	Aug. 1987	83940
Predicting Response Time in On-line Transaction Processing Systems	A. Khatri	TSR	2,2	June 1986	83937
Remote Support Strategy	J. Eddy	TSR	3,1	March 1987	83939
Streaming Tape Drives	J. Blakkan	TSR	3,2	Aug. 1987	83940
Subscription Policy for Software Manuals	T. McSweeney	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,2	June 1986	83937
Tandem's Software Support Plan	R. Baker, D. McEvoy	TSR	3,1	March 1987	83939
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 Performance Characteristics of Tandem NonStop Systems	J. Day	TJ	1,1	Fall 1983	83930
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	3,2	Aug. 1987	83940

[1]This category is composed of articles that contain product information but are not specifically product-related.

TandemSystemsReview*OrderForm*

The *Tandem Systems Review* is published quarterly. Subscriptions are $75 per year; single copies are $20 each.

☐ I am a current Tandem customer. Please invoice me for the requests on this form.

My Tandem sales representative is _____

☐ I am enclosing a check or money order for the requests on this form. (Make check 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

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 more information or to place your order by telephone, call: 1-800-473-5868

Send this form to:
Tandem Computers Incorporated
Tandem Systems Review, Loc 216-05
18922 Forge Drive
Cupertino, CA 95014-0701

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

Menu sequence: Marketing Information→ Literature Orders→ Technical Marketing Pubs→ Tandem Systems Review

▲ 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 216-05
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95015–9862

▼ 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

An Overview of Client/Server Computing on Tandem Systems

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

Implementing Client/Server Applications Using Remote Server Call

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

Designing Client/Server Applications for OLTP on Guardian 90 Systems

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

Technical Information and Education

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 ☐ Other_____

3. Your title or position:

27 ☐ President, VP, Director 28 ☐ Systems analyst 29 ☐ System operator

30 ☐ MIS manager 31 ☐ Software developer 32 ☐ End user

33 ☐ Other _____

4. Your association with Tandem:

34 ☐ Tandem customer 35 ☐ Tandem employee 36 ☐ Third-party vendor 37 ☐ Consultant

38 ☐ Other _____

5. Comments

_____

_____

_____

_____

NAME

_____

COMPANY NAME

_____

ADDRESS

▲ 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 216-05
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95015–9862

▼ FOLD

▼ FOLD

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599

Part No. 89803

10/92 Printed in USA